BENERAG: BENEFIT SUPERVISION FOR BLACK-BOX SELECTIVE RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Answer correctness alone does not determine whether retrieval will improve a language model's response. Retrieval may leave an incorrect answer unimproved when it provides insufficient evidence. We study black-box selective retrieval through *protocol-conditional context benefit*, the improvement obtained by executing the deployed retrieval-augmented generation (RAG) branch. BeneRAG learns from paired branch outcomes offline and predicts whether retrieval will improve the current answer using only the question, sampled no-retrieval answers, and verbalized confidence. With observations, architecture, and training held fixed, benefit supervision yields higher F1 than correctness supervision at every tested budget in exact-budget ranking evaluations, gaining 3.6 F1 points at a 50% invocation rate. A question-only benefit estimator also outperforms the full-observation correctness estimator at these budgets. Across three QA datasets with controlled evidence availability, the policy improves answer quality over ALWAYS RAG while invoking retrieval less often. Automatic semantic evaluation with GPT-4.1 also favors BeneRAG over ALWAYS RAG. Benefit ranking remains useful under corpus and generator changes, but quality gains depend on the protocol and operating point. Acquiring the observations also incurs additional execution cost. These results support supervising the incremental benefit of the available RAG branch when allocating a limited retrieval budget.

1 INTRODUCTION

Retrieval-augmented generation (RAG) improves knowledge-intensive generation by conditioning a language model on retrieved evidence (Lewis et al., 2020; Ram et al., 2023). Retrieved context, however, is not uniformly helpful. It can be redundant when the model can already answer the question, and incomplete or conflicting evidence can turn a correct answer into an incorrect one (Amiraz et al., 2025; Wang et al., 2025). An incorrect answer may also remain incorrect when retrieval provides no usable evidence. Whether to retrieve therefore depends on the improvement the available RAG branch can provide, which answer correctness alone does not determine.

Adaptive retrieval methods use token-level generation signals (Jiang et al., 2023; Su et al., 2024) or hidden representations (Yao et al., 2025) to identify uncertain content and determine when additional evidence is needed. Other approaches train reflection tokens to control retrieval (Asai et al., 2024). These methods require access to internal signals or the ability to train the generator, which may be unavailable when a hosted model exposes only a text interface. Black-box methods instead use entity popularity (Mallen et al., 2023), learned query features (Marina et al., 2025), or inconsistency across repeated generations (Ding et al., 2025). Generation outcomes also provide supervision. Adaptive-RAG chooses among no retrieval, single-step retrieval, and iterative retrieval using labels derived from branch success and dataset-level complexity assumptions (Jeong et al., 2024). Uplift-RAG uses document-level quality improvements to rerank passages after retrieval (Qu et al., 2025). We instead consider whether executing an entire RAG branch will improve the current answer, with the decision made before retrieval.

We define protocol-conditional context benefit as the difference in answer quality between executing the deployed RAG branch and retaining the current no-retrieval answer. This quantity depends on the generator, retrieval procedure, context construction, and decoding scheme. The same question can therefore have different benefit under different execution protocols. A correctness label does not

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.

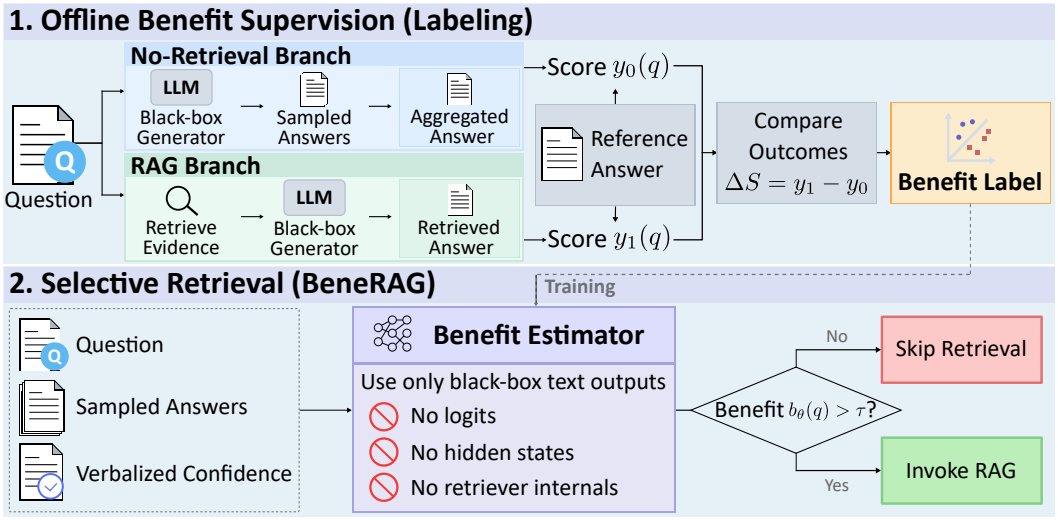


Figure 1: **Overview of BeneRAG.** Offline, paired branch outcomes define the context benefit $\Delta_S(q) = y_1(q) - y_0(q)$ and the supervision label $z_{\text{ben}}(q)$ (top). At deployment, the estimator reads only the query, sampled no-retrieval answers, and a verbalized self-assessment, and invokes the RAG branch when the predicted probability of $\Delta_S(q) > \varepsilon$ exceeds a validation-selected threshold (bottom).

specify this difference. We ask whether learning benefit from pre-retrieval observations allocates a limited retrieval budget better than learning answer correctness.

BeneRAG (Figure 1) learns whether benefit exceeds a fixed margin from paired branch outputs scored against reference answers offline. At deployment, a text encoder estimates that probability from the question, sampled no-retrieval answers, and verbalized self-assessment. A validation-selected threshold determines whether to invoke the RAG branch or retain the aggregated no-retrieval answer. The decision uses no model-internal signals, retrieved passages, or reference answers, and leaves the generator and retriever unchanged. We compare benefit and correctness supervision using the same observations, architecture, and training procedure.

We evaluate on a pooled 10,000-question test set from SQuAD, HotpotQA, and AdversarialQA under controlled evidence availability. Each query-specific candidate pool contains supporting evidence and distractors, but the evidence need not survive retrieval. The correctness-supervised estimator reaches correctness AUROC 0.8714, yet its benefit AUROC is only 0.5671. Benefit supervision reaches benefit AUROC 0.7498 from the same observations. At identical retrieval budgets, benefit supervision yields higher F1 at all three tested rates, reaching 0.4023 against 0.3664 at a 50% budget. A question-only benefit estimator also outperforms the full-observation correctness estimator at each budget. The validation-selected policy improves pooled F1 from 0.4000 for ALWAYS RAG to 0.4149 while invoking retrieval on 58.3% of queries.

Automatic semantic evaluation with GPT-4.1 also favors BeneRAG over ALWAYS RAG. Shared-corpus retrieval and transfer to five unseen generators further test the scope of the result. In the shared-corpus setting, threshold recalibration and retraining recover the F1 advantage lost by the frozen policy. Across generators, benefit ranking transfers, but neither the source threshold nor target-label recalibration preserves an F1 advantage on three targets. Acquiring the observations also costs more than question-only invocation in the measured execution comparison. Fewer retrieval calls therefore do not by themselves imply lower execution cost.

- **Benefit supervision improves retrieval allocation.** BeneRAG learns from paired offline outcomes to predict whether the deployed RAG branch improves the current answer using only pre-retrieval black-box observations. With observations, architecture, and training held fixed, benefit supervision yields higher F1 than correctness supervision at every tested retrieval budget.
- **Semantic evaluation and conditions for deployment.** On queries where BeneRAG skips retrieval, GPT-4.1 assigns higher strict and lenient semantic accuracy to the retained answers

than to their RAG alternatives. Benefit ranking transfers, but F1 gains remain protocol dependent. Shared-corpus adaptation restores an F1 advantage over ALWAYS RAG, whereas utility-based recalibration leaves F1 below this baseline on three target generators. Acquiring observations also costs more than question-only invocation in the measured comparison.

2 RELATED WORK

Adaptive retrieval has been studied as a mechanism for deciding when external evidence should be introduced into language-model generation. A major line of work relies on model-internal or model-specific signals. FLARE and DRAGIN use token-level generation confidence or information-need signals to trigger retrieval (Jiang et al., 2023; Su et al., 2024), while Probing-RAG and SeaKR exploit hidden representations to estimate retrieval need (Baek et al., 2025; Yao et al., 2025). Query-level uncertainty estimation derives a pre-generation signal from self-evaluations across internal layers and tokens (Chen et al., 2026), and Self-RAG trains specialized reflection tokens that jointly control retrieval, generation, and critique (Asai et al., 2024). These methods apply when the generator can be inspected or retrained; BeneRAG assumes neither.

Black-box selective retrieval uses information observable outside the generator. Entity popularity (Mallen et al., 2023) and model-independent query features (Marina et al., 2025) decide from the question alone, SCAAR uses the semantic importance of generated words to adjust retrieval timing and construct queries (Lin et al., 2025), and Rowen triggers retrieval when responses generated in different languages or by different models disagree (Ding et al., 2025). These signals serve as retrieval criteria in themselves. BeneRAG instead uses sampled responses and self-assessment as inputs to a supervised estimator, and obtains them from the same target generator without translation or an auxiliary model. Because uncertainty estimators alone can match more elaborate adaptive pipelines on QA (Moskvoretskii et al., 2025), we also report training-free agreement and confidence triggers built from these same observations.

Generation outcomes have also been used to supervise retrieval decisions. Adaptive-RAG trains a question classifier to choose among no retrieval, single-step retrieval, and iterative retrieval; its labels combine the success of these strategies with dataset-level assumptions about question complexity (Jeong et al., 2024). That classifier reads the question alone, whereas our estimator also reads the answers the generator has already produced. Uplift-RAG measures the difference in answer quality with and without a retrieved document and uses this signal to train a document reranker (Qu et al., 2025). Its predictions require the candidate passages, so the decision follows retrieval. BeneRAG applies a related notion of incremental benefit to an entire RAG branch and predicts whether invoking that branch will improve the current answer before any passage is retrieved. We hold the estimator inputs, architecture, and training procedure fixed when comparing benefit and correctness supervision, which isolates the effect of the learning target on retrieval allocation.

The observations we use come from work on black-box reliability estimation. Agreement across sampled generations supports answer selection and factuality assessment (Wang et al., 2023; Manakul et al., 2023), semantic uncertainty and entropy group responses by meaning (Kuhn et al., 2023; Farquhar et al., 2024), and verbalized confidence provides an explicit self-assessment of answer reliability (Tian et al., 2023; Xiong et al., 2024). We reuse these observations but supervise them against measured branch benefit rather than answer reliability, and our input ablations separate their contribution from that of the supervision target.

3 BENERAG: CONTEXT-BENEFIT-GUIDED SELECTIVE RETRIEVAL

BeneRAG treats selective retrieval as a prediction problem, and its central design choice is the prediction target: rather than treating query difficulty, no-retrieval incorrectness, or uncertainty as retrieval necessity, it learns from the measured improvement produced by the deployed RAG branch.

Figure 1 summarizes the resulting separation between offline supervision and deployment. Offline, paired no-retrieval and RAG outcomes define the protocol-conditional context-benefit target; at deployment, BeneRAG estimates that target using only pre-retrieval black-box observations before deciding whether to invoke the RAG branch. The following subsections formalize the benefit target, its black-box estimation, and the resulting invocation rule.

3.1 PROTOCOL-CONDITIONAL CONTEXT BENEFIT

For a query q, the no-retrieval branch produces K sampled responses $A^{(0)}(q) = \{a_1^{(0)}, \ldots, a_K^{(0)}\}$ and aggregates them into a single candidate $a_0(q)$; in our canonical setting, $K = 3$ and $a_0(q)$ is the normalized majority-vote answer. The deployed RAG branch retrieves evidence under its fixed retrieval and context-construction procedure and generates a context-conditioned answer $a_1(q)$. Let S denote the complete execution and evaluation protocol—the generator, retrieval procedure, available evidence, context construction, prompting, decoding, and answer scoring. Given the reference-answer set $G(q)$, the branch scores are

$$y_j(q) = \mathrm{Score}_S(a_j(q),\, G(q))\,, \quad j \in \{0, 1\}, \tag{1}$$

where Score_S is normalized token-level F1 in our canonical experiments.

We define the *protocol-conditional context benefit* as the measured improvement obtained by replacing the no-retrieval answer with the output of the deployed RAG branch:

$$\Delta_S(q) = y_1(q) - y_0(q). \tag{2}$$

A positive value indicates that the deployed branch improves the current answer under S; a negative value indicates that invoking it reduces answer quality. Importantly, $\Delta_S(q)$ is not an intrinsic property of the query: the same question can have different context benefit when any component of S changes. A difficult or uncertain question therefore does not necessarily require retrieval, while a confidently answered one may still benefit from it.

Because arbitrarily small score differences need not correspond to meaningful improvements, we construct the binary benefit target

$$z_{\mathrm{ben}}(q) = \mathbb{I}[\Delta_S(q) > \varepsilon]\,, \quad \varepsilon = 0.05, \tag{3}$$

with the margin ε held fixed across all datasets (Table 12). Offline label construction requires paired branch outputs and reference answers. At decision time, the no-retrieval samples and their aggregated candidate are already available; the estimator has no access to the RAG output, reference answers, or reference-based branch scores.

3.2 OBSERVABLE BENEFIT ESTIMATION

Because $\Delta_S(q)$ cannot be measured without executing the RAG branch, BeneRAG estimates whether the benefit exceeds the margin using only pre-retrieval observations. For each query, the available observation is

$$O(q) = \Big(q,\, A^{(0)}(q),\, c_{\mathrm{conf}}(q)\Big)\,, \tag{4}$$

where $c_{\mathrm{conf}}(q)$ is a verbalized self-assessment elicited from the same generator after producing the no-retrieval samples. Importantly, this confidence is not an assessment of the question's general difficulty or of the aggregated majority-vote answer $a_0(q)$. Instead, the generator is shown the query together with its first sampled no-retrieval answer $a_1^{(0)}(q)$ and is asked to output a single scalar in $[0, 1]$ indicating how likely that specific answer is to be correct; the response is generated greedily with at most eight new tokens. We retain the shorthand $c_{\mathrm{conf}}(q)$ for readability, although the elicited value is conditioned on both q and the realized sample $a_1^{(0)}(q)$. Because this signal is answer-specific, it is re-elicited whenever the no-retrieval samples are regenerated. All elements of $O(q)$ are obtained through ordinary text generation; the estimator receives no retrieved passages, RAG outputs, reference answers, branch scores, token probabilities, hidden states, gradients, or model parameters.

We serialize $O(q)$ into a single text sequence and learn a benefit estimator

$$b_\theta(q) = P_\theta(z_{\mathrm{ben}}(q) = 1 \mid O(q)) = P_\theta(\Delta_S(q) > \varepsilon \mid O(q))\,. \tag{5}$$

The score estimates the probability that the improvement exceeds ε, rather than the magnitude of the improvement. The estimator is a pretrained DeBERTa-v3-base encoder that reads the serialized observation directly as text, followed by mean pooling and a lightweight prediction head, trained end-to-end with binary cross-entropy on z_{ben}; serialization, architecture, optimization, and reproducibility details are given in Appendix C.

A correctness estimator predicts whether $a_0(q)$ is likely to be wrong. Benefit supervision instead labels whether executing the RAG branch improves that answer by more than ε, with sampled answers and answer-specific verbalized confidence serving as predictive inputs.

3.3 SELECTIVE INVOCATION

At inference time, BeneRAG invokes the RAG branch only when

$$r(q) = \mathbb{I}[b_\theta(q) > \tau], \tag{6}$$

where τ is selected on the validation split and fixed before test evaluation. If $r(q) = 0$, the system returns the already available candidate $a_0(q)$; because the sampled responses serve both as the observation and as the returned answer, skipping requires no additional answer-generation step beyond the already produced no-retrieval samples. If $r(q) = 1$, the deployed retrieval and context-conditioned generation are executed and $a_1(q)$ is returned. The resulting deployment path, shown in the lower panel of Figure 1, is $O(q) \to b_\theta(q) \to \{\text{SKIP}, \text{RETRIEVE}\}$, with the generator and retrieval system themselves unchanged.

The benefit distribution and appropriate operating point may change with the deployed RAG protocol. We therefore evaluate benefit ranking and threshold calibration separately in Section 4.4 and the appendix.

4 EVALUATION

The experiments test whether benefit supervision improves retrieval allocation over correctness supervision when observations and retrieval budgets are controlled. We first isolate the effects of the supervision target and pre-retrieval observations (Section 4.2), then evaluate the overall quality–invocation trade-off and semantic answer quality with an automated judge (Section 4.3), and finally examine how these findings extend to a shared retrieval corpus and unseen generators (Section 4.4).

4.1 EXPERIMENTAL PROTOCOL

We evaluate on SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), and AdversarialQA (Bartolo et al., 2020), spanning extractive, multi-hop, and adversarial QA. The pooled training and validation splits contain 146,572 and 36,644 questions. Canonical comparisons share 10,000 held-out questions, with 34.33% positive benefit labels. Dataset counts and branch scores appear in Appendix A.3. A single estimator is trained on the pooled data with Qwen2.5-7B-Instruct as the source generator, without dataset-specific thresholds. Reported performance values are five-seed means. The canonical protocol, shared-corpus evaluation (Section 4.4), and separate execution comparison (Appendix G.5) use their own branch outputs and labels and are reported separately.

The canonical protocol controls evidence availability through query-specific candidate pools. Overlapping source chunks are mixed with five BM25-mined distractors per source chunk. The top $k = 3$ chunks are truncated to 700 characters. Supporting evidence is present in the pool but need not survive retrieval and truncation (Appendix B).

The no-retrieval branch samples $K = 3$ answers, returns their normalized majority vote, and elicits one verbalized confidence score. The RAG branch generates a single answer greedily. Both use matched answer-style instructions, with the RAG prompt additionally providing context and an instruction to use it (Appendix A.1).

We compare with ALWAYS LLM, ALWAYS RAG, the full-system reference Self-RAG (Asai et al., 2024), and protocol-adapted SeaKR (Yao et al., 2025) and Rowen (Ding et al., 2025) (Appendix D). Controlled estimators share the encoder, prediction head, split, and optimization while varying input and supervision. We report token-level F1, Exact Match (EM), the context-branch invocation rate (RR), and two AUROCs. Benefit AUROC uses the same z_{ben} for all estimators regardless of their training target, while correctness AUROC uses the low-quality label in Section 4.2. The call-adjusted utility is

$$U_{\text{call}} = \text{F1} - \alpha\,\text{RR}, \quad \alpha = 0.1, \tag{7}$$

which measures quality relative to invocation frequency, not total inference cost. Policy evaluation selects τ to maximize validation U_{call} and freezes it before testing, with $\varepsilon = 0.05$ throughout. A rate-matched policy control instead selects a competing estimator's threshold to match BeneRAG's validation invocation rate, then freezes it. Test rates remain close but unequal. Exact-budget ranking

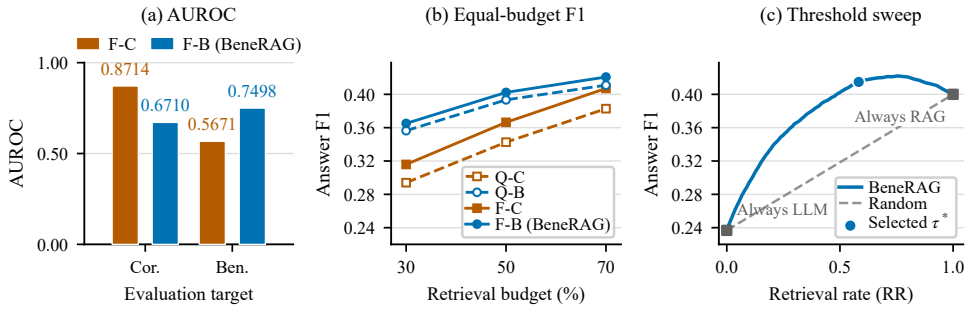


Figure 2: **Supervision target and retrieval allocation.** Q/F denote question-only/full observations and C/B denote correctness/benefit supervision. **F-B is BeneRAG.** (a) Correctness and benefit AUROC. (b) F1 under exact-budget ranking. (c) F1 versus RR across thresholds. The dot marks the validation-selected operating point. Values are five-seed means.

diagnostics have each estimator rank the same test questions and invoke retrieval on a fixed fraction without outcome labels. They equalize invocation counts but do not define a fixed-threshold deployment policy. These comparisons reuse cached branch answers, holding generation outcomes fixed across estimators. Automatic-comparison intervals use 10,000 paired bootstrap resamples over questions. LLM-judge and repeated-generation procedures appear in Appendices E.1 and F.2.

4.2 Supervision Target and Observable Inputs

We cross two input settings with two supervision targets. Q contains the question alone; F contains the question, sampled answers, and verbalized confidence. C predicts the low-quality label $z_{\mathrm{cor}}(q) = \mathbb{I}[y_0(q) < 0.5]$, whereas B predicts $z_{\mathrm{ben}}(q)$. Within each input setting, the encoder, prediction head, split, and optimization are identical. F-B is BeneRAG. Figure 2(a,b) compares target-specific ranking and allocation at exact retrieval budgets, while panel (c) shows BeneRAG's quality–invocation trade-off across thresholds; detailed policy metrics appear in Appendix D.1.

Figure 2(a) shows that the correctness estimator learns its assigned target: F-C reaches correctness AUROC 0.8714, but its benefit AUROC is only 0.5671. F-B reaches benefit AUROC 0.7498 from the same observations, and the question-only pair shows the same pattern (0.5264 vs. 0.7287). Training-free agreement and confidence triggers built from these observations rank benefit at 0.5767 and 0.5410 (Appendix D.2), so the evaluated correctness estimators and simple triggers remain in a narrow band well below the benefit-supervised estimators. C's weak benefit ranking therefore does not reflect a failure to learn correctness. An answer can be incorrect without being repairable by the available retrieval branch, while a partially correct answer can still benefit from context.

The advantage persists at a common budget. In the exact-budget diagnostic (Figure 2(b)), F-B exceeds F-C by 0.0492, 0.0359, and 0.0137 F1 at budgets of 0.30, 0.50, and 0.70; at the middle budget this is 0.4023 against 0.3664. The rate-matched policy control gives the same ordering with a threshold frozen before testing: correctness supervision reaches F1 0.3827 at RR 0.579, against 0.4149 at RR 0.583 for BeneRAG. Q-B also exceeds Q-C at all three budgets and exceeds F-C despite observing less, so richer observations do not compensate for correctness supervision in these comparisons. In the exact-budget comparisons, these differences reflect which questions receive retrieval rather than how many (Appendix D.1). At fixed budgets, identical invocation penalties make the F1 and U_{call} orderings equivalent. At RR 0.50, an oracle ranking by the realized branch difference reaches F1 0.4939, compared with 0.4023 for BeneRAG (Table 7).

Comparing F-B with Q-B isolates the observations' contribution under benefit supervision. The gain is smaller but consistent: F-B exceeds Q-B by 0.0087, 0.0089, and 0.0098 F1 at the three budgets. The input ablation attributes benefit-AUROC gains of 0.0173 to sampled answers and a further 0.0038 to confidence (Appendix F.1). This does not establish that decisions track individual answer realizations. Across repeated generations of the same question, exchanging decisions changes F1 by only -0.0003 (95% CI $[-0.0008, +0.0002]$, Table 10). The observations therefore help across

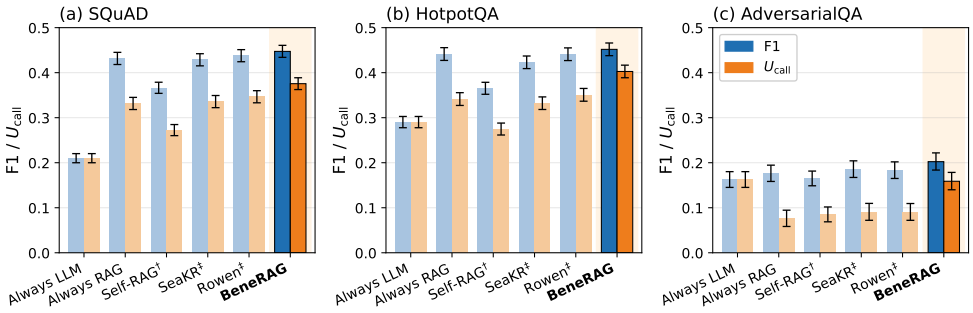


Figure 3: **Per-dataset answer quality (F1) and call-adjusted utility (U_{call}).** Self-RAG† uses its own generator; SeaKR‡/Rowen‡ are protocol adaptations. Bars: 95% bootstrap CIs (Appendix A.3). BeneRAG improves both metrics over ALWAYS RAG on all three datasets.

questions, with limited evidence for an additional effect of matching decisions to individual answer realizations.

The observations also carry execution cost. In the separate execution comparison, parallel $K = 3$ reaches F1 0.5192 against 0.5127 for Q-B, which generates an answer only after the invocation decision, but uses 0.77 rather than 0.36 GPU-s per query and has median latency 698 rather than 286 ms. Higher quality comes at higher cost, and fewer retrieval calls alone do not establish an efficiency gain (Figure 4).

4.3 QUALITY–INVOCATION TRADE-OFF AND SEMANTIC EVALUATION

We next evaluate the validation-selected operating point. BeneRAG improves pooled F1 from 0.4000 for ALWAYS RAG to 0.4149 while invoking the context branch on 58.3% of queries. The improvement holds on each dataset (Figure 3): 0.4473 vs. 0.4317 on SQuAD, 0.4518 vs. 0.4412 on HotpotQA, and 0.2023 vs. 0.1762 on AdversarialQA. Figure 2(c) traces the threshold trade-off and marks the validation-selected point. Benefit scores outperform random allocation at intermediate invocation rates (Appendix G.3).

We also assess semantic quality with GPT-4.1 on 1,000 stratified examples from the canonical Qwen2.5-7B-Instruct protocol. The judge grades each answer without seeing its branch identity (Appendix E.1). BeneRAG improves semantic accuracy from 0.468 for ALWAYS RAG to 0.509 (Table 1). Subset sizes and accuracies are computed within each seed before averaging. The policies return the same RAG answer on invoked examples, so their difference arises entirely on skipped examples. On that subset, the retained no-retrieval answers have semantic and lenient accuracies 9.3 and 18.6 percentage points higher than the RAG answers, respectively. This automated assessment supports the quality gain beyond token overlap.

The adaptive references invoke retrieval more often. Self-RAG, with its own retrieval-aware generator, obtains F1 0.3378 at RR 0.905; the protocol-adapted SeaKR and Rowen obtain 0.3923 at 0.923 and 0.4035 at 0.910, respectively. Their U_{call} values are 0.2472, 0.2999, and 0.3125, compared with 0.3566 for BeneRAG. BeneRAG $K = 3$ also uses less measured GPU time than the adapted SeaKR and Rowen systems (1.22 vs. 9.36 and 14.95 GPU-s per query). These costs reflect the evaluated protocol-adapted implementations and execution settings (Appendices D.3 and G.5). These system-level comparisons complement the supervision controls in Section 4.2.

The result also survives a document-disjoint split: benefit AUROC is 0.7389, and F1 is 0.4110 against 0.4061 for ALWAYS RAG. Official-normalization rescoring and a greedy-answer substitution control preserve the policy ordering (Appendices E.3 and A.4). These checks support the quality gain beyond the canonical split and token-overlap convention.

4.4 CROSS-MODEL TRANSFER AND PROTOCOL DEPENDENCE

We examine benefit ranking and policy quality under changes in retrieval protocol or generator. The first evaluation replaces query-specific candidate pools with a shared BM25 index of 20,879 source

Table 1: **Semantic evaluation overall and by retrieval decision.** GPT-4.1 evaluates 1,000 examples. Counts and accuracies are five-seed means. Semantic accuracy counts correct answers, while lenient accuracy also counts partially correct answers. ALWAYS RAG and BeneRAG return the same answer on invoked examples.

Subset	Mean n	Policy	Semantic	Lenient
All	1,000	ALWAYS LLM	0.325	0.514
		ALWAYS RAG	0.468	0.526
		BeneRAG	**0.509**	**0.608**
Retrieval skipped	438.6	ALWAYS RAG	0.349	0.425
		BeneRAG	**0.442**	**0.611**
Retrieval invoked	561.4	ALWAYS RAG / BeneRAG	0.561	0.605

passages from SQuAD v1.1, without inserting supporting documents into individual pools. Table 2 compares the frozen source estimator and threshold, threshold recalibration on validation labels from the new setting, and retraining with new labels (Appendix H.3).

Without retraining, shared-corpus benefit ranking remains useful, with AUROC 0.7267 compared with 0.7498 in the canonical protocol. At the source-selected threshold, however, F1 falls below ALWAYS RAG (0.5358 vs. 0.5532). Threshold recalibration recovers an F1 advantage at 0.5563, and retraining raises F1 further to 0.5617 while skipping 17.5% of retrieval calls. At their validation-selected operating points, retrained F-B achieves slightly higher F1 than F-C (0.5617 vs. 0.5607) with 10.9 percentage points fewer retrieval invocations (RR 0.825 vs. 0.934) and higher U_{call} (0.4792 vs. 0.4673). These operating points do not establish allocation performance at a common retrieval budget. A controlled distractor-free shift also shows the limitation of the source operating point, with F1 falling to 0.5383 against 0.6305 for ALWAYS RAG and protocol-specific retraining restoring an advantage at 0.6315 (Tables 18 and 19).

We next transfer the source estimator to five unseen generators with its weights and source-selected threshold fixed. Each target produces its own no-retrieval samples, verbalized confidence, and context-conditioned answer. Target labels are used only for evaluation in this zero-shot setting. Table 3 summarizes the results, with absolute branch scores and full confidence intervals in Appendix H.1.

Benefit ranking transfers without target-side training or threshold calibration, with AUROC ranging from 0.6912 to 0.7344. This includes GPT-4.1 and Claude Sonnet 4.5, which are queried through text generation without using internal model states. Target-generated observations also contribute beyond the question alone, with F-B obtaining higher F1 than Q-B at RR 0.50 on all five targets (Table 16). At the source-selected threshold, U_{call} exceeds each target's ALWAYS RAG by 0.0177 to 0.0621, while F1 improves on GPT-4.1 and Claude Sonnet 4.5 and decreases on the three open-weight targets. The separate evaluation with target-label threshold recalibration also yields lower F1 than ALWAYS RAG on those three targets. Thresholds maximize validation U_{call}, which can favor fewer invocations even when F1 decreases. Benefit ranking remains useful, while the F1 advantage and adaptation effects vary across settings.

Table 2: **Shared-corpus adaptation.** Validation-selected operating points use unmatched retrieval budgets.

Estimator	Ben. AUROC	F1	RR	U_{call}
ALWAYS LLM	–	0.2100	0.000	0.2100
ALWAYS RAG	–	0.5532	1.000	0.4532
F-B, frozen	0.7267	0.5358	0.717	0.4641
F-B, threshold recalibrated	0.7267	0.5563	0.842	0.4720
F-B, retrained	**0.7445**	**0.5617**	0.825	**0.4792**
Q-B, retrained	0.7272	0.5574	0.851	0.4723
F-C, retrained	0.5726	0.5607	0.934	0.4673

Table 3: **Zero-shot transfer** ($n = 10{,}000$ per target). Differences are relative to each target's ALWAYS RAG. Brackets give 95% paired-bootstrap CIs for ΔF1.

Target	Ben. AUROC	RR	ΔF1	ΔU_{call}
Qwen2.5-7B (source)	0.7498	0.583	$+0.0150$	$+0.0567$
GPT-4.1	0.7248	0.475	$+0.0096$ $[+0.0035, +0.0160]$	$+0.0621$
Claude Sonnet 4.5	0.6912	0.621	$+0.0110$ $[+0.0051, +0.0169]$	$+0.0489$
Llama-3.1-8B	0.7146	0.602	-0.0216 $[-0.0266, -0.0164]$	$+0.0182$
Phi-3.5-mini	0.7136	0.637	-0.0066 $[-0.0111, -0.0021]$	$+0.0297$
EXAONE-3.5-2.4B	0.7344	0.633	-0.0189 $[-0.0241, -0.0139]$	$+0.0177$

5 DISCUSSION

The controlled comparisons show that benefit supervision improves retrieval allocation over correctness supervision when observations, architecture, training, and retrieval budgets are held fixed (Figure 2). The correctness estimator learns its target, yet even a question-only benefit estimator outperforms the full-observation correctness estimator. Knowing that an answer is poor therefore does not establish that the available RAG branch offers a better alternative. Sampled answers and verbalized confidence provide a smaller additional gain, but their contribution within individual questions remains unresolved. Exchanging decisions among repeated generations yields an F1 difference whose confidence interval includes zero (Appendix F.2). The evidence supports the observations' value across questions without establishing that decisions track individual answer realizations. Our external comparisons include Self-RAG and protocol-adapted SeaKR and Rowen. They do not establish superiority over black-box selective-retrieval methods more broadly.

The shared-corpus and cross-generator results distinguish useful benefit ranking from a suitable operating point. Threshold recalibration and retraining restore an F1 advantage in the shared-corpus setting, whereas target-label recalibration still leaves F1 below ALWAYS RAG on three generators (Section 4.4). Because thresholds maximize validation U_{call}, higher utility can coexist with lower F1. Observation acquisition also affects practical value. Canonical measurements show higher execution cost than ALWAYS RAG, despite lower cost than the adapted SeaKR and Rowen implementations. In the separate execution comparison, parallel $K = 3$ gains 0.0065 F1 over Q-B at 2.14 times the GPU use (Appendix G.5). Neither invocation rate nor U_{call} accounts for this cost. Deployment therefore requires considering answer quality, invocation rate, latency, and compute use together.

Benefit supervision requires paired branch outputs and reference answers offline, and protocol changes may require new outputs and labels. Token-level F1 remains an imperfect supervision signal. Its benefit labels agree with GPT-4.1's semantic-benefit labels on 89.1% of the evaluated subset, although this comparison is also subject to judge error (Appendix E.1). High agreement and verbalized confidence do not eliminate missed benefits. Unanimous samples with high confidence account for approximately 46% of missed benefits (Figure 5). Learning from semantic labels and selectively acquiring additional observations are possible extensions. Errors shared by both branches instead require improvements to retrieval or generation rather than to the invocation policy.

6 CONCLUSION

BeneRAG learns from paired branch outcomes to predict whether retrieval will improve the current answer using pre-retrieval black-box observations. With observations, architecture, training, and retrieval budgets held fixed, benefit supervision yields higher F1 than correctness supervision at every tested budget. GPT-4.1 evaluation also favors the no-retrieval answers retained on skipped queries over their RAG alternatives. Shared-corpus adaptation restores the F1 advantage lost by the frozen policy, while cross-generator transfer preserves useful benefit ranking without consistently outperforming ALWAYS RAG in F1. Additional observations also incur execution cost. These results support benefit supervision for retrieval allocation, with its practical value depending on the deployed protocol, operating point, and observation cost.

Reproducibility Statement

Our primary artifact is the encoder-based benefit estimator reported in the main text. Prompts, decoding settings, answer aggregation, and normalization for both branches are specified in Appendix A.1; corpus construction, chunking, distractor injection, and retrieval parameters in Appendix B; and the estimator's input serialization, architecture, optimization, and threshold-selection rule in Appendix C. Supervision labels are fully determined by Equations (1)–(3) given the branch outputs, and reported operating points follow the validation-selection rule of Section 4.1. Performance is reported as means over five seeds, with per-run validation threshold selection as described in Appendix G.4. Statistical intervals use $10{,}000$ paired bootstrap resamples with fixed seeds, and the automated semantic-evaluation protocol is documented in Appendix E.1.

AI Disclosure

AI assistants were used for code assistance during implementation of the analysis pipeline, for language editing of the manuscript, and to give feedback on experimental design choices, comparison methods, and the interpretation of results. All generated code was reviewed, executed, and verified by the authors; every reported numerical result was produced by running that code, not generated or estimated by an AI system.

No AI system was used to generate synthetic data or to produce any reported numerical result directly. Experimental design, the choice of comparison methods, and the interpretation of results received AI-assisted feedback, but every decision, every execution, and the final scientific judgment are the authors' own. The authors take full responsibility for all content of this paper, including any text, code, or analysis produced with AI assistance.

References

Chen Amiraz, Florin Cuconasu, Simone Filice, and Zohar Karnin. The distracting effect: Understanding irrelevant passages in RAG. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18228–18258. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.892. URL https://aclanthology.org/2025.acl-long.892/.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Ingeol Baek, Hwan Chang, ByeongJeong Kim, Jimin Lee, and Hwanhee Lee. Probing-RAG: Self-probing to guide language models in selective document retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3287–3304. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.181. URL https://aclanthology.org/2025.findings-naacl.181/.

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678, 2020. URL https://aclanthology.org/2020.tacl-1.43/.

Lihu Chen, Gerard de Melo, Fabian M. Suchanek, and Gaël Varoquaux. Query-level uncertainty in large language models. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=11QZITAMUO.

Hanxing Ding, Liang Pang, Zihao Wei, Huawei Shen, and Xueqi Cheng. Rowen: Adaptive retrieval-augmented generation for hallucination mitigation in LLMs. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP 2025, pp. 12–21. Association for Computing Machinery, 2025. doi: 10.1145/3767695.3769500. URL https://doi.org/10.1145/3767695.3769500.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, June 2024. doi: 10.1038/s41586-024-07421-0. URL https://doi.org/10.1038/s41586-024-07421-0.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.389. URL https://aclanthology.org/2024.naacl-long.389/.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Qinhong Lin, Zhongliang Yang, Yuang Cai, Dingfu Yu, Xuan Xu, Yu Li, and Linna Zhou. Semantic contribution-aware adaptive retrieval for black-box models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 16979–16994. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.921. URL https://aclanthology.org/2025.findings-emnlp.921/.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.acl-long.546/.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Maria Marina, Nikolay Ivanov, Sergey Pletenev, Mikhail Salnikov, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Alexander Panchenko, and Viktor Moskvoretskii. LLM-independent adaptive RAG: Let the question speak for itself. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8697–8709. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.439. URL https://aclanthology.org/2025.emnlp-main.439/.

Viktor Moskvoretskii, Maria Marina, Mikhail Salnikov, Nikolay Ivanov, Sergey Pletenev, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Irina Nikishina, and Alexander Panchenko. Adaptive retrieval without self-knowledge? Bringing uncertainty back home. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6355–6384. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.319. URL https://aclanthology.org/2025.acl-long.319/.

Changle Qu, Sunhao Dai, Hengyi Cai, Yiyang Cheng, Jun Xu, Shuaiqiang Wang, and Dawei Yin. Uplift-RAG: Uplift-driven knowledge preference alignment for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 9632–9644, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.511. URL https://aclanthology.org/2025.findings-emnlp.511/.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392. Association for Computational Linguistics, 2016. URL https://aclanthology.org/D16-1264/.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023. doi: 10.1162/tacl_a_00605. URL https://aclanthology.org/2023.tacl-1.75/.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.702. URL https://aclanthology.org/2024.acl-long.702/.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, 2023. doi: 10.18653/v1/2023.emnlp-main.330. URL https://aclanthology.org/2023.emnlp-main.330/.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan O Arik. Astute RAG: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30553–30571. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1476. URL https://aclanthology.org/2025.acl-long.1476/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gjeQKFxFpZ.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. URL https://aclanthology.org/D18-1259/.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Weichuan Liu, Lei Hou, and Juanzi Li. SeaKR: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27022–27043. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1312. URL https://aclanthology.org/2025.acl-long.1312/.

A Generation and Evaluation Protocol

A.1 Generation and Answer Aggregation

Qwen2.5-7B-Instruct serves as the source generator in the canonical experiments. The same branch-generation protocol is used across SQuAD, HotpotQA, and AdversarialQA, and all prompts and decoding parameters are fixed before test evaluation.

No-retrieval branch. For each question q, the generator produces $K = 3$ no-retrieval responses using temperature 0.8, top-$p = 0.95$, and a maximum of 64 new tokens. Each response is normalized using Appendix A.2. The final candidate $a_0(q)$ is selected by majority vote over normalized responses. If all normalized responses are distinct, or if multiple candidates are tied, the earliest sampled response among the tied candidates is returned.

Verbalized self-assessment. After producing the no-retrieval samples, the same generator is shown the question and the first sampled answer and receives the instruction

```
Output only a number between 0 and 1 indicating how likely
the answer is correct.
```

The response is generated greedily with a maximum of 8 new tokens. The first parseable scalar is mapped to $[0, 1]$ and used as $c_{\text{conf}}(q)$; if no valid scalar can be parsed, the fallback value is 0.5. This quantity is obtained entirely from generated text rather than from token probabilities, logits, or another model-internal confidence signal. The same elicitation and parsing procedure is used for the source generator and all cross-model transfer targets.

Context branch. The deployed RAG branch generates one answer greedily with a maximum of 64 new tokens. Retrieved chunks are separated by blank lines and truncated to at most 700 characters in the canonical setting before being inserted into the prompt. During offline label construction, this branch is executed for every training example to measure the paired branch outcome. At deployment, it is executed only when BeneRAG invokes retrieval.

For all fixed-branch evaluations, the two branches use matched concise-answer instructions:

No retrieval

System: `Answer with the shortest span that answers the question. Do not explain.`
User:
```
Question: {question}
Answer:
```

RAG

System: `Use the context. Answer with the shortest span that answers the question. Do not explain.`
User:
```
Context:
{context}
Question: {question}
Answer:
```

Thus, the answer-style instruction is matched across branches; the RAG prompt differs only by the supplied context and the instruction to use it.

A.2 Answer Normalization and Scoring

Generated answers and references are normalized by lowercasing, removing punctuation, and collapsing repeated whitespace. Unlike the official SQuAD evaluation script, the canonical normalization does not remove the English articles *a*, *an*, and *the*. The same normalization is applied to every policy and branch. Exact Match (EM) is 1 when the normalized prediction matches any normalized reference answer and 0 otherwise. Token-level F1 is the maximum overlap score over the reference-answer set $G(q)$:

$$\text{F1}(a, G) = \max_{g \in G} \text{tokenF1}(\text{norm}(a), \text{norm}(g)) \,. \tag{8}$$

This F1 score is the canonical Score_S used to construct the branch outcomes and benefit labels in Section 3.1. Sensitivity to the official SQuAD normalization, including article removal, is reported in Appendix E.3.

For an invocation policy with per-example invocation indicator r_i, the call-adjusted utility is

$$u_i = \mathrm{F1}_i - \alpha r_i, \qquad \alpha = 0.1, \tag{9}$$

so that averaging u_i over the evaluation set yields the reported U_{call}. This quantity summarizes answer quality and context-branch usage; it is not a measure of total latency, token consumption, or monetary cost. Full token and generation-cost accounting is reported in Appendix G.5.

A.3 Data Splits and Statistical Testing

The pooled dataset contains $229{,}021$ questions. We first construct an $80{:}20$ train–test partition stratified by the canonical benefit label z_{ben}. The training portion is split using the same stratification, yielding $146{,}572$ training questions and $36{,}644$ validation questions. Canonical evaluations use a fixed set of $10{,}000$ questions sampled from the held-out test partition. These questions are shared across fixed policies, BeneRAG, adaptive references, and canonical ablations. The shared-corpus, execution-cost, document-disjoint, and repeated-generation analyses are separate evaluations. Target-label recalibration uses a held-out subset as described in Appendix H.2. Split construction and evaluation sampling use seed 42. A document-disjoint reconstruction appears in Appendix E.2.

Unless otherwise stated, automatic test-set comparisons use paired bootstrap resampling. For canonical comparisons, we form per-example paired differences on the shared $10{,}000$ questions and resample these pairs with replacement for $B = 10{,}000$ replicates. Analyses on other evaluation sets resample their respective questions. All generation repetitions of the same question remain grouped in Appendix F.2. We report percentile-based 95% confidence intervals. When reported, two-sided bootstrap p-values for automatic comparisons are computed from the paired bootstrap distribution. Bootstrap resampling uses seed 42.

Table 4 gives the composition of the canonical test set. Pooled scores weight individual questions equally, rather than averaging the three dataset means. The positive-benefit rate refers to z_{ben} with $\varepsilon = 0.05$.

Table 4: **Canonical test composition and branch performance.** Performance and positive-benefit rates are five-seed means.

Dataset	n	$P(z_{\mathrm{ben}} = 1)$	LLM F1	RAG F1	BeneRAG F1	RR
SQuAD	4,418	0.4244	0.2100	0.4317	0.4473	0.717
HotpotQA	4,185	0.3092	0.2902	0.4412	0.4518	0.491
AdversarialQA	1,397	0.1890	0.1622	0.1762	0.2023	0.435
Pooled	10,000	0.3433	0.2368	0.4000	0.4149	0.583

A.4 Greedy-Answer Substitution Control

The canonical branches intentionally use different generation procedures: the no-retrieval branch uses $K = 3$ stochastic samples followed by majority voting, whereas the RAG branch uses one greedy generation. Because context benefit is defined between these fixed end-to-end branches, this asymmetry is part of the canonical protocol S.

We assess sensitivity to the returned no-retrieval answer by replacing the canonical majority-vote candidate with an answer generated in a single greedy pass. With the RAG answer unchanged, this substitution changes 1.7% of the context-benefit labels, and the continuous benefit values have Pearson correlation $r = 0.9582$ with the canonical values. For policy evaluation, the estimator continues to use the canonical sampled observations, and its weights, threshold, and invocation decisions are kept fixed; only the answer returned after a skip is replaced. BeneRAG, Always RAG, and Always LLM then obtain F1 scores of 0.4137, 0.4000, and 0.2378, respectively. The policy ordering therefore persists under greedy-answer substitution. This control assesses the returned answer while retaining the canonical observation-generation procedure.

Table 5: **Sensitivity to retrieval and context construction** on the canonical 10,000-example test set. BeneRAG decisions are fixed across conditions. The final column reports HotpotQA ALWAYS RAG F1; ALWAYS LLM obtains 0.2902 F1 on HotpotQA.

Configuration	RAG F1	BeneRAG F1	U_{call}	HotpotQA RAG
Canonical (420/280, top-3, 700 chars)	0.4000	0.4149	0.3566	0.4412
Window 280/180	0.4107	0.4228	0.3645	0.4461
Window 560/380	0.4015	0.4168	0.3584	0.4363
Top-$k = 1$	0.3314	0.3639	0.3056	0.3884
Top-$k = 5$	0.4123	0.4240	0.3657	0.4419
Context 500 chars	0.3821	0.4018	0.3435	0.4179
Context 1,000 chars	0.4081	0.4209	0.3626	0.4512

B RETRIEVAL ENVIRONMENT

B.1 CANONICAL CANDIDATE POOL AND RETRIEVAL SETUP

The canonical evaluation uses a controlled, query-specific candidate pool. The shared-corpus evaluation is described separately in Appendix H.3. Each question's pool is constructed from its source documents, including the supporting passages provided for HotpotQA. Source documents use a 420-character window and stride 280, retaining at most 16 chunks per document. Short documents also use sentence-level segmentation when appropriate. This construction yields approximately 1–10 source chunks per query before distractor injection.

For every source chunk, we add five hard distractor chunks from other documents in the same dataset. Distractors are selected for high BM25 relevance to the query while excluding chunks from the query's source documents. Candidate-pool construction and ordering are fixed with seed 42. The pool is roughly six times the source-only pool. Supporting evidence is present in the pool but may be absent from the retrieved context.

BM25 (`BM25Okapi`) ranks the candidates. We select the top $k = 3$ chunks, or all chunks if fewer are available. The selected chunks are concatenated with blank-line separators and truncated to 700 characters. Relative to the source-only pool, distractor injection changes the retrieved top-3 context for 91.7% of test queries. Because the pools are small and query specific, BM25 mainly determines which source and distractor passages appear before truncation rather than performing open-corpus document discovery. The distractor-free condition is examined in Appendix H.5.

B.2 CHUNKING, TOP-k, AND CONTEXT-LENGTH SENSITIVITY

Table 5 compares chunk windows and strides, BM25 top-k, and context lengths while keeping the generator, prompts, scoring procedure, BeneRAG estimator, and invocation decisions fixed. Because BeneRAG uses only pre-retrieval observations, these changes affect only the context-conditioned branch.

Across all seven configurations, BeneRAG obtains U_{call} between 0.3056 and 0.3657 and exceeds the corresponding ALWAYS RAG utility. Its F1 advantage over ALWAYS RAG ranges from $+0.0117$ to $+0.0325$. The size of the advantage varies across the tested retrieval and context configurations. The canonical configuration was fixed during development; the sweep assesses sensitivity to the tested changes.

C BENERAG IMPLEMENTATION

C.1 INPUT SERIALIZATION, ARCHITECTURE, AND OPTIMIZATION

The observable state is serialized into a single text sequence with labeled fields in the following order:

`[QUESTION]` q
`[SAMPLE 1]` $a_1^{(0)}$

```
...
[SAMPLE K] a_K^(0)
[SELF-ASSESSMENT] c_conf
```

The retrieved context, RAG output, reference answer, and branch scores are not included. The sequence is truncated to at most 256 encoder tokens.

The canonical estimator initializes a DeBERTa-v3-base encoder and mean-pools its final hidden layer. A two-layer prediction head with hidden dimension 256, GELU activation, and dropout 0.1 maps the pooled representation to the benefit probability. The encoder and head are trained jointly with binary cross-entropy on z_{ben}. Optimization uses AdamW with weight decay 0.01, a one-cycle learning-rate schedule peaking at 2×10^{-5}, a 10% warm-up period, batch size 32, and 2 epochs. Training uses single-precision floating point.

C.2 Invocation-Threshold Selection

The label margin ε and deployment threshold τ play distinct roles. The margin ε defines the binary offline target before training, whereas τ converts the predicted benefit probability into an invocation decision. The canonical margin is fixed at $\varepsilon = 0.05$. The threshold is selected on the validation split to maximize U_{call} and is then frozen for all test evaluations. Threshold selection is performed separately for each training run; $\tau^* = 0.21$ is the reference operating point used for the illustrative threshold sweep. No test-set information is used in checkpoint or threshold selection.

C.3 Reproducibility and Deterministic Training

Training uses fixed seeds, deterministic GPU kernels, and the shared data split and preprocessing pipeline described above. Reported performance values are averaged over five seeds (Appendix G.4).

D Baselines and Controlled Comparisons

D.1 Supervision and Retrieval-Budget Controls

We train question-only (Q) and full-observation (F) estimators with correctness (C) or benefit (B) supervision. All four use the encoder, prediction head, split, and optimization in Appendix C.1; Q removes the sampled-answer and confidence fields. The correctness target is

$$z_{\mathrm{cor}}(q) = \mathbb{I}[y_0(q) < \delta], \qquad \delta = 0.5. \tag{10}$$

The cutoff is fixed before training. Correctness AUROC is evaluated against this low-quality label, so larger scores represent a greater probability of a low-quality no-retrieval answer. Benefit AUROC uses the same z_{ben} for all estimators. Table 6 includes the full policy metrics and training-free references.

Table 6: **Detailed supervision and trigger comparisons.** Unless marked RR-matched, operating points are selected by validation U_{call}.

Estimator	Cor. AUROC	Ben. AUROC	F1	EM	RR	U_{call}
Q-C	0.8008	0.5264	0.4001	0.3048	0.808	0.3194
Q-B	0.5904	0.7287	0.4034	0.3075	0.578	0.3455
F-C	0.8714	0.5671	0.4113	0.3087	0.742	0.3371
F-B	0.6710	0.7498	0.4149	0.3164	0.583	0.3566
F-C, RR-matched	0.8714	0.5671	0.3827	0.2790	0.579	0.3248
Agreement trigger	0.7279	0.5767	0.3781	0.2858	0.571	0.3210
Confidence trigger	0.7091	0.5410	0.3970	0.3002	0.786	0.3184

The earlier RR-matched correctness control selects its threshold to match BeneRAG's invocation rate on validation data and then freezes it. Test rates are close but not equal (0.579 vs. 0.583); the corresponding F1 values are 0.3827 and 0.4149. We retain this policy comparison but use exact-budget selection to remove the remaining difference in invocation rates.

For an exact-budget diagnostic at rate r, the estimator ranks the same test questions and invokes the RAG branch on the top rN questions. The measured budgets are $r \in \{0.30, 0.50, 0.70\}$ with $N = 10{,}000$, so the invocation counts are integers. Ties are resolved without outcome labels. Every method uses the same cached branch answers; only the selected subset differs. Table 7 gives the numerical results underlying Figure 2(b), together with the confidence trigger and outcome oracle. This batch ranking evaluation does not define a fixed-threshold deployment policy.

Table 7: **F1 at exact retrieval budgets.** The oracle ranks by the observed continuous branch difference Δ_S and uses exactly the stated budget. It has access to outcomes unavailable to a deployed router.

Estimator	RR 0.30	RR 0.50	RR 0.70
Q-C	0.2940	0.3426	0.3828
Q-B	0.3564	0.3934	0.4110
F-C	0.3159	0.3664	0.4071
F-B	0.3651	0.4023	0.4208
Confidence trigger	0.3050	0.3417	0.3802
Outcome oracle	0.4861	0.4939	0.4939

At a common RR, differences in U_{call} equal differences in F1. F-B exceeds F-C at all three measured budgets; Q-B also exceeds F-C despite using fewer observations. The oracle values quantify remaining allocation headroom rather than an AUROC ceiling for the observable-input estimator.

D.2 Training-Free Agreement and Confidence Triggers

The agreement trigger uses normalized agreement among the K no-retrieval responses; the confidence trigger uses the verbalized scalar alone. Retrieval is favored for lower agreement or confidence. Neither trains an encoder or observes retrieved passages. Both thresholds are selected on validation data under the same U_{call} criterion. Their benefit AUROCs are 0.5767 and 0.5410, respectively, compared with 0.7498 for F-B. Their policy outcomes are listed in Table 6.

D.3 Fixed Policies and Adaptive References

Always LLM always returns the aggregated no-retrieval answer and has RR 0. Always RAG invokes the same deployed RAG branch used by BeneRAG for every query and has RR 1. These policies define the endpoints of the invocation problem.

Self-RAG is evaluated as a specialized full-system reference rather than as a controlled estimator attached to the fixed source generator. Its generator is trained for retrieval-aware generation and reflection, so its absolute F1 should not be attributed solely to a different invocation rule. SeaKR and Rowen are evaluated as protocol adaptations: their invocation principles are retained, while final answer quality is measured using the common fixed branches and scoring protocol. We therefore mark them as protocol adaptations rather than exact reproductions of their original end-to-end systems.

E Core Validity Checks

E.1 LLM-Judge Semantic Re-evaluation

We assess semantic answer quality with GPT-4.1 at temperature 0 on a stratified sample of $1{,}000$ examples that preserves the dataset proportions of the pooled evaluation set. The evaluated answers come from the canonical Qwen2.5-7B-Instruct protocol. The judge receives the question, reference answer, and the no-retrieval and RAG-branch answers in randomized A/B order without branch identity. Each answer is graded independently as *correct*, *partially correct*, or *incorrect*. Semantic accuracy counts only answers graded correct, while lenient accuracy also counts partially correct answers. A semantic-benefit label is positive when the RAG answer receives a higher grade than the no-retrieval answer. BeneRAG and its invocation threshold are kept fixed.

Token-F1 and LLM-judge semantic-benefit labels agree on 0.891 of the examples. Under the judge, semantic accuracy is 0.325 for ALWAYS LLM, 0.468 for ALWAYS RAG, and 0.509 for BeneRAG. Under the lenient criterion, the corresponding values are 0.514, 0.526, and 0.608 (Table 1).

For each seed, we partition the evaluation sample by BeneRAG's invocation decisions and compute the count and accuracy within each subset. Table 1 reports the overall and subset results as means across five seeds. The mean subset sizes are 438.6 skipped and 561.4 invoked examples. Both policies return the same RAG answer on invoked examples and therefore receive the same judge grade. Their overall accuracy difference arises from the skipped subset, where BeneRAG retains the no-retrieval answer. On this subset, mean semantic accuracy is higher by 0.093, and mean lenient accuracy is higher by 0.186. These gains account for the overall improvements of 0.041 and 0.082, respectively, with the decomposition computed within each seed before averaging. These point estimates support retaining the no-retrieval answers on the selected subset under the automated judge.

E.2 DOCUMENT-DISJOINT SPLIT

The canonical split is defined at the question level, so documents can otherwise appear across partitions, particularly because AdversarialQA includes questions constructed from SQuAD passages. We therefore create a global $8{:}1{:}1$ document-level split with zero document overlap and repeat training and evaluation. The resulting estimator obtains benefit AUROC 0.7389, close to the canonical 0.7498. On the document-disjoint evaluation partition ($19{,}358$ examples), BeneRAG obtains F1 0.4110 versus 0.4061 for ALWAYS RAG, with a U_{call} advantage of $+0.0489$. The main ordering therefore does not depend on document overlap across the canonical partitions.

E.3 NORMALIZATION SENSITIVITY

We rescore all cached no-retrieval and context-conditioned answers using the official SQuAD normalization, which additionally removes English articles relative to our canonical normalization. Because the context-benefit target is derived from token-level F1, changing the normalization convention can affect not only the reported answer scores but also the resulting benefit labels. For this sensitivity analysis, the trained estimator and its invocation decisions are kept fixed, and only the cached branch outputs are rescored under the alternative convention. Table 8 summarizes the resulting benefit-label changes and pooled policy scores. Official SQuAD normalization flips 1.23% (123 of $10{,}000$) of the benefit labels, while the pooled F1 advantage of BeneRAG over ALWAYS RAG remains essentially unchanged, from $+0.0150$ to $+0.0155$. The same ordering is preserved on each dataset: 0.4512 vs. 0.4347 on SQuAD, 0.4517 vs. 0.4412 on HotpotQA, and 0.2010 vs. 0.1729 on AdversarialQA. Thus, the main comparison is robust to the choice between our canonical normalization and the official SQuAD convention.

Table 8: **Sensitivity to answer normalization** on the pooled $10{,}000$-example test set. Label flip is the fraction of z_{ben} labels that change relative to the canonical normalization.

Normalization	Label flip	BeneRAG F1	ALWAYS RAG F1	ΔF1
Canonical	–	0.4149	0.4000	$+0.0150$
Official SQuAD	1.23%	0.4164	0.4009	$+0.0155$

F MODEL AND SIGNAL ANALYSES

F.1 OBSERVABLE-INPUT ABLATION

We remove input fields while retaining benefit supervision and the common training procedure. Table 9 reports each estimator at its own validation-selected threshold. Question-only input already provides substantial benefit-ranking information. Adding the sampled answers raises benefit AUROC from 0.7287 to 0.7460; confidence raises it further to 0.7498. The exact-budget comparison in Figure 2 confirms that the full input improves allocation over question-only input independently of threshold selection.

Table 9: **Observable-input ablation with benefit supervision.**

Input	Ben. AUROC	F1	EM	RR	U_{call}
Question only	0.7287	0.4034	0.3075	0.578	0.3455
Sampled answers only	0.6318	0.3997	0.3089	0.739	0.3258
Question + samples	0.7460	0.4074	0.3087	0.547	0.3527
Question + samples + confidence	0.7498	0.4149	0.3164	0.583	0.3566

F.2 Within-Question Decision Exchange

To test whether the decision benefits from its association with a particular answer realization, we repeat no-retrieval generation five times for each of 1,000 questions. Each repetition independently generates the $K = 3$ samples and confidence, forms the no-retrieval answer, and applies the fixed estimator and threshold. The greedy RAG answer is held fixed within a question. These five generation repetitions are distinct from the five seeds used to summarize experimental performance.

We then randomly exchange invocation decisions among the five repetitions of the same question, keeping the answers being evaluated unchanged. This preserves each question's invocation count while breaking the association between its decision and answer realization. Results are averaged over random exchanges. The paired comparison groups all repetitions of a question together for bootstrap resampling. Table 10 compares the exchanged decisions with the original F-B decisions and a Q-B policy at the same overall RR.

Table 10: **Within-question exchange results.** Differences are relative to the original F-B decisions. *Varying* restricts evaluation to questions whose decisions change across repetitions. The Q-B comparison uses the same overall RR.

Decision rule	F1	RR	Difference [95% CI]	Varying F1
F-B, original	0.4023	0.568	Reference	0.2943
F-B, exchanged	0.4020	0.568	-0.0003 $[-0.0008, +0.0002]$	0.2898
Q-B	0.3890	0.568	-0.0133 $[-0.0237, -0.0035]$	–

Across the five repetitions, representative answers differ on 55.7% of questions, benefit labels on 4.0%, and invocation decisions on 6.5%. Exchanging decisions produces a small overall F1 decrease whose interval includes zero. The varying-decision subset has a larger point difference (0.2943 vs. 0.2898), but it covers few questions and is a conditional diagnostic. F-B exceeds Q-B in this evaluation, yet the exchange result does not establish a reliable advantage from matching decisions to individual answer realizations. Nor does it show that the estimator uses question information alone. The benefit-label change rate reflects changes in both the answer and observation; it is not an estimate of irreducible label noise or an AUROC ceiling.

G Sensitivity and Cost Analysis

G.1 Number of No-Retrieval Samples

Table 11 compares quality, benefit ranking, and token use for $K \in \{1, 3, 5\}$. Changing K changes both the aggregated no-retrieval candidate and the paired benefit labels, so the estimator is relabeled and retrained for every setting. For $K = 1$, the first response is used directly; for $K = 5$, two additional responses are generated using the canonical sampling protocol.

The canonical $K = 3$ obtains the highest benefit AUROC and U_{call}. Increasing to $K = 5$ adds tokens without improving ranking or utility, whereas $K = 1$ preserves most of the performance with lower observation cost.

Table 11: **Effect of the number of no-retrieval samples** on the canonical 10,000-example test set. Tokens/query includes the preliminary samples, self-assessment, and context-branch calls at the observed invocation rate.

K	Ben. AUROC	F1	RR	U_{call}	Tokens/query
1	0.7448	0.4080	0.558	0.3522	232
3 (canonical)	**0.7498**	**0.4149**	0.583	**0.3566**	342
5	0.7477	0.4115	0.562	0.3553	443

G.2 Context-Benefit Margin Sensitivity

Table 12 reports the sensitivity to $\varepsilon \in \{0.02, 0.05, 0.10, 0.15\}$ with the data split fixed. Each non-canonical setting recomputes the benefit labels, retrains the estimator with the canonical procedure, and reselects τ on the validation split; selected thresholds range from 0.19 to 0.22.

Table 12: **Sensitivity to the context-benefit margin ε.**

ε	Ben. AUROC	F1	RR	U_{call}
0.02	0.7502	0.4155	0.587	0.3567
0.05	0.7498	0.4149	0.583	0.3566
0.10	0.7492	0.4111	0.549	0.3562
0.15	0.7471	0.4146	0.578	0.3569

Across the sweep, benefit AUROC remains between 0.7471 and 0.7502 and utility between 0.3562 and 0.3569. The 192 evaluation examples whose labels can change across the extreme margins constitute 1.9% of the test set, and the resulting utility range is only 0.0007.

G.3 Decision-Threshold Sensitivity

Figure 2(c) shows the canonical F1–RR threshold curve and the validation-selected operating point. Sweeping τ over the fixed canonical estimator yields maximum F1 0.4221 near RR 0.76. The validation-selected threshold $\tau^* = 0.21$ lies on a broad, nearly flat utility region. When the deployment preference changes, the threshold can be adjusted without retraining the estimator: for $\alpha = 0.05$, the descriptive test optimum is $\tau = 0.18$ with $U_{\text{call}} = 0.3866$, close to 0.3858 at the validation-selected threshold; for $\alpha = 0.1$, the selected operating point obtains 0.3566; and for $\alpha = 0.2$, the preferred threshold shifts to 0.40 with utility 0.3054.

G.4 Five-Seed Reporting

Reported performance values are means over five seeds. Learned estimators use the fixed split and the same configuration within each comparison, with the invocation threshold selected independently on validation data for each run. The reported question-bootstrap intervals describe paired variation over evaluation questions; they should not be interpreted as standard deviations across training seeds. Five repeated generations per question in Appendix F.2 serve a different purpose: they test the association between observations and decisions within the same question.

G.5 Generation Cost and Measured Execution Cost

U_{call} penalizes invocation frequency but omits preliminary sampling and self-assessment. We therefore distinguish token-based generation accounting from measured GPU use and wall-clock latency. The output-token-equivalent cost is 0.25 times input tokens plus output tokens, using the source tokenizer. This conversion is illustrative and does not represent total latency or monetary cost. Table 13 retains the token counts; Table 14 adds the measured execution results.

Measurements use a single NVIDIA RTX 4090 GPU (PyTorch 2.11, Transformers 5.8.0, float16 weights), single-stream execution (batch size 1, concurrency 1), and per-question timers (`time.perf_counter`, with `torch.cuda.synchronize` immediately before each timer

Table 13: **Token accounting under the canonical protocol.** Counts are per query; generation cost is in output-token equivalents. Dashes indicate quantities not reported for that system.

Method	Input tokens	Output tokens	Generation cost
ALWAYS LLM	145	13	49
ALWAYS RAG	195	5	54
Self-RAG	–	–	–
SeaKR (adapted)	474	5	124
Rowen (adapted)	1493	572	945
Q-B	–	–	–
BeneRAG ($K = 1$)	217	15	69
BeneRAG ($K = 3$)	319	24	103
BeneRAG ($K = 5$)	411	32	135

Table 14: **Answer quality and measured execution cost.** GPU-s and equivalent generation cost are per-query means. p50 and p95 are latency percentiles in ms. In the upper block, F1, RR, and equivalent cost use the canonical $10{,}000$-question evaluation, while GPU-s and latency use a 500-question shared-corpus sample. In the lower block, all metrics use that same 500-question sample. Only the lower block measures quality and execution cost together under the same conditions.

Method / execution	F1	RR	Equiv. cost	GPU-s	p50	p95
Canonical evaluation						
ALWAYS LLM	0.2368	0.000	49	0.76	694	1514
ALWAYS RAG	0.4000	1.000	54	0.35	266	946
SeaKR (adapted)	0.3923	0.923	124	9.36	9863	12646
Rowen (adapted)	0.4035	0.910	945	14.95	14623	19256
Q-B	0.4034	0.578	–	–	–	–
BeneRAG ($K = 1$)	0.4080	0.558	69	0.66	586	1321
BeneRAG ($K = 3$)	0.4149	0.583	103	1.22	1112	2231
BeneRAG ($K = 5$)	0.4115	0.562	135	1.76	1576	3173
Separate execution comparison						
ALWAYS LLM, parallel	0.1994	0.000	49.3	0.30	248	696
Q-B, route then generate	0.5127	0.710	113.6	0.36	286	890
BeneRAG ($K = 3$), sequential	0.5113	0.704	166.0	1.22	1112	2231
BeneRAG ($K = 3$), parallel	0.5192	0.712	167.9	0.77	698	1501

starts and stops); GPU-s is this wall-clock latency in seconds, not integrated device utilization or a separate device-time counter. Each run excludes an initial set of warm-up questions from the reported statistics (20 for the canonical BeneRAG/Always-LLM/Always-RAG/SeaKR/Rowen measurements, 10 for the Q-B and K=1/K=3/K=5 comparisons below) and reuses no cache across questions (no KV-cache or retrieval-result reuse). The reported GPU-s and latency for a given system sum every stage its policy performs on that question: retrieval when invoked, the no-retrieval sample-and-confidence generation, the estimator's forward pass, and the context-branch generation when invoked; a skipped question still includes the no-retrieval generation and the estimator's forward pass, since the estimator must run to produce the skip decision itself, and excludes only retrieval and the context-branch generation.

The two blocks of Table 14 are not measured on the same evaluation set and must not be read as before/after variants of one comparison. The *canonical evaluation* block reports F1 and RR from the pooled $10{,}000$-question, query-specific-candidate-pool protocol used throughout the paper; the GPU-s/p50/p95 entries in this block come from real single-stream execution, measured on the 500-question shared-corpus sample of Appendix H.3 and reported here only to give a comparable cost figure alongside the canonical quality numbers, not as a matched measurement of that exact $10{,}000$-question run. The *separate execution comparison* block instead measures F1, RR, and cost together on that same 500-question shared-corpus sample; both quantities always come from the same run there. The difference in ALWAYS LLM's F1 between blocks (0.2368 canonical vs. 0.1994 separate) is not a paired estimate of an execution-order effect. The lower block uses a different evaluation sample and explicitly reports parallel execution. These measurements do not isolate the contributions of

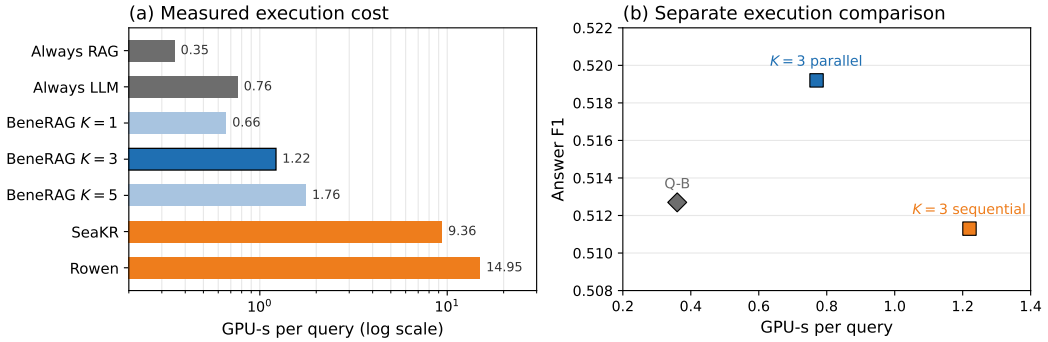


Figure 4: **Measured execution cost.** Left: GPU-s per query for the canonical-protocol methods, log scale, cost only; the corresponding F1 values in Table 14's canonical block are measured on a different evaluation set and are not paired with these GPU-s values as a single operating point. Right: quality and cost measured together on the same 500-question shared-corpus execution, for Q-B and the sequential/parallel $K = 3$ separate comparison. Full results, including latency and ALWAYS LLM, appear in Table 14.

sample selection and execution configuration to the F1 difference. We do not have evidence that switching from sequential to parallel execution leaves answer content unchanged in general: the paired $K = 3$ sequential and parallel rows in Table 14 have different F1 and RR, so their latency and token-accounting differences should not be read as isolating execution order alone. Q-B's canonical row is left blank because Q-B was not executed as a real single-stream system under the canonical protocol; its canonical F1/RR come from the token-based evaluation described above, and the 0.36 GPU-s reported for Q-B belongs only to the separate execution comparison and should not be copied into the canonical row.

ALWAYS LLM executes the $K = 3$ sample-and-aggregate no-retrieval branch to completion, including three sampled generations and their aggregation, whereas ALWAYS RAG performs one greedy generation. This difference in generation workload helps explain the higher measured GPU-s shown alongside the canonical results (0.76 vs. 0.35). The lower block has no ALWAYS RAG measurement and therefore does not support the same comparison.

Figure 4(a) compares measured GPU use alone, on a logarithmic scale; it carries no F1 axis. BeneRAG $K = 3$ uses 1.22 GPU-s per query, well below the adapted SeaKR (9.36) and Rowen (14.95) systems, and more than ALWAYS RAG (0.35) and ALWAYS LLM (0.76); median and p95 latency in Table 14 follow the same ordering. The $K = 1$ variant reduces GPU use to 0.66, consistent with the sample-count comparison in Table 11. These GPU-s values are measured on the 500-question shared-corpus execution of Appendix H.3; the canonical-protocol F1 column of Table 14 instead comes from the separate pooled 10,000-question evaluation used throughout the paper. The two columns describe the same methods under different evaluation conditions and are not a matched quality–cost operating point.

Figure 4(b) shows the separate execution comparison. Q-B decides before generation and generates a no-retrieval answer only when needed after a skip. Parallel $K = 3$ has F1 0.5192, 0.0065 above Q-B, but uses 2.14 times as much GPU time and has 2.44 times the median latency. Compared with sequential $K = 3$, the parallel execution uses 36.9% less GPU time and reduces median and p95 latency by 37.2% and 32.7%. Its token-equivalent generation cost does not decrease. Because F1 and RR also differ between the two executions, their quality difference cannot be attributed to parallelization alone.

H CROSS-MODEL AND CROSS-PROTOCOL GENERALIZATION

H.1 DETAILED CROSS-MODEL TRANSFER RESULTS

Table 15 complements the main-text summary in Table 3 with absolute branch F1 scores and confidence intervals for both differences. In the zero-shot evaluation, estimator weights and the

Table 15: **Detailed zero-shot transfer results** ($n = 10{,}000$ per target). Differences are relative to each target's own ALWAYS RAG; brackets are paired-bootstrap 95% confidence intervals.

Target	LLM-only F1	Always RAG F1	BeneRAG F1	RR	Ben. AUROC	ΔF1	ΔU_{call}
Qwen2.5-7B (source)	0.2368	0.4000	0.4149	0.583	0.7498	+0.0150	+0.0567
GPT-4.1	0.4260	0.5126	0.5223	0.475	0.7248	+0.0096 [+0.0035, +0.0160]	+0.0621 [+0.0558, +0.0686]
Claude Sonnet 4.5	0.2795	0.4184	0.4294	0.621	0.6912	+0.0110 [+0.0051, +0.0169]	+0.0489 [+0.0430, +0.0549]
Llama-3.1-8B	0.2361	0.4398	0.4182	0.602	0.7146	-0.0216 [−0.0266, −0.0164]	+0.0182 [+0.0132, +0.0234]
Phi-3.5-mini	0.2100	0.3928	0.3863	0.637	0.7136	-0.0066 [−0.0111, −0.0021]	+0.0297 [+0.0252, +0.0343]
EXAONE-3.5-2.4B	0.1779	0.4074	0.3884	0.633	0.7344	-0.0189 [−0.0241, −0.0139]	+0.0177 [+0.0126, +0.0229]

source-selected threshold remain fixed for every target. Each target generator produces its own no-retrieval samples, self-assessment, and context-conditioned answer. BeneRAG receives only the pre-retrieval observations when making a decision; target branch outputs and labels are used only for evaluation. No target-side benefit labels are used for adaptation or threshold selection. The source sampling parameters are retained, except that Claude Sonnet 4.5 uses its default top-p because its API does not accept temperature and top-p simultaneously.

Benefit AUROC remains between 0.6912 and 0.7344 on all five unseen targets, and ΔU_{call} is positive on every target. The source threshold does not preserve an F1 advantage over ALWAYS RAG on every model, illustrating the distinction between transferable ranking information and a protocol-specific operating point.

H.2 TARGET OBSERVATIONS AND THRESHOLD RECALIBRATION

Table 16 compares source-trained Q-B and F-B at exactly RR 0.50 for each target and reports a separate F-B threshold recalibration using 2,000 target-labeled examples. The recalibrated policy is evaluated on the remaining 8,000 examples; estimator weights are unchanged. This use of target labels is distinct from the zero-shot evaluation in Table 15. To make the effect of recalibration itself directly readable, we also re-aggregate the frozen source threshold on that same $8{,}000$-example evaluation split (mean over 50 random calibration/evaluation partitions, matching the recalibration procedure); its ΔF1 and ΔU_{call} columns are therefore a like-for-like before/after comparison with the recalibrated columns that follow, both against ALWAYS RAG on the same examples.

Table 16: **Target-observation controls and recalibration.** The first two columns report exact-budget ranking diagnostics. The remaining columns report the frozen source threshold and threshold recalibration, both aggregated on the same $2{,}000$ calibration / $8{,}000$ evaluation split per target (mean over 50 random splits). Differences use ALWAYS RAG on the same 8,000 evaluation examples.

	F1 at RR 0.50		Frozen (same split)		Recalibrated F-B			
Target	Q-B	F-B	ΔF1	ΔU_{call}	F1	RR	ΔF1	ΔU_{call}
GPT-4.1	0.5160	0.5242	$+0.0099$	$+0.0624$	0.5129	0.378	$+0.0003$	$+0.0625$
Claude Sonnet 4.5	0.4036	0.4113	$+0.0112$	$+0.0490$	0.4354	0.680	$+0.0171$	$+0.0491$
Llama-3.1-8B	0.3965	0.4053	-0.0213	$+0.0186$	0.4215	0.649	-0.0183	$+0.0168$
Phi-3.5-mini	0.3668	0.3676	-0.0065	$+0.0298$	0.3837	0.623	-0.0087	$+0.0289$
EXAONE-3.5-2.4B	0.3614	0.3679	-0.0184	$+0.0182$	0.3979	0.717	-0.0093	$+0.0190$

At the matched budget, the F-B point estimate exceeds Q-B on all five targets, with F1 differences from 0.0008 to 0.0088. This supports a contribution from target-generated observations beyond the question-only input. Recalibrated U_{call} exceeds ALWAYS RAG on all targets, but F1 remains lower on the three open-weight targets. Recalibration optimizes the stated quality–invocation objective and does not guarantee an F1 advantage. Since Table 15 uses all 10,000 questions, its absolute scores cannot be used as paired before–after estimates of recalibration on the 8,000-example subset; the frozen-threshold columns of Table 16, aggregated on that same subset, provide the matched before–after comparison instead. Relative to the frozen threshold, recalibration changes ΔU_{call} by at most 0.0018 (on Llama-3.1-8B; 0.0001–0.0009 on the other four targets), so most of its effect on U_{call} here is already present at the frozen threshold and recalibration mainly shifts the operating point (RR). The corresponding F1 shifts are target-dependent, increasing on Claude Sonnet 4.5, Llama-3.1-8B, and EXAONE-3.5-2.4B and decreasing on GPT-4.1 and Phi-3.5-mini relative to the frozen threshold.

H.3 SHARED-CORPUS RETRIEVAL

The shared-corpus evaluation replaces query-specific candidate pools with one index searched by every question, without forcing the query's supporting documents into a candidate pool. Within this evaluation, methods share the retrieval protocol and branch answers. We evaluate the frozen source model and threshold, threshold-only recalibration, and protocol-specific retraining; recalibration and retraining use labels from the new protocol. The retrained Q-B and F-C controls use the same input definitions and training procedure as their canonical counterparts.

We use SQuAD v1.1. Of its questions, $82{,}759$ carry a canonical split assignment from Appendix A.3; we reuse that per-question train/validation/test label directly rather than drawing a new split, giving $62{,}849$ training, $15{,}492$ validation, and $4{,}418$ test questions. Because each question keeps its original canonical label, no question used to train or select the threshold of the frozen source estimator appears in this shared-corpus test set. The shared index is built from the $20{,}879$ unique source passages of these same questions, retrieved by BM25 (verified against a reference implementation to a maximum absolute score difference of 3.4×10^{-11} on 30 sampled questions); the retrieval unit is the whole passage, and the top $k = 3$ passages are concatenated as context. The context budget is 768 tokens under the source tokenizer, chosen as the smallest of seven candidate budgets (256–$2{,}048$ tokens) whose validation-set answer-containment reaches at least 99% of the untruncated value; the realized context averages approximately 485–490 tokens per query, so truncation rarely binds. Passage-level Recall@k on the test split is 0.514, 0.636, 0.682, and 0.732 for $k = 1, 3, 5, 10$; post-truncation answer containment in the retrieved top-3 context is 0.652, matching the untruncated value to three decimal places.

Table 2 reports the measured operating points. Freezing the source model gives benefit AUROC 0.7267 but loses 0.0174 F1 against this setting's ALWAYS RAG. Threshold recalibration retains the same ranking and raises F1 to 0.5563; retraining raises benefit AUROC to 0.7445 and F1 to 0.5617 at RR 0.825. Relative to ALWAYS RAG, the retrained model gains 0.0085 F1 and 0.0260 in U_{call}. The retrained correctness control has lower benefit AUROC (0.5726) and invokes more often (RR 0.934), but these validation-selected operating points are not an exact-budget comparison. We therefore use them to assess adaptation and utility in this corpus, while the controlled allocation claim rests on Figure 2.

H.4 ROBUSTNESS TO CONTEXT CONDITIONS

We keep the trained estimator and source threshold fixed and alter only the context branch. Table 17 evaluates a longer $2{,}000$-character context budget and a partial-evidence HotpotQA condition in which one of the two supporting paragraphs is removed from the question's own source document before its query-specific candidate chunks are built. The distractor pool excludes a query's own document the same way as Appendix B.1, by matching each candidate chunk's source-document text against the query's own document text; for this condition, however, the query's own document text is the edited (paragraph-removed) copy, while the pool's per-chunk document identity is built from the original, unmodified corpus. Because the edited copy no longer matches any entry in that original-text lookup, the exclusion never fires for this condition, and the removed paragraph can be drawn back in as a same-topic distractor from the query's own (unedited) document via the shared pool; this is a lookup-key mismatch in the exclusion step, not a separate filter or content from another document, and it is what the small residual both-included rate below reflects. The distractor-free condition is reported separately in Appendix H.5.

Table 17: **Robustness to context conditions.** BeneRAG decisions and the source threshold remain fixed; only the context branch is reconstructed.

Condition	Scope	RAG F1	BeneRAG F1	RR	U_{call}
Canonical	Pooled	0.4000	0.4149	0.583	0.3566
2,000-char context	Pooled	0.4126	0.4231	0.583	0.3648
Canonical	HotpotQA	0.4412	0.4518	0.491	0.4027
Partial evidence	HotpotQA	0.3352	0.3797	0.491	0.3306

Increasing the context budget improves both policies and narrows the F1 difference from $+0.0150$ to $+0.0105$, while BeneRAG retains the utility advantage because its invocation rate remains 0.583. Under partial evidence, HotpotQA ALWAYS RAG falls by 0.1060 and BeneRAG by 0.0721, increasing the F1 difference from $+0.0106$ to $+0.0445$. The proportion of examples whose retrieved context contains both supporting documents falls from 15.1% to 0.8%, while the proportion containing neither rises from 28.3% to 53.4%; the residual 0.8% is consistent with the lookup-key mismatch described above reintroducing the removed paragraph from the query's own document, rather than an incomplete removal of the targeted paragraph.

H.5 RETRIEVAL-PROTOCOL SENSITIVITY AND RECOVERY

Table 18 isolates the effect of distractor strength by varying the number of distractors per source chunk while keeping the estimator and source threshold fixed.

Table 18: **Effect of distractor strength on frozen BeneRAG.** The number of distractors per source chunk is varied while the estimator and source threshold ($\tau^* = 0.21$) remain fixed.

Pool	RAG F1	BeneRAG F1	ΔF1	BeneRAG U_{call}	RAG U_{call}
1:0 (clean)	0.6305	0.5383	−0.0922	0.4800	0.5305
1:1	0.4517	0.4441	−0.0076	0.3858	0.3517
1:2	0.4117	0.4211	+0.0095	0.3628	0.3117
1:5 (canonical)	0.4000	0.4149	+0.0150	0.3566	0.3000

Across the four tested conditions, the frozen relative F1 advantage increases with distractor strength and changes sign between the 1:1 and 1:2 conditions. At the distractor-free extreme, the source operating point under-invokes even though the score retains useful benefit ranking. Table 19 then compares threshold recalibration with protocol-specific retraining to assess how much of the loss can be recovered.

Table 19: **Recovery under the distractor-free protocol.** Threshold recalibration uses validation labels from the shifted protocol without updating estimator weights. Protocol-specific retraining uses the same pre-retrieval observations with new benefit labels. AUROC is measured against the distractor-free benefit label.

Estimator	Ben. AUROC	F1	RR	U_{call}
ALWAYS RAG	–	0.6305	1.000	0.5305
Frozen ($\tau^* = 0.21$)	0.7018	0.5383	0.583	0.4800
Threshold recalibration ($\tau = 0.04$)	0.7018	0.6296	0.954	0.5343
Protocol-specific retraining ($\tau^* = 0.16$)	0.7682	**0.6315**	0.863	**0.5452**

Threshold recalibration recovers the utility advantage without changing ranking, although F1 remains slightly below ALWAYS RAG. Protocol-specific relabeling and retraining further improve benefit AUROC, F1, and utility. This diagnostic requires no new observable fields but does require labels under the shifted retrieval protocol; it is not a label-free deployment remedy.

I ERROR ANALYSIS

I.1 ERROR DEFINITIONS AND ANALYSIS PROTOCOL

We analyze two invocation errors on the canonical $10{,}000$-example test set. A *harmful invocation* occurs when BeneRAG invokes the context branch although its token-F1 score is lower than that of the no-retrieval branch,

$$r(q) = 1 \quad \text{and} \quad y_1(q) < y_0(q), \tag{11}$$

whereas a *missed benefit* occurs when BeneRAG skips context despite an improvement exceeding the canonical margin,

$$r(q) = 0 \quad \text{and} \quad y_1(q) - y_0(q) > 0.05. \tag{12}$$

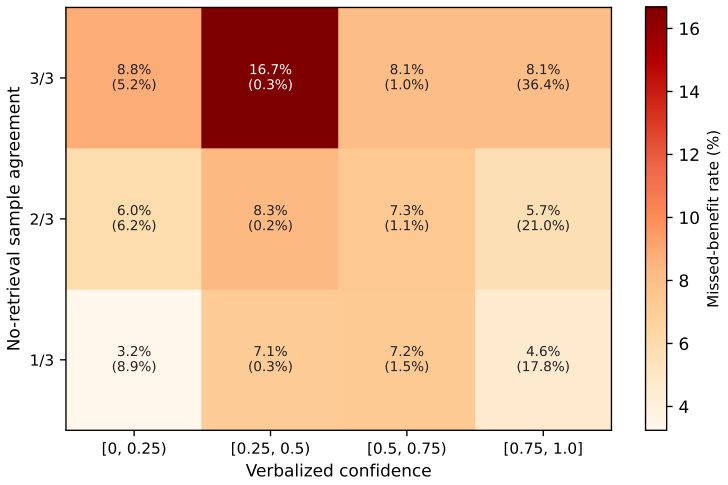


Figure 5: **Missed-benefit rate by observable invocation signals** on the canonical test set ($n = 10{,}000$). Cells are defined by no-retrieval sample agreement and verbalized-confidence ranges. Parentheses show the share of the full evaluation set in each cell.

The evaluation set contains 696 harmful invocations and 643 missed benefits. For qualitative diagnosis, we sample 20 examples for each error type and dataset, yielding 120 examples in total (seed 42).

I.2 Aggregate Failure Patterns

Figure 5 shows missed-benefit rates across sample-agreement and verbalized-confidence groups. Queries for which all three samples agree and confidence is at least 0.75 constitute 36.4% of the evaluation set. Within this group, 8.1% of queries are missed benefits. The group accounts for 295 of the 643 missed benefits, approximately 46% of all missed benefits.

Harmful invocations concentrate at intermediate agreement levels. Failure composition also differs by dataset: HotpotQA has the highest missed-benefit rate (7.6%) at an invocation rate of 0.491, while SQuAD has the highest harmful-invocation rate (9.1%) and the highest invocation rate (0.717). AdversarialQA has the lowest invocation rate (0.435; Table 4).

I.3 Qualitative Examples

Table 20 presents two successful decisions and two selected failures. Case 3 misses an improvement despite unanimous, high-confidence incorrect samples. In Case 4, retrieval replaces a correct no-retrieval answer with an incorrect response.

These examples show that sample agreement and verbalized confidence alone do not establish whether the no-retrieval answer is correct or whether the RAG branch will improve it. Case 3 illustrates a confidently repeated error that retrieval could repair, while Case 4 shows that invocation can discard a correct answer.

Table 20: Successful and failed invocation examples from the canonical test set (validation-selected $\tau^* = 0.21$). Correct answers are shown in green (✓), incorrect answers in black (✗), and estimator decisions in **[blue]**.

Case 1: Correct skip — unanimous, high-confidence samples for which skipping was correct

Question: What was the capital of India when the Taj Mahal was commissioned? *HotpotQA*
Gold: Agra **Samples** ($K = 3$)**:** "Agra" / "Agra" / "Agra" **Confidence:** 0.80
Estimator: score $0.113 < \tau^* = 0.21 \Rightarrow$ **[skip]**
No-retrieval answer: Agra ✓ (F1 1.00) **Context answer:** Delhi ✗ (F1 0.00) *(not invoked)*

Case 2: Correct invocation — inconsistent samples; context recovers the correct answer

Question: Who is the city supervisor of San Francisco? *SQuAD*
Gold: Jane Kim / Jane Kim / Kim
Samples ($K = 3$)**:** "Supervisor Aaron Peskin" / "Supervisor London Breed served as the acting m [. . .]" / "Supervisor London Breed served as the Acting M [. . .]"
Confidence: 0.85
Estimator: score $0.249 > \tau^* = 0.21 \Rightarrow$ **[invoke]**
No-retrieval answer: Supervisor Aaron Peskin ✗ (F1 0.00) **Context answer:** Jane Kim ✓ (F1 1.00)

Case 3: Missed benefit — unanimous but incorrect samples

Question: What movie, For the Bible Tells Me So or Spencer Halpin's Moral Kombat, changed the name of the title to avoid the risk of a lawsuit? *HotpotQA*
Gold: Spencer Halpin's Moral Kombat
Samples ($K = 3$)**:** "For the Bible Tells Me So" / "For the Bible Tells Me So" / "For the Bible Tells Me So"
Confidence: 1.00
Estimator: score $0.074 < \tau^* = 0.21 \Rightarrow$ **[skip]**
No-retrieval answer: For the Bible Tells Me So ✗ (F1 0.00) **Context answer:** Spencer Halpin's Moral Kombat ✓ (F1 1.00) *(not invoked)*

Case 4: Harmful invocation — the context branch scores lower than the no-retrieval branch

Question: What was the first Super Bowl branded with Roman numerals? *SQuAD*
Gold: Super Bowl V / Super Bowl V / V **Samples** ($K = 3$)**:** "Super Bowl V" / "Super Bowl V" / "Super Bowl V" **Confidence:** 1.00
Estimator: score $0.407 > \tau^* = 0.21 \Rightarrow$ **[invoke]**
No-retrieval answer: Super Bowl V ✓ (F1 1.00) **Context answer:** Not mentioned ✗ (F1 0.00)