DIFFERENTIAL NEURAL RERANKING: UPDATING RANKINGS WITHOUT RERANKING EVERYTHING

Anonymous authors
Paper under double-blind review

ABSTRACT

A revised query may need a new ranking, but not a rescore of every candidate. We ask how few candidates must be rescored to reproduce the top-k of a full rerank, and propose *differential neural reranking*, which uses the previous ranking to choose them. Across eight retrieval domains and two reranker architectures, revisions shift nearly every candidate's score. Much of this movement is shared across candidates and preserves their order, while the candidate-specific changes that alter the top-k concentrate near the cutoff. On 273 successive session revisions, an exact oracle needs only 2.65 of 64 fresh evaluations on average to recover the ordered top-10. A two-parameter model predicts cutoff crossings across rerankers and benchmarks without refitting, capturing the rank structure that determines which candidates require fresh evaluation. On fixed candidate pools, differential reranking speeds up the reranking stage by about $6\times$ on GPU at fidelity nDCG@10 above 0.99. The effect persists in end-to-end pipelines, with savings large enough to offset the added scoring cost as the candidate pool changes. On 250 held-out TopiOCQA conversations, with the policy and evaluation protocol fixed in advance, differential reranking reduced mean end-to-end latency by 14.40%, outperformed a BM25 baseline with the same reranker budget, and kept fidelity nDCG@10 at or above 0.98 on every later turn in 249 of 250 conversations.

1 INTRODUCTION

Neural rerankers start from scratch whenever a query changes, even after a small revision. We ask whether an existing ranking can instead be *updated* with less computation.

In a common two-stage search pipeline (Cohere, 2026; Sentence Transformers, 2026; Vespa.ai, 2026), a fast sparse or dense retriever returns a small *candidate pool* from a large corpus, and a more accurate neural reranker scores and sorts it. Cross-encoders read the query and document together, which makes them effective but requires a separate forward pass for every query-document pair (MacAvaney et al., 2020; Yadav et al., 2022).

Related queries arise in conversational search and query reformulation (Mo et al., 2023; Jang et al., 2024). TopiOCQA conversations, for example, average 13 turns across four Wikipedia topics, and only 28% of turns draw their answer from a different document than the previous turn (Adlakha et al., 2022). When the query changes, the new pool mixes new documents, which must be scored, with documents already scored and ranked on the previous turn (Figure 1).

Updating previously computed results instead of recomputing them is well studied outside neural ranking: database systems update materialized views when the query definition changes (Gupta et al., 1995), and classical top-k algorithms minimize the number of expensive evaluations needed to identify the highest-ranked k objects (Chang & Hwang, 2002). Cross-encoders lack such an update rule, since nothing tells us which old scores remain usable after the query changes. Prior work reduces neural reranking cost through document-side precomputation (MacAvaney et al., 2020; Dai et al., 2026), approximate cross-encoder search (Yadav et al., 2022), adaptive score acquisition (Rathee et al., 2025), and ranked-list truncation (Meng et al., 2024). Concurrent work (EARM) reuses relevance judgments across a stream of related queries (Feng et al., 2026). We instead frame query revision as a *minimum-recomputation problem*: after a query changes, what is the smallest number of candidates that must be rescored to get the same top-k as a full rerank?

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.

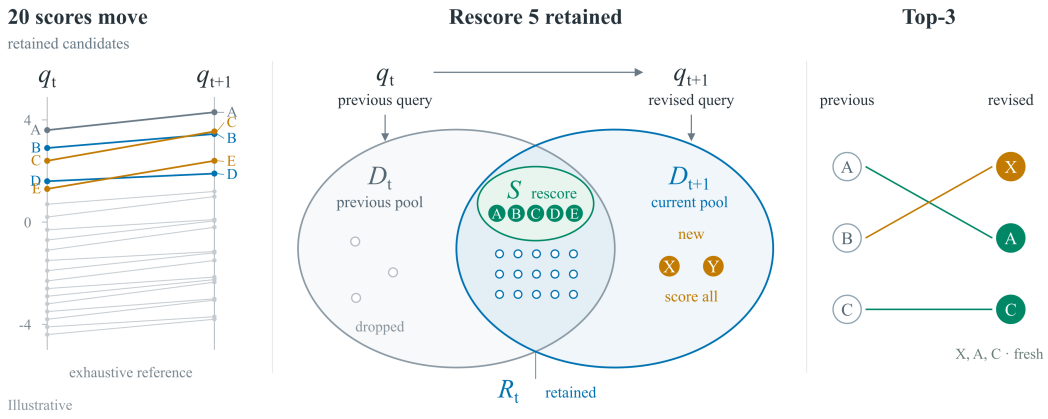

Figure 1: One query revision from q_t to q_{t+1}. Differential reranking rescores a subset S of retained candidates R_t, here the old top five (A–E), and scores every new document (X, Y). Documents only in D_t are dropped. Although all 20 retained scores change, these seven evaluations recover the full rerank's top-3 ($A, B, C \to X, A, C$).

Let D_t be the candidate pool retrieved and reranked for query q_t. After revision to q_{t+1}, retrieval returns a new pool D_{t+1}. The *retained* candidates $R_t = D_t \cap D_{t+1}$ have scores under q_t; the *new* candidates $E_{t+1} = D_{t+1} \setminus D_t$ require scoring. We rescore a subset $S \subseteq R_t$ under q_{t+1} and keep the old scores for the rest. Let $\tilde{\mathbf{s}}^S_{t+1}$ denote the mixed scores: new scores for E_{t+1} and S, and old scores for $R_t \setminus S$. If these scores yield the same top-k as a full rerank, candidates outside S need no rescoring. The smallest such set is

$$S^* = \arg\min_{S \subseteq R_t} |S| \quad \text{s.t.} \quad \mathrm{TopK}\big(\tilde{\mathbf{s}}^S_{t+1}\big) = \mathrm{TopK}(\mathbf{s}_{t+1})\,,$$

where $\mathbf{s}_{t+1}$ is the full rerank of D_{t+1} under q_{t+1} and TopK returns the ordered top-k. Computing S^* requires all the new scores, so it serves as an oracle that measures, in hindsight, how much of a full rerank was necessary; a practical method must choose which candidates to rescore before seeing their new scores. We compute the oracle exactly with a binary linear program in which each retained candidate either keeps its old score or receives its new score, and the constraints require the full rerank's top-k to stay in the same order and above every other candidate, with deterministic tie handling. All instances were solved to optimality (Appendix B.1).

For intuition, suppose a user revises "novels set in Paris" to "novels set in Paris published before 1950." Nearly every score drops, but high-ranked documents drop by similar amounts and stay high, so only candidates near the top-k cutoff are likely to cross it. Our analysis refines this picture: much of the score change is an order-preserving shift shared by all candidates, each revision type changes scores in a consistent way, and related candidates move together, so real revisions cause far fewer crossings than independent random changes of the same size. A model of each candidate's distance to the cutoff, adjusted for how many candidates score near it, predicts which candidates cross and transfers across rerankers and benchmarks.

Differential reranking turns this picture into a rule. It scores every new candidate and, among the retained candidates, rescores a block at the top of the previous ranking that extends a little below the old cutoff. Because predicted crossing risk falls with distance from the cutoff, the candidates whose risk exceeds any fixed threshold always form such a block, so the block depth is the method's one essential parameter (Section 5). Batching, quantization, faster rerankers, and cached document representations make each evaluation cheaper, whereas differential reranking reduces the number of evaluations, so the two can be combined.

We evaluate on controlled revisions over fixed candidate pools and on TopiOCQA conversations with full retrieval (Section 2). The conversational evaluation measures how pool overlap limits the savings available in practice (Sections 7 and 9).

Our contributions are (1) a formulation of the minimum recomputation needed after a query revision, with an exact oracle; (2) evidence that most score movement is shared across candidates and that

changes to the top-k occur near the cutoff, together with a crossing model showing why rescoring a block at the top of the old ranking suffices; and (3) a policy derived from that model and evaluated for ranking fidelity, GPU speedup, and end-to-end latency, along with an analysis of how pool overlap between turns limits the available savings.

2 Experimental Setting

Controlled revisions use deterministic text transformations over fixed 64-candidate pools. Both queries are scored offline; replays reveal scores only when requested by a policy. MultiConIR (Lu et al., 2025) provides queries with explicit conditions in five domains (Books, Legal, Medical, Movies, People; 1,779 revisions), allowing one condition to be added, removed, or negated. Three BEIR domains (Thakur et al., 2021) (ArguAna, NFCorpus, SciFact; 420 revisions) provide less structured changes and more closely spaced candidate scores. A multi-edit set applies nine edit types to 91 source queries (819 revisions); 91 four-query sessions add conditions successively (273 revisions). Qwen3-Reranker-8B (Zhang et al., 2025) scores all 2,199 revisions; mxbai-rerank-large-v1 (Shakir et al., 2024) scores a matched subset of 1,380 (Appendix A.1). Several analyses use the 1,815 Qwen revisions that change the ordered top-10, termed *ranking-changing* revisions. MultiConIR pools combine judged candidates with seeded, duplicate-free sampling from other sources in the same domain; BEIR pools combine judged relevant documents with unjudged corpus documents selected by identifier order. The 91 session sources were hash-partitioned into calibration, development, and confirmation splits.

For TopiOCQA, Anserini BM25 ($k_1 = 0.9$, $b = 0.4$) retrieves a new top-64 pool each turn from the complete official Wikipedia passage corpus. Retrieval and reranking use the same *answer-free question history*: the current question concatenated with all previous user questions, excluding answers, rationales, topics, future turns, and answer-derived rewrites. We chose the policy on development conversations and used a separate sample of 250 held-out conversations only for final evaluation (Section 8).

Unless stated otherwise, $k = 10$ and ties are broken by document identifier. We measure agreement with the full rerank, rather than relevance to gold labels. *Exact ordered agreement* requires the same top-10 in order; *membership agreement* requires the same ten documents. *Fidelity nDCG@10* uses the full rerank as ground truth and graded gains that decrease linearly with rank, so an exact match scores 1.0. We also count reranker evaluations and, where measured, wall-clock latency.

3 How Many Old Scores Must Be Recomputed

The previous ranking cannot simply be reused, because most revisions change the ordered top-10, but the change is usually small. About one document enters the top-10 per revision (the union of the old and new top-10 averages 10.57 candidates for Qwen), usually from just below the old cutoff, and candidates below old rank 20 almost never enter (Appendix B.1).

Knowing which documents enter the top-10 does not yet tell us which scores to recompute, because a ranking that mixes old and new scores can misorder candidates even when the right documents were rescored. We therefore solve the minimum-recomputation problem exactly for each revision. On all 273 successive revisions from the 91 condition-addition sessions, including those with unchanged rankings, recovering the exact revised top-10 requires 2.65 fresh evaluations out of 64 on average (95% source-cluster bootstrap CI 2.31–3.01; median 2); a quarter of revisions need none, and 90% need at most six (Figure 2a). Across the 1,815 ranking-changing Qwen revisions, the needed evaluations concentrate around the old cutoff (Figure 2b). A full rerank therefore spends most of its computation on candidates whose old scores could have been kept. Because the oracle sees every new score, this bounds what can be saved; the rest of the paper asks how close a policy that must choose in advance can come.

Two old positions deserve attention. The first candidate below the cutoff matters disproportionately: rescoring old ranks $1, \ldots, 11$ instead of $1, \ldots, 10$ raises exact top-10 membership from 54.4% to 74.7% for Qwen, with similar gains at every cutoff and for Mxbai, so the effect comes from the cutoff's position rather than rank 11 itself (Appendix B.2). At the other end, at least one of old

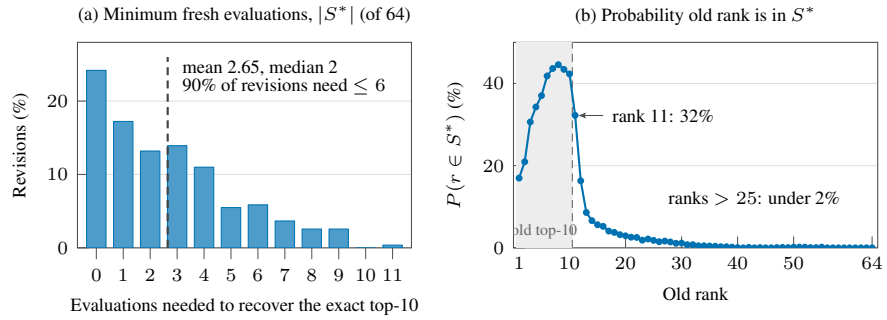

Figure 2: The work a full rerank does versus the work the revised top-10 needs. (a) On 273 successive revisions, recovering the revised top-10 exactly needs few of the 64 possible evaluations. (b) Across 1,815 ranking-changing Qwen revisions, the needed evaluations concentrate around the previous cutoff; shaded ranks were in the old top-10.

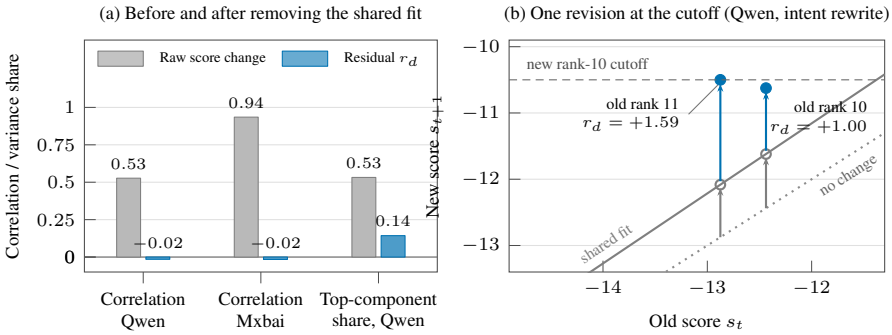

Figure 3: Shared versus candidate-specific movement. (a) Raw score changes are strongly correlated across candidates; after a per-revision affine fit is removed, the residuals are nearly uncorrelated. (b) In one revision, the shared fit (grey arrows) raises both documents at the cutoff and keeps their order, while their residuals (blue arrows) swap old ranks 10 and 11. Hollow and filled markers show fitted and actual new scores, respectively.

ranks 1–3 leaves the top-10 in only 0.88% of ranking-changing Qwen revisions, yet those candidates participate in reordering in 47.9%. They matter more for order than for membership (Appendix B.2).

4 Why Broad Score Changes Rarely Change the Top-k

A revision changes almost every score, yet only about one document enters the top-10. To see why, we split each candidate's score change into a part shared by the pool, which cannot change the order, and a candidate-specific part, which can:

$$s_{t+1}(d) = a_t\, s_t(d) + b_t + r_d,$$

where a_t and b_t are fitted per revision over the pool and the residual r_d is what the shared fit leaves unexplained. When $a_t > 0$, the shared part applies the same increasing map to every score, so it can move logits far without changing the order, and only the residuals reorder candidates. Raw score changes are strongly correlated across candidates (mean pairwise correlation 0.53 for Qwen and 0.94 for Mxbai), whereas the residuals are nearly uncorrelated (Figure 3a): a revision mostly moves the pool as a whole, and the part that reorders candidates is much smaller (Figure 3b).

The residuals depend on the kind of revision. Adding a condition lowers scores on average and moves the most candidates into the top-10 (31% entry rate), whereas a synonym paraphrase barely moves anything (4%), even after accounting for old rank and the number of candidates near the cutoff. Document edits agree: removing or contradicting the clause that satisfies a newly added condition lowers a document's score more than removing unrelated text of equal length (Appendix C).

Algorithm 1 Differential neural reranking for one revision

Require: previous full ranking (D_t, s_t), revised query q_{t+1}, new pool D_{t+1}, cutoff k, depth m, margin λ

1: $E \leftarrow D_{t+1} \setminus D_t$; score E under q_{t+1} ▷ new candidates
2: $S \leftarrow$ top-m of R_t by s_t; score S under q_{t+1} ▷ top block of old ranking
3: **loop** ▷ optional expansion
4: $\quad \hat{c} \leftarrow k$-th largest fresh score; $\hat{\mu} \leftarrow$ mean score change over S
5: $\quad T \leftarrow \{d \in R_t \setminus S : s_t(d) + \hat{\mu} + \lambda > \hat{c}\}$
6: $\quad$ **if** $T = \emptyset$ **then break**
7: $\quad$ score T under q_{t+1}; $S \leftarrow S \cup T$
8: **return** top-k of $E \cup S$ by new scores

A candidate below the cutoff enters the top-k only if its residual exceeds its old margin to the cutoff. The same margin is riskier when many competitors sit near the cutoff. We capture both effects with a two-parameter logistic model,

$$\hat{p}_t(d) \;=\; P(d \text{ crosses the cutoff}) \;=\; \sigma\Big(\beta_0 - \beta_1\, \tilde{\Delta}_t(d)\Big), \qquad \beta_1 > 0,$$

where $\tilde{\Delta}_t(d)$ is d's old margin to the rank-k score, normalized by the spread of old scores near the cutoff and adjusted for the number of candidates within one logit of it (Appendix C). Fitted only on Qwen–MultiConIR (ROC-AUC 0.99), the model predicts crossings without refitting on BEIR (0.91), for Mxbai (0.94), and for both together (0.84), with calibration correlations of at least 0.988, so the pattern is not tied to one score scale or dataset.

Real revisions also cross the cutoff far less often than independent random changes of the same size: at an old margin of 1.0–1.5 logits, 0.70% of Qwen candidates cross, against 6.1% under independent Gaussian changes with the same variance, an $8.7\times$ gap (Figure 7). Neighboring candidates move together, so for adjacent ranks the variance of the difference between their score changes is only 0.29 of the independent case, and their relative order is more stable than their individual scores.

5 Differential Neural Reranking

After a full rerank of D_t under q_t, the system stores each candidate's identifier, score, and rank with the query that produced them. We call this state, in which every candidate was scored under the same query, a *full ranking*.

Section 4 suggests rescoring a retained candidate when its risk $\hat{p}_t(d)$ of entering the top-k is high. We always rescore candidates in the old top-k to recover the revised order. For other candidates, both normalizers in $\tilde{\Delta}_t(d)$ are computed once per revision. Thus $\tilde{\Delta}_t(d)$ grows with the old margin $s_t(\pi_k) - s_t(d)$, while $\hat{p}_t(d)$ falls with worse old rank. Thresholding risk at any ϵ therefore selects a prefix of the old ranking,

$$\{d \in R_t : \mathrm{rank}_t(d) \le k\} \;\cup\; \{d \in R_t : \hat{p}_t(d) > \epsilon\} \;=\; \{d \in R_t : \mathrm{rank}_t(d) \le m(\epsilon)\}$$

for some depth $m(\epsilon) \ge k$, up to ties. Choosing a risk level therefore amounts to choosing a depth m. A contiguous block has a second advantage. Because much of the score movement is shared, old and new scores lie on different scales, and a few rescored candidates ranked against neighbors that kept their old scores can swap with them through this scale difference alone. Rescoring a contiguous block puts close competitors on the same scale.

The prefix rule uses old scores only; after rescoring its first block, new scores locate the revised cutoff. Let $\hat{c}$ be the k-th largest fresh score and $\hat{\mu}$ the block's mean score change, which estimates the shared shift when $a_t \approx 1$. An unrescored retained candidate d has estimated new score $s_t(d) + \hat{\mu}$. The expansion step rescores it when its estimated margin to the new cutoff is below a safety margin λ, that is, $s_t(d) + \hat{\mu} + \lambda > \hat{c}$. This removes the shared shift from the same margin threshold. Since the estimated new score increases with $s_t(d)$, added candidates form the next block of the old ranking. The loop repeats until none pass, or becomes a full rerank if every candidate is rescored (Algorithm 1). The margin λ plays the role of the risk level ϵ and can be calibrated with conformal risk control (Section 6).

Table 1: Controlled revisions with a fixed 64-candidate pool. In the offline policies, candidates that are not rescored keep their old scores. Top: 1,815 ranking-changing Qwen revisions. Bottom: measured GPU reranking stage on 91 multi-turn session cases. Speedup is reranking-stage wall-clock time.

Policy	Eval.	Skipped	Exact set	Exact order	Fid. nDCG@10	Speedup
Full rerank	64	0%	100%	100%	1.000	–
Old top-m, $m = 10$	10	84.4%	54.4%	50.1%	0.994	–
Old top-m, $m = 11$	11	82.8%	74.7%	73.4%	0.995	–
Differential, $\lambda = 0.4$	12.61	80.3%	87.6%	86.9%	0.997	–
Differential, $\lambda = 0.8$	14.82	76.8%	95.1%	94.7%	0.999	–
Old top-m, $m = 20$	20	68.8%	94.8%	94.8%	0.998	–
Differential, Qwen3-8B	12.41	80.6%	85.7%	58.2%	0.993	6.17×
Differential, Mxbai-large	15.71	75.4%	90.1%	69.2%	0.992	5.97×

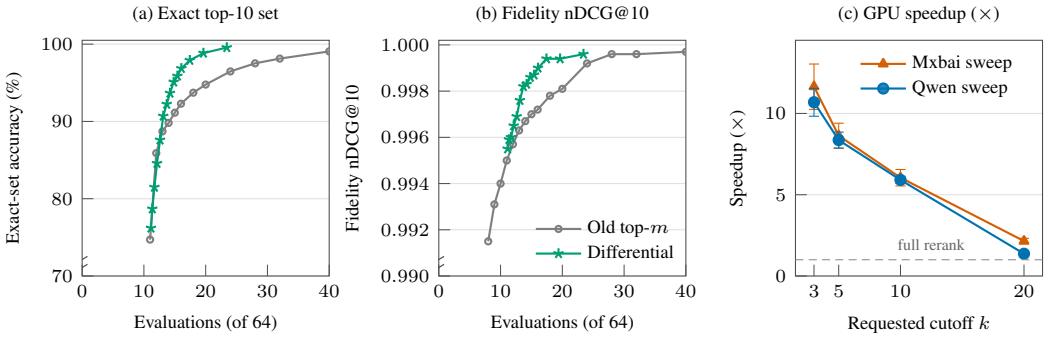

Figure 4: Cost–fidelity trade-off and measured GPU speedup. (a) Exact top-10 set accuracy and (b) fidelity nDCG@10 for offline policies on 1,815 ranking-changing Qwen revisions; grey rescores old ranks $1{:}m$ and green sweeps the differential margin λ. Axis breaks mark truncated high-fidelity ranges; lower-fidelity points fall outside them. (c) Measured GPU reranking-stage speedup on 91 multi-turn session cases with 95% CIs, for Mxbai and Qwen across output cutoffs k.

On fixed pools (Section 6), $E = \emptyset$, the first block covers old ranks $1, \ldots, k{+}1$, the expansion loop is on, and unrescored candidates keep their old output scores. In conversations (Section 8), each turn adds a full user question and many candidates are new, making the shift estimated from the first block less reliable. We also evaluated adaptive stopping and fallback rules on development data (Appendix D.3). The conversational policy uses only the prefix rule: it scores all new candidates, rescores retained candidates among the top $m = 32$ in the last full ranking, and returns the top-k using new scores only. We chose $m = 32$, the smallest depth at which every development conversation maintained fidelity nDCG@10 ≥ 0.99 on every later turn. If the rule would rescore the whole pool, the policy reranks fully. After a partial update, the next turn also reranks fully, so every update starts with scores from one query, as required by the decomposition in Section 4.

The cost of a revision is $|E_{t+1}| + |S_{t+1}|$ evaluations, plus, over a conversation, the full reranks on the first turn and after each partial update.

6 Revisions over a Fixed Candidate Set

Fixed pools isolate the value of the previous ranking from the cost of new candidates. Table 1 and Figure 4 show the trade-off between evaluations and fidelity. Rescoring only the old top 10 reproduces the exact ordered top-10 in half of the ranking-changing revisions, and adding rank 11 ($m = 11$) raises this to 73%. The expansion step reaches 94.7% with 14.8 evaluations ($\lambda = 0.8$), whereas a fixed block without expansion needs $m = 20$ for similar accuracy. With split conformal risk control (Angelopoulos et al., 2024) and a 1% error target on held-out MultiConIR revisions,

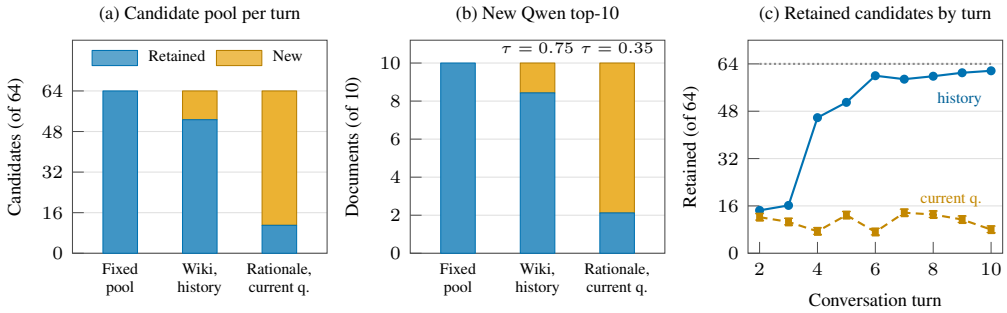

Figure 5: How much of the pool carries over between turns. (a) Newly retrieved candidates must be scored, which limits how much reranking any update method can avoid. (b) With question-history retrieval over Wikipedia, most of the new top-10 was already in the previous pool, and retained candidates kept more of their order (Kendall τ). (c) With question-history retrieval, the retained share grows as a conversation settles on a topic; with the current question alone it stays low. Panels (a) and (b) use a four-conversation Wikipedia sample and a ten-conversation rationale-passage study; panel (c) uses a six-conversation Wikipedia pilot and the same rationale study.

the calibrated policy used 13.3 evaluations and missed a top-10 document on 0.11% of test queries (Appendix D.1).

As Section 4 predicts, results depend on how many candidates score close to the cutoff. Exact agreement is 89% on ArguAna, where ranks 10 and 11 are about 1.2 logits apart, but 36% on SciFact, where near-identical evidence sentences crowd the cutoff (0.08 logits); fidelity nDCG@10 stays at 0.993 even there.

On GPU, the reranking stage ran $6.17\times$ faster with the decoder-based Qwen and $5.97\times$ faster with the encoder-based Mxbai, both above 0.99 fidelity nDCG@10, with about 0.3 ms per query spent choosing candidates. Speedups fall from about $11\times$ at $k = 3$ to 1.4–$2.2\times$ at $k = 20$ as more candidates must be rescored (Figure 4c).

The method also handles explicit document insertions and deletions (about 11 of 64 evaluations, fidelity 1.000 against 0.987 when retained candidates keep their old scores) and three-turn sessions, where it used about a fifth of full reranking's evaluations (Appendix D.1). All of these results assume that most candidates carry over between queries, which a real retriever does not guarantee.

7 Pool Overlap Between Turns

The *overlap* between consecutive pools determines how many candidates have reusable scores. In a four-conversation diagnostic, retrieving with the current question alone made 63.8 of 64 candidates new on each later turn, leaving no savings, so we retrieve with the answer-free question history instead.

Overlap saves work only when retained documents make up much of the new top-k and previously ranked within the rescored block. With question-history retrieval over full Wikipedia, 8.4 of the new Qwen top-10 were already in the previous pool, all within the old top 32, and retained candidates largely kept their order (Kendall $\tau = 0.75$). In a ten-conversation study over a smaller rationale-passage collection with current-question retrieval, most of the new top-10 was new and retained candidates were reordered much more ($\tau = 0.35$, Figure 5). Because the setups also differ in corpus and query format, this contrast is only descriptive.

Query format matters for reranking too: in a four-conversation study, old scores helped only when the previous full ranking used the same format as the current query (exact top-10 on 37 of 37 later turns, against at most 13 otherwise). Almost all reuse came from the immediately preceding turn, so storing only the last full ranking loses little (Appendix E.1). Better retrieval and higher overlap can also pull in different directions: query rewrites from a released T5-QReCC model (Anantha et al., 2021; Elgohary et al., 2019) retrieved the gold passage more often than the question history

Table 2: Primary live serving results on 250 held-out TopiOCQA conversations. Fidelity is measured against full Qwen reranking. The exact-order margin is conversation-averaged over warm turns; intervals are shown where available.

Measure	Result	95% interval / bound
Mean latency reduction	14.40%	13.84–14.93%
P95 latency ratio	0.886	0.866–0.903
Qwen evaluations saved	27,896 (14.94%)	exact count
Fidelity nDCG margin vs. same-budget BM25	+1.39 pp	+1.26–+1.53 pp
Exact ordered top-10 margin vs. BM25	+13.16 pp	–
Conversation reliability at nDCG ≥ 0.98	249/250	lower bound 98.12%
Conversation reliability at nDCG ≥ 0.99	245/250	lower bound 95.84%

but overlapped much less between turns (Appendix E.3). The final evaluation holds retrieval fixed, and we do not claim that maximizing overlap is the best retrieval strategy.

8 Conversational Evaluation with Full Retrieval

We evaluated the fixed policy ($m = 32$ with the rules of Section 5) on 250 held-out TopiOCQA conversations, processed turn by turn, with sample size and stopping rule set before inspecting results. On each turn, BM25 (Lin et al., 2021) retrieved a new top-64 pool from full Wikipedia, on which three methods ran: full Qwen reranking, differential reranking, and a *same-budget BM25 baseline*. Full reranking supplied the reference ranking but did not influence candidate selection. The baseline spent exactly as many Qwen evaluations as differential reranking used on that turn, but on the top BM25 candidates, testing whether the previous ranking selects better candidates than the retriever's own order. Latency covers the complete request, from query construction and retrieval to Qwen inference and sorting, excluding one-time model and index loading. Evaluation counts include first turns and the full reranks after partial updates. Conversations, not turns, are the unit of statistical inference (Appendices E.2 and F).

Differential reranking reduced mean end-to-end latency per conversation by 14.40% (95% CI 13.84–14.93%), with a 95th-percentile latency ratio of 0.886 (0.866–0.903), and saved 27,896 of 186,688 Qwen evaluations (14.94%). The latency reduction, which includes candidate selection and cache updates, closely tracks the reduction in Qwen calls (Table 2, Figure 6a). At the same Qwen budget, differential reranking improved fidelity nDCG@10 by 1.39 points (1.26–1.53) and conversation-averaged exact ordered top-10 on warm turns by 13.16 points (74.15% versus 60.99%) over the BM25 baseline (Figure 6b). These live exact-order rates include near-tie swaps caused by BF16 score differences between inference batches, even when the selected top-10 set is unchanged. With common reference scores for the selected candidates, differential reranking reaches 99.41% exact order; this diagnostic is not a live serving result (Appendix E.2).

At the 0.98 fidelity threshold, 249 of 250 conversations passed on every later turn (one-sided exact 95% lower bound 98.12%). At the stricter 0.99 threshold, 245 passed (Figure 6c), with a lower bound of 95.84%; only the bound adjusted for six comparisons, 94.73%, falls below 95%. Each of the five failures came from a single partial turn on which one full-rerank top-10 document lay outside the rescored block, a rare deep crossing that the prefix rule accepts missing. The scheduled full rerank on the next turn restored fidelity to 1.0. Development-only comparisons with an Mxbai-guided selector and an EARM-style reuse adaptation appear in Appendix D.2.

9 Where the Savings Come From

On fixed pools the policy skips 75–80% of evaluations; in conversations it skips about 15%. Four factors explain the difference. First, every new candidate must be scored, so the cost approaches a full rerank as the number of new candidates approaches the pool size; in the held-out run, about two thirds of all later-turn evaluations went to newly retrieved documents. Second, overlap helps only when enough documents in the new top-k were scored before and ranked high enough to be rescored. Third, an old score is useful only when the query that produced it has the same format

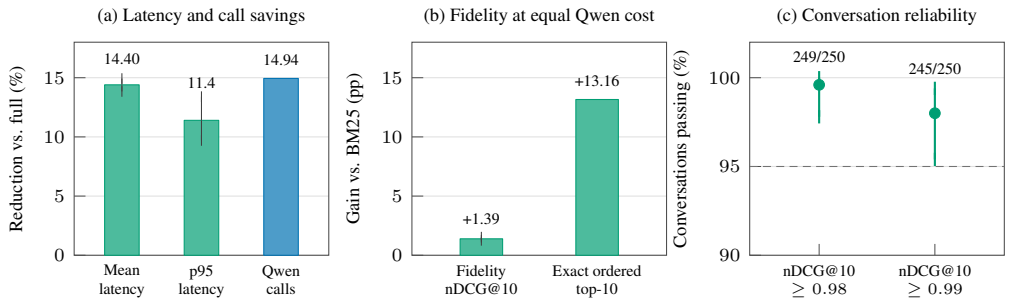

Figure 6: Evaluation on 250 held-out conversations (2,917 turns). (a) Latency and call reductions include first turns and all full reranks; mean latency uses conversation-level totals. (b) Fidelity gains over equal-call BM25; the exact-order gain is a conversation average over warm turns. (c) Conversations meeting each fidelity threshold on every later turn; the dashed line marks 95%. Whiskers show 95% intervals where available and two-sided Clopper–Pearson intervals for conversation reliability. Call savings are exact counts.

as the current one. Fourth, a cautious rescoring depth and the periodic full reranks add evaluations. The last factor is large: of the 2,667 later turns, only 1,227 were partial updates, which saved 35.5% of the evaluations those turns would otherwise have needed; the rest were scheduled full reranks after a partial update (1,094) or turns on which new and rescored candidates already covered the whole pool (346). Relaxing the depth trades fidelity for savings: in an offline replay on the held-out conversations, $m = 24$ raised call savings from 14.9% to 20.2%, while the number of conversations above 0.99 fidelity on every later turn fell from 245 to 239 (Appendix D.4).

The oracle provides a reference point for the practical policy's evaluation cost. On fixed pools, the oracle needs 2.65 evaluations on average, against 11–15 for the adaptive policy. When most candidates are new each turn, the oracle, given a fully scored previous turn, needs 3.43 retained rescores after all new candidates are scored. The gap has three sources: the oracle chooses after seeing every new score, while the policy must choose before; the oracle may keep old scores in its output, while the conversational policy outputs only new scores; and the policy rescores a contiguous block rather than arbitrary candidates. We also evaluated proxy rerankers and richer learned crossing predictors; results are reported in Appendix D.3.

Because the top-k set changes less often than its order, applications that need only the set, such as assembling context for retrieval-augmented generation, can reuse more scores: on fixed pools, a variant that skips old ranks 1–3 used 9.68 evaluations and recovered the exact top-10 set on 86.6% of revisions.

10 Conclusion

A revised query often changes the neural ranking, but most previous scores remain usable: much of the score movement is shared, and changes that alter the top-k concentrate near the old cutoff.

Differential reranking uses the existing ranking to choose which candidates to rescore, and because crossing risk falls with distance from the cutoff, those candidates form a block at the top of the old ranking. On fixed pools, it skips most reranker evaluations; in the held-out conversational evaluation, it reduced mean end-to-end latency by 14.40%, outperformed the same-budget BM25 baseline, and kept every later turn at or above 0.98 fidelity in 249 of 250 conversations.

The savings depend on how many candidates carry over between queries and on how well old ranks predict relevance under the revised query. This paper evaluates the update method and its end-to-end behavior under one fixed retrieval setup; it does not evaluate downstream answer generation or claim that this retrieval formulation is optimal for recall. Future work can study retrievers that optimize both recall and overlap between turns, cheap signals that bound a candidate's score before the full reranker runs, adaptive schedules for the full reranks that currently follow every partial update, and better ways to choose which stored ranking to reuse.

AI USE STATEMENT

For transparency, we used LLMs for research, including identifying potential novelty collisions; coding, including implementing and running experiments we designed; drafting, including describing methodology and converting text to LaTeX; and polishing, including correcting grammar and providing feedback. We did not use generative AI to create or edit datasets. A human manually verified all LLM-generated writing, code, and cited papers before use. Humans have always led the research direction and system design, while AI assisted with execution. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A.1 describes the revision sets, the TopiOCQA sample construction, the query representations, and the retrieval configuration, and Appendix A.2 the reranker and serving setup. Appendix B.1 gives the exact minimum-recomputation formulation. Algorithm 1 and Appendix D.1 specify the policies and their parameters, Appendix D.2 the baselines, and Appendix F the statistical procedures and resampling units. Code and configuration are available at `https://anonymous.4open.science/r/ICLR-2027_diff_neural_reranking/README.md`.

REFERENCES

Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy. TopiOCQA: Open-domain conversational question answering with topic switching. *Transactions of the Association for Computational Linguistics*, 10:468–483, 2022. doi: 10.1162/tacl_a_00471. URL https://aclanthology.org/2022.tacl-1.27/.

Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. Open-domain question answering goes conversational via question rewriting. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 520–534, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.44. URL https://aclanthology.org/2021.naacl-main.44/.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=33XGfHLtZg.

Kevin Chen-Chuan Chang and Seung-won Hwang. Minimal probing: Supporting expensive predicates for top-k queries. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 346–357. Association for Computing Machinery, 2002. doi: 10.1145/564691.564731. URL https://doi.org/10.1145/564691.564731.

Cohere. Master reranking with cohere models. https://docs.cohere.com/docs/reranking-with-cohere, 2026. Accessed 2026-09-23.

Beiya Dai, Yifan Wei, Guang Yang, Xing Shi, Xinbing Wang, and Zhouhan Lin. DoPR: Reusable compressed document prefixes for efficient LLM reranking, 2026. URL https://arxiv.org/abs/2609.03311.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. Can you unpack that? learning to rewrite questions-in-context. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5918–5924, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1605. URL https://aclanthology.org/D19-1605/.

Qi Feng, Chris Ding, and Jicong Fan. The retriever should remember: Experience-amortized reranking for long-term agent memory, 2026. URL https://arxiv.org/abs/2608.22767.

Ashish Gupta, Inderpal Singh Mumick, and Kenneth A. Ross. Adapting materialized views after redefinitions. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp. 211–222. ACM Press, 1995. doi: 10.1145/223784.223817. URL https://doi.org/10.1145/223784.223817.

Yunah Jang, Kang-il Lee, Hyunkyung Bae, Hwanhee Lee, and Kyomin Jung. IterCQR: Iterative conversational query reformulation with retrieval guidance. 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. 8121–8138, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.449. URL https://aclanthology.org/2024.naacl-long.449/.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pp. 2356–2362, 2021. doi: 10.1145/3404835.3463238. URL https://doi.org/10.1145/3404835.3463238.

Xuan Lu, Sifan Liu, Bochao Yin, Yongqi Li, Xinghao Chen, Hui Su, Yaohui Jin, Wenjun Zeng, and Xiaoyu Shen. MultiConIR: Towards multi-condition information retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13471–13494, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.726. URL https://aclanthology.org/2025.findings-emnlp.726/.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 49–58, 2020. doi: 10.1145/3397271.3401093. URL https://arxiv.org/abs/2004.14255.

Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. Ranked list truncation for large language model-based re-ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 141–151, 2024. doi: 10.1145/3626772.3657864. URL https://doi.org/10.1145/3626772.3657864.

Fengran Mo, Kelong Mao, Yutao Zhu, Yihong Wu, Kaiyu Huang, and Jian-Yun Nie. ConvGQR: Generative query reformulation for conversational search. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4998–5012, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.274. URL https://aclanthology.org/2023.acl-long.274/.

Mandeep Rathee, Venktesh V, Sean MacAvaney, and Avishek Anand. Breaking the lens of the telescope: Online relevance estimation over large retrieval sets. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2287–2297, 2025. doi: 10.1145/3726302.3729910. URL https://arxiv.org/abs/2504.09353.

Sentence Transformers. Cross encoder usage. https://www.sbert.net/docs/cross_encoder/usage/usage.html, 2026. Accessed 2026-09-23.

Aamir Shakir, Darius Koenig, Julius Lipp, and Sean Lee. Boost your search with the crispy mixedbread rerank models. https://www.mixedbread.ai/blog/mxbai-rerank-v1, 2024.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Vespa.ai. Ranking with transformer cross-encoder models. https://docs.vespa.ai/en/ranking/cross-encoders, 2026. Accessed 2026-09-23.

Nishant Yadav, Nicholas Monath, Rico Angell, Manzil Zaheer, and Andrew McCallum. Efficient nearest neighbor search for cross-encoder models using matrix factorization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2171–2194, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.140. URL https://aclanthology.org/2022.emnlp-main.140/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

A Experimental Setup and Data

A.1 Data and Revision Construction

Each controlled revision pairs an original query with a revised query over a fixed pool of 64 candidates, and the target reranker scores every candidate under both queries, so the precomputed score table has one score for every query-document pair. MultiConIR supplies queries with several explicit conditions in five domains, so a single condition can be added, removed, or negated. The three BEIR domains supply less structured queries whose candidates have more closely spaced scores. Table 3 gives the counts. MultiConIR condition edits add, remove, or negate a condition; BEIR uses seven deterministic edit types. A multi-edit set of 819 cases applies nine edit types to each of 91 source queries (intent rewrite, entity replacement, synonym paraphrase, temporal qualifier, surface paraphrase, numeric change, constraint removal, negation, and constraint addition). A session set of 91 four-query sessions $q_0 \to q_1 \to q_2 \to q_3$, built from successive condition additions, supports the multi-turn analyses and provides the 273 successive revisions used for the oracle. These controlled sets keep the candidate pool fixed or change it only through explicit insertions and deletions, whereas the TopiOCQA experiments retrieve a new pool on every turn.

Table 3: Controlled revision sets.

Source	Domains (cases)	Total
MultiConIR	Books (384), Legal (384), Medical (384), Movies (384), People (243)	1,779
BEIR	ArguAna (140), NFCorpus (140), SciFact (140)	420
Qwen3-Reranker-8B	all of the above	2,199
mxbai-rerank-large-v1	matched subset	1,380

For TopiOCQA, retrieval uses Anserini BM25 ($k_1 = 0.9$, $b = 0.4$) over the complete official Wikipedia passage corpus and its Lucene index, returning the top 64 passages for each turn independently. Official document identifiers map to corpus rows through the released passage index, and each turn's gold passage is matched to the corpus by identifier and text. The *answer-free question-history* representation concatenates the current user question with all previous user questions in the conversation, separated by a fixed delimiter, and excludes every answer, rationale, topic, future turn, and answer-derived rewrite. The *current-question* representation uses the current user question only. In each conversational run, retrieval and reranking use the same representation. The official all-history serialization, which includes previous gold answers, appears only in retrieval diagnostics.

Unless stated otherwise, $k = 10$, and ranks are formed by descending score with ties broken by document identifier. Exact ordered agreement requires the returned top-10 to equal the full-rerank top-10 position by position. Membership agreement requires the same ten documents. Fidelity nDCG@10 treats the full rerank as ground truth, with graded gains that decrease linearly with rank, so an exact match gives 1.0. For the final conversational run, we report reliability at both fidelity thresholds, 0.98 and 0.99.

A.2 Reranker and Inference Configuration

Qwen3-Reranker-8B runs in BF16 with SDPA attention and scores a pair by the model's relevance logit for its standard instruction-formatted prompt. mxbai-rerank-large-v1 runs in FP16 with its classification head and pinned tokenizer at a 512-token limit. Offline score tables record one scalar score per query-document pair. Replays reveal a stored score only when a policy requests it, so every counted evaluation corresponds to a score the policy would have paid for. Live runs score only the requested pairs. Full reranking scores all 64 candidates together, while a partial update scores a smaller, varying set; BF16 logits can change slightly with batch composition. The GPU hardware comprised four NVIDIA Quadro RTX 6000, two NVIDIA GeForce RTX 3090, and two NVIDIA RTX A6000 cards. Controlled GPU measurements used the Quadro RTX 6000 cards with CUDA synchronization around each reranker call. The final conversational evaluation ran the full retrieve-and-rerank loop on one RTX A6000, loading the model and index once per worker.

The stored state for each conversation records, for every scored candidate, its document identifier, its Qwen score, its rank within the pool, the query representation and model version that produced the score, and the turn at which it was computed. A state counts as a full ranking only when every candidate in its pool has a score from the same query. Partial states are used for output but never as the starting point of the next update.

A.3 Conversational Sample

Policy development used TopiOCQA development conversations and the controlled revision sets. These informed the choice of $m = 32$ but are not included in the final estimates. The held-out evaluation contains 250 TopiOCQA conversations; its sample size and stopping rule were fixed before inspection of results. Appendix E.2 describes its protocol.

B Minimum Recomputation and Ranking Structure

B.1 Minimum-Recomputation Oracle

The union of the old and new top-k lists the candidates involved in a ranking change. The minimum-recomputation set instead lists the scores that must be refreshed to reproduce the new ordered top-k while every other candidate keeps its old score. The two can differ, because mixing new and old scores can misorder members of the union, so rescoring the union is not always enough.

For a revision with retained set R, new candidates E, old scores s^o_d, full-rerank new scores s^n_d, and full-rerank ordered top-k $\pi_1, \ldots, \pi_k$, let $x_d \in \{0, 1\}$ indicate that $d \in R$ is refreshed, and define

$$\tilde{s}_d = x_d\, s^n_d + (1 - x_d)\, s^o_d \quad (d \in R), \qquad \tilde{s}_d = s^n_d \quad (d \in E).$$

The oracle solves

$$\min_x \sum_{d \in R} x_d \quad \text{s.t.} \quad \tilde{s}_{\pi_i} \geq \tilde{s}_{\pi_{i+1}} + \epsilon_{\pi_i \pi_{i+1}}, \;\; i < k, \qquad \tilde{s}_{\pi_k} \geq \tilde{s}_d + \epsilon_{\pi_k d}, \;\; d \notin \pi,$$

where ϵ_{ab} is zero when a wins a tie against b under the identifier rule and a small positive constant otherwise. Every constraint is linear in x. The membership variant keeps only the second family of constraints. All instances were solved to optimality.

On the 273 revisions of the session set, the ordered minimum averaged 2.65 evaluations (95% source-cluster bootstrap CI 2.31–3.01), with median 2, 90th percentile 6, 95th percentile 8, and maximum 11; 24.18% of revisions required no fresh evaluation, 84.98% required at most five, and 99.63% at most nine. The membership minimum averaged 0.30. When most candidates are new each turn, the oracle, given a fully scored previous turn, needs 3.43 retained rescores after all new candidates are scored (not counting the cost of scoring the previous turn). Table 4 gives the old ranks that most often appear in S^* on the 1,815 ranking-changing Qwen revisions. The oracle uses every new score, so we use it only to measure how much of a full rerank is redundant, and never as an online budget or safety test.

The union of old and new top-10 contains 10.57 ± 0.06 candidates for Qwen and 11.35 ± 0.11 for Mxbai. We also measured, for each range of old ranks, how often a candidate in that range enters

Table 4: Most frequent old ranks in the minimum-recomputation set (Qwen, 1,815 ranking-changing revisions, 95% CIs).

Old rank	$P(r \in S^*)$	Old rank	$P(r \in S^*)$	Old rank	$P(r \in S^*)$
8	$44.6 \pm 2.9\%$	6	$41.8 \pm 3.2\%$	11	$32.2 \pm 2.8\%$
7	$43.6 \pm 3.1\%$	5	$37.0 \pm 3.2\%$	3	$30.6 \pm 3.1\%$
9	$43.4 \pm 2.8\%$	4	$34.3 \pm 3.4\%$	2	$20.9 \pm 2.9\%$
10	$42.3 \pm 3.0\%$	12	$16.3 \pm 2.3\%$	1	$17.0 \pm 2.9\%$

the new top-10. A candidate at old ranks 11–20 enters in $5.09\% \pm 0.53$ of (revision, candidate) instances, one at ranks 21–50 in $0.18\% \pm 0.06$, and one at ranks 51–64 in $0.04\% \pm 0.05$. These are per-candidate rates within each range, not shares of all documents that enter the top-10, so they do not sum to 100%.

B.2 Rank-$(k+1)$ Analyses

Table 5 compares refreshing old ranks $1..k$ with refreshing $1..k{+}1$ across cutoffs and models. The single extra candidate produces a large gain at every cutoff, so the gain comes from the position of the cutoff rather than from rank 11 itself, and rank $k{+}1$ should be among the first candidates rescored. It does not guarantee that lower-ranked candidates stay out, which is what the expansion step and fallback handle.

Table 5: Exact top-k membership accuracy when refreshing old ranks $1..k$ versus $1..k{+}1$ on ranking-changing revisions (95% CIs; McNemar p).

k	Model	N	$1..k$	$1..k{+}1$	Gain	p
3	Qwen	870	50.5%	66.9%	$+16.4 \pm 3.6$	5×10^{-21}
3	Mxbai	679	16.9%	27.3%	$+10.3 \pm 2.9$	2×10^{-12}
5	Qwen	1,274	49.3%	65.0%	$+15.7 \pm 2.7$	7×10^{-37}
5	Mxbai	957	20.0%	29.6%	$+9.6 \pm 1.9$	6×10^{-18}
10	Qwen	1,815	54.4%	74.7%	$+20.3 \pm 2.3$	1×10^{-84}
10	Mxbai	1,181	26.9%	39.6%	$+12.7 \pm 2.0$	1×10^{-36}
20	Qwen	2,126	21.4%	31.1%	$+9.7 \pm 1.5$	3×10^{-47}
20	Mxbai	1,331	9.8%	16.0%	$+6.2 \pm 1.5$	6×10^{-17}

The repairs contributed by rank $k{+}1$ fall into three groups (Table 6). Its old score may place it inside the top-k above rescored competitors, so rescoring it removes a false entry; it may be the one lower-ranked candidate that crosses; or its new score may set the cutoff correctly and keep lower candidates out. At $k = 10$ the first group is the largest, consistent with the scale mismatch between old and new scores described in Section 5.

Table 6: Decomposition of repairs from adding rank $k{+}1$ (Qwen, 95% Wilson CIs).

k	Repairs	False entry removed	Sole crossing captured	Cutoff fixed
3	193	$24.9 \pm 6.6\%$	$62.2 \pm 6.6\%$	$13.0 \pm 5.5\%$
5	237	$40.1 \pm 6.3\%$	$45.1 \pm 6.4\%$	$14.8 \pm 5.1\%$
10	397	$55.4 \pm 4.8\%$	$34.8 \pm 4.8\%$	$9.8 \pm 3.3\%$
20	224	$17.0 \pm 5.4\%$	$60.3 \pm 6.2\%$	$22.8 \pm 5.9\%$

Across ranking-changing Qwen revisions, old ranks 1, 2, and 3 sit 8.73, 7.12, and 5.92 logits above rank 10 on average, but only 1.61, 1.21, and 0.94 logits above their next neighbor. They leave the top-10 in 0.17%, 0.39%, and 0.55% of instances, and at least one leaves in 0.88% of revisions. Reordering involving ranks 1–3 occurs in 47.9% of revisions, split between swaps among the top three (19.4%) and swaps with ranks 4–10 (28.5%).

C Mechanism Analyses

C.1 Shared Movement and Candidate-Specific Residuals

Table 7 summarizes the correlation structure of score changes on the full controlled sets. For both models, removing a per-revision affine fit from the strongly correlated raw changes leaves residuals with near-zero mean correlation. Qwen keeps a small positive residual correlation between adjacent ranks, which disappears when residuals are shuffled among candidates of similar rank. On raw changes the leading singular component explains 53% of variance; after affine removal it explains 14% and the first five components 35%.

Table 7: Correlation structure of score changes.

Quantity	Qwen	Mxbai
Mean pairwise correlation, raw changes	0.527	0.935
Mean pairwise correlation, affine residuals	-0.015	-0.016
Adjacent-rank residual correlation	0.110	0.065
Adjacent-rank residual correlation, shuffled control	-0.031	-0.035

For Qwen, the relative variance $\mathrm{Var}(\Delta_i - \Delta_j)/(\mathrm{Var}\Delta_i + \mathrm{Var}\Delta_j)$ is 0.29 for adjacent ranks, 0.32 at rank distance 5, 0.36 at distance 10, 0.40 at distance 20, and 0.41 across the pool, compared with 1 for independent changes. The shared component can be estimated from a few new scores: fitting the affine map on four rescored candidates and applying it to the old scores of the rest raised exact agreement from 35.7% to 38.2% at a fixed budget ($p = 1.7 \times 10^{-6}$). The shared part does not reveal the residual of a candidate that was not rescored, which is why the method rescores candidates rather than adjusting their old scores.

C.2 Effects of Revision Type and Document Content

Adding, removing, or negating a condition and rewriting the query each change scores in a distinct pattern. Table 8 lists the mean score change, its spread, and the rate at which a candidate enters the top-10 for each edit type in the multi-edit set. Shuffling edit-type labels among candidates with similar old rank and a similar number of candidates near the cutoff gives between-type variance of 0.00009, against 0.164 observed, and none of 500 shuffles reaches the observed value, so the differences remain after accounting for old rank and cutoff density. Each edit type's pattern of score changes is also similar across domains, with mean cross-domain correlation 0.95 for condition negation, 0.81 for temporal qualification, and 0.21–0.55 for most other edit types.

Table 8: Score response by edit type (Qwen multi-edit set; 95% source-cluster CIs).

Edit type	Mean change (logits)	Std.	Top-10 entry rate
Intent rewrite	$+0.92 \pm 0.10$	0.98	24.8%
Entity replacement	$+0.13 \pm 0.07$	0.61	18.2%
Synonym paraphrase	$+0.08 \pm 0.02$	0.23	4.2%
Temporal qualifier	$+0.07 \pm 0.05$	0.56	14.7%
Surface paraphrase	$+0.04 \pm 0.04$	0.43	5.1%
Numeric change	-0.00 ± 0.05	0.55	13.9%
Constraint removal	-0.11 ± 0.08	0.71	17.5%
Negation	-0.13 ± 0.09	1.05	28.4%
Constraint addition	-0.76 ± 0.05	0.88	31.2%

For the document interventions, we took 30 session cases in which a revision adds a condition, edited each satisfying document in three ways, and rescored with Mxbai. Removing the satisfying clause lowered the score by 0.64 ± 0.14 logits, removing an unrelated clause of equal length by 0.37 $\pm$ 0.18 ($t = 2.17$, $p = 0.038$), and replacing the satisfying clause with a contradiction by $1.36 \pm$ 0.15 ($t = 8.80$, $p = 1.1 \times 10^{-9}$ against the control). These interventions show that scores depend

on the document content the revision is about, though they do not reveal how the model computes this.

On the session set, a document's residual under one revision correlates with its residual under the next at 0.055, the residual at the same old rank at 0.078, and a within-source permutation control at -0.001. In out-of-fold prediction, adding rank history reduced error by 0.43% and adding the same document's history by 0.07%. A per-document sensitivity index fitted over 3,000 frequently retrieved documents reduced held-out error by 3.8% relative to predicting no change. Candidates with similar prior scores (within 0.2 logits) diverge in their changes by 0.40 logits, against 0.92 for candidates at least 2 logits apart.

C.3 Candidate Density Near the Cutoff, Crossing Risk, and Transfer

MultiConIR has 0.96 ± 0.16 candidates within one logit of the cutoff and a bimodality coefficient of 0.515 ± 0.005; BEIR has 8.58 ± 1.26 and 0.349 ± 0.026 ($p < 10^{-75}$ for both differences). The crossing model is $\operatorname{logit} P(\text{cross}) = -0.6123 - 2.9057\,\tilde{\Delta}_t(d)$, where

$$\tilde{\Delta}_t(d) = \frac{\Delta_t(d)}{\hat{\sigma}\big(1 + \ln(1 + \rho_k)\big)}, \qquad \Delta_t(d) = s_t(\pi_k) - s_t(d). \tag{1}$$

Here $\hat{\sigma}$ is the empirical standard deviation of the old scores at ranks $1, \ldots, k{+}1$, and $\rho_k = |\{d : |s_t(d) - s_t(\pi_k)| \le 1.0\}|$ counts candidates within one logit of the rank-k score. Because $\hat{\sigma}$ and ρ_k are computed once per revision, $\tilde{\Delta}_t(d)$ increases strictly with the old margin within that revision when $\hat{\sigma} > 0$, and the fitted crossing probability decreases strictly. This is the property Section 5 uses to turn a risk threshold into a rescoring depth. Table 9 reports transfer, and Table 10 and Figure 7 compare crossing rates with independent random changes drawn from a Gaussian with each revision's observed variance. The model describes where crossings tend to occur; it is not a guarantee for any single request.

Table 9: Transfer of the crossing model fitted on Qwen–MultiConIR.

Setting	Regime	Candidate pairs	ROC-AUC	Calibration r
Qwen–MultiConIR	fit	95,958	0.989	0.996
Qwen–BEIR	new dataset	22,788	0.907	0.996
Mxbai–MultiConIR	new model	51,840	0.941	0.993
Mxbai–BEIR	new model and dataset	22,680	0.837	0.988

Table 10: Rates of crossing the cutoff by initial margin.

Margin (logits)	Independent random	Qwen	Mxbai
$[0.0, 0.5)$	33.7%	23.5%	7.9%
$[0.5, 1.0)$	16.4%	4.9%	0.9%
$[1.0, 1.5)$	6.1%	0.70%	0.12%
$[1.5, 2.0)$	1.8%	0.08%	0.02%
$[2.0, 3.0)$	0.17%	0.01%	0.00%

D Additional Policies, Baselines, and Ablations

D.1 Controlled Policies and Proxies

Table 11 extends Table 1. The cutoff-margin gate rescores old ranks $1..k{+}1$ and then, in a single pass, every candidate whose old score plus the estimated average change lies within a fixed margin of the new cutoff. The controlled two-stage replay preserves pool order on ties, estimates movement from old ranks 4–11, scales $\lambda \in \{0.4, 0.8\}$ by the old-score interquartile range, and uses mixed stage-1 scores for the cutoff. The scale-invariant gate sets the margin to a quarter of the interquartile range of the old scores. The membership variant does not rescore old ranks 1–3.

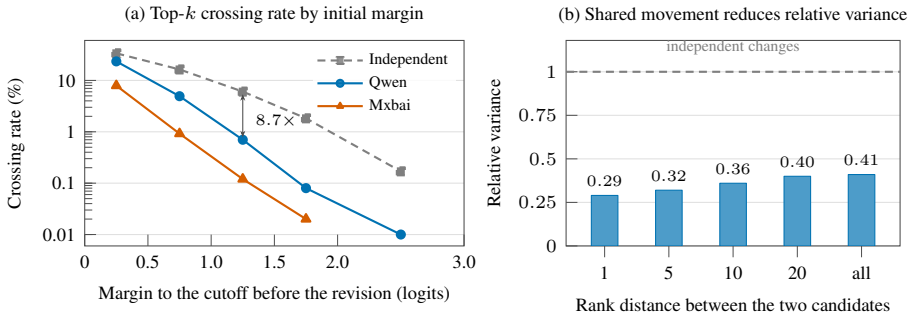

Figure 7: Real query revisions cross the top-k cutoff far less often than independent random score changes at the same margin. (a) Crossing rates by margin bin, plotted at bin midpoints on a log scale, with independent random changes dashed in grey; the arrow marks the $8.7\times$ gap for Qwen at 1.0–1.5 logits. The random changes are Gaussian with each revision's observed variance. (b) Whether two candidates swap depends on the difference between their score changes, $\Delta_i - \Delta_j$. For Qwen, correlated changes make that difference vary much less than it would for independent changes; bars show $\mathrm{Var}(\Delta_i - \Delta_j)/(\mathrm{Var}\,\Delta_i + \mathrm{Var}\,\Delta_j)$.

Table 11: Controlled policies on 1,815 ranking-changing Qwen revisions.

Policy	Eval.	Skipped	Membership	Exact order	Fid. nDCG@10
Boundary window, ranks 4–10	7.00	89.1%	54.3%	24.9%	0.953
Boundary window, ranks 4–11	8.00	87.5%	74.6%	37.1%	0.954
Boundary window, ranks 4–12	9.00	85.9%	85.7%	44.5%	0.954
Static window, ranks 1–11	11.00	82.8%	74.7%	73.4%	0.995
Cutoff-margin gate	14.05	78.1%	94.7%	94.4%	0.999
Scale-invariant gate	14.10	78.0%	95.0%	94.7%	0.999
Two-stage, $\lambda = 0.4$	12.61	80.3%	87.6%	86.9%	0.997
Two-stage, $\lambda = 0.8$	14.82	76.8%	95.1%	94.7%	0.999
Membership variant	9.68	84.9%	86.6%	44.3%	0.954

For conformal calibration, the nonconformity score of a query is the largest score gain of any candidate below the cutoff, relative to the gain of the rank-k candidate, $E_i = \max_{j>k}[(s^n_j - s^o_j) - (s^n_{(k)} - s^o_{(k)})]$, and the expansion margin $\hat{\lambda}$ is its $\lceil (n+1)(1-\epsilon) \rceil / n$ quantile over 889 calibration queries. On 890 held-out MultiConIR queries (Table 12), observed violation rates stayed below target at $\epsilon = 10\%$, 2%, and 1%, with 11.0 to 13.3 evaluations per query.

Table 12: Conformal calibration on held-out MultiConIR (95% source-cluster CIs).

Target ϵ	$\hat{\lambda}$ (logits)	Violation rate	Eval.	Fid. nDCG@10
10%	stage 1 only	$7.53 \pm 2.28\%$	11.00	0.99995
2%	2.07	$1.01 \pm 0.67\%$	12.25	0.99999
1%	3.40	$0.11 \pm 0.24\%$	13.33	1.00000
0.5%	4.06	$0.11 \pm 0.24\%$	13.91	1.00000

Greedy probing starts from old ranks 1, 10, and 11 and repeatedly rescores the remaining candidate whose old score is closest to the current cutoff estimate. At budgets of 3, 5, 8, 11, 14, and 20 it reaches nDCG@10 of 0.939, 0.939, 0.940, 0.948, 0.976, and 0.982, against 0.961, 0.980, 0.992, 0.995, 0.997, and 0.998 for a contiguous window of the same size. Probing by predicted crossing probability matches the contiguous window from 11 evaluations upward. Proxy gating runs a smaller reranker under both queries and rescores with Qwen any candidate whose change under the smaller model, relative to the cutoff, exceeds a threshold, plus old ranks $1..k{+}1$. On the multi-edit set, MiniLM-L6 at threshold 2.0 used 11.9 Qwen evaluations and Ettin at 1.5 used 11.5, each at

0.99996 nDCG@10 and about 89.5% exact agreement, in addition to the proxy's two passes over the pool. Few-probe reconstruction fits a five-dimensional basis of score changes on training revisions, chooses probes by column-pivoted QR, and reconstructs all 64 scores; with six probes it reaches 0.961 nDCG@10 and 24% exact agreement on 283 held-out revisions, against 0.544 for the six top-ranked candidates.

On SciFact (140 queries), differential reranking used 14.00 $\pm$ 1.06 evaluations, matched the exact top-10 in 36.4 $\pm$ 9.3% of queries, and reached 0.993 nDCG@10; on ArguAna (140), where the gap between ranks 10 and 11 averages 1.24 logits against 0.08 on SciFact, it used 11.74 $\pm$ 0.45 evaluations, matched in 89.3 $\pm$ 5.7%, and reached 0.9997.

In the 150 controlled cases with pool changes, 50 each of deletion, insertion, and mixed query and document change, differential reranking scores the new documents and rescores retained ones as before. It used 11.00, 12.00, and 11.16 evaluations and matched in 100%, 100%, and 82% of cases (94.0% overall; 11.39 evaluations on average), reaching fidelity nDCG@10 of 1.000 against 0.987 when all retained candidates keep their old scores ($p = 0.0015$). Over 91 three-turn Mxbai sessions it used 39.7 of 192 evaluations, matched in 85.3% of turns, and reached 0.984 nDCG@10, against 128 evaluations and 0.981 for a schedule that fully reranks turns 1 and 3. The number of new top-10 documents relative to each session's first query grew only from 1.38 at turn 1 to 1.54 at turn 3.

For the GPU measurements, the full-rerank and differential arms were timed separately for every query, and the speedup is the per-query ratio of reranking-stage time averaged over 91 session cases. Deciding which candidates to rescore took about 0.3 ms per query. Table 13 gives the cutoff sweep for Mxbai.

Table 13: Measured GPU reranking stage across cutoffs (Mxbai, 91 session cases, 95% CIs).

k	Eval.	Skipped	Speedup	Exact top-k	nDCG@k
3	8.86 [7.23, 10.49]	86.2%	11.65× [10.25, 13.04]	81.3%	0.966
5	10.79 [9.29, 12.29]	83.1%	8.63× [7.86, 9.40]	80.2%	0.974
10	15.71 [13.78, 17.65]	75.4%	6.05× [5.54, 6.55]	90.1%	0.992
20	35.33 [32.66, 38.00]	44.8%	2.15× [1.98, 2.32]	52.7%	0.998

D.2 Baseline Implementations

The *same-budget BM25 baseline* receives, on every turn, exactly as many Qwen evaluations as differential reranking used on that turn, spends them on the top current BM25 candidates, and returns the top-k of those scores. It does not use the previous ranking, so it measures what the same budget achieves without it.

The *Mxbai-guided selector* is a development-only baseline with the same Qwen budget. mxbai-rerank-large scores all 64 candidates with the current answer-free question-history query, and Qwen rescores Mxbai's highest-scoring candidates using the same per-turn Qwen budget used by $m = 32$. $m = 32$ reaches 0.73561 exact ordered top-10 and 0.999751 fidelity nDCG@10; Mxbai-selected Qwen reaches 0.59402 and 0.993807. The paired differences ($m = 32$ minus Mxbai) are +0.14158 exact order (95% conversation-cluster bootstrap CI [0.12010, 0.16389]) and +0.005943 nDCG (CI [0.00494, 0.00708]), with Holm-adjusted sign-flip $p = 0.000020$ for both primary metrics. Mxbai additionally requires a full auxiliary pass over the candidate pool before Qwen rescoring. This experiment compares selectors at the same Qwen cost on development data; it is separate from the held-out evaluation and does not establish equal total compute or the latency of a live two-model pipeline.

The *EARM-style adaptation* runs on the development data at a fixed per-conversation budget equal to differential reranking's total evaluations there. It reserves the full rerank on the first turn, spends the remaining budget on candidates in current BM25 order, and fills unscored entries by matrix completion over the conversation's score history, initialized as in the released online implementation. At that budget it reached 0.571 exact ordered agreement and 0.935 nDCG@10, against 0.599 and 0.986 for the same-budget BM25 baseline (-2.7 points exact, 95% CI -4.4 to -1.0). EARM was

designed for long streams of related queries against a persistent store, so this comparison concerns only how its reuse mechanism behaves under explicit revisions with a matched budget.

For *document-representation caching*, we measured how often exact document content recurs within a conversation's T5-rewrite retrieval stream, which bounds what query-independent caching of document encodings can save. On 24 held-out conversations, 25.1% of document tokens (95% CI 19.7–30.9%) had already appeared earlier in the same conversation, with almost no reuse across conversations. In a separate eight-conversation development timing study, caching BGE-M3 document vectors by exact content reduced the scoring-stage cost by 29.69% relative to re-encoding every top-64 document, while an optimistic offline precomputation of the documents accessed later in each conversation saved 33.38%. The latter knows future accesses and is not a serving policy, but it shows that ordinary document precomputation is a strong comparator. Such caching reduces the cost per evaluation and is complementary to differential reranking.

Temporal reciprocal-rank fusion maintains the candidate list itself, fusing the previous list with the current T5 top-64 at equal weight ($k = 60$). On 24 conversations it would admit about half as many new candidates as independent retrieval, at a slight cost in nDCG@10 against the gold passage (-0.019, 95% CI -0.041 to $+0.005$). We report it as a retrieval-side control and do not use it in the main pipeline.

D.3 Other Selection Signals

The following experiments test other signals for choosing which stored ranking to reuse, deciding how many candidates to rescore, and correcting old scores.

Choosing a stored ranking. Using different query paths over shared pools, BGE-M3 query similarity tracks the minimum number of candidates that must be rescored (Spearman -0.62, 95% CI -0.73 to -0.53). A router that picks the stored ranking with the most similar query, checks it by rescoring eight candidates, and otherwise reranks fully recovered the exact top-10 on 119 of 120 held-out revisions, saved 34% of evaluations, and added 7.7 ms. Learned routers that predict how many candidates to rescore from query, lexical, and old-ranking features met the fidelity target on at most 84% of held-out cases at their chosen budgets, below a constant budget chosen on training data. Query similarity thus helped choose a stored ranking but was a weak guide to how many candidates to rescore; across 546 session revisions, embedding distance between queries correlated with ranking change at only $r = 0.28$.

Candidate selection. Table 14 gives the agreement between each smaller (proxy) reranker's score changes and Qwen's on the multi-edit set. The best proxy, Qwen3-Reranker-4B, reaches $r = 0.49$, and the rest are below 0.25. A model hierarchy that predicts each candidate's change from rank and margin, then adds statistics about scores near the cutoff, average score change, edit-type labels, and polynomial interactions, raises in-domain AUC for predicting which candidates enter the top-10 from 0.839 to 0.992. The richest model, however, transfers with AUC 0.642 on Qwen–BEIR, 0.542 on Mxbai–MultiConIR, and 0.364 on Mxbai–BEIR, where the two-parameter crossing model continues to transfer well. On changing pools, BM25 rank is a useful guide among new candidates (those at BM25 ranks 1–10 reach the Qwen top-10 33.8% of the time, against 8.9% at ranks 51–64), but it gives no safe point at which to stop scoring.

Table 14: Proxy rerankers on the multi-edit set (95% source-cluster CIs).

Proxy	Pearson r of changes	New-candidate recall@10	Eval.
Qwen3-Reranker-4B	0.488 ± 0.031	$88.2 \pm 4.0\%$	27.1
Qwen3-Reranker-0.6B	0.238 ± 0.031	$83.9 \pm 4.5\%$	32.1
MiniLM-L6	0.136 ± 0.035	$85.3 \pm 4.2\%$	28.0
Ettin	0.113 ± 0.032	$81.5 \pm 4.6\%$	33.4
bge-reranker-large	0.090 ± 0.036	$83.2 \pm 4.5\%$	27.3

Correcting old scores. Fitting an affine map from old to new scores on rescored candidates and applying it to the rest improves fidelity at a fixed budget, for example from 0.9934 to 0.9955 nDCG@10

when rescoring the old top 56 on natural fixed-pool sessions. However, rank agreement between old and new scores on the rescored candidates did not predict whether a partial update would reach fidelity nDCG@10 ≥ 0.99 (AUC 0.49). The correction improves old scores on average but cannot tell which unscored candidate will cross.

Rescoring lower-ranked candidates. On natural fixed-pool sessions, rescoring the old top 32 plus the documents whose embeddings are nearest the new query raised exact agreement from 0.38 to 0.52, compared with adding the same number of candidates at fixed positions. Repeatedly rescoring any old-score candidate that appears in the top-10 of mixed old and new scores stops early on more than 90% of revisions but cannot catch candidates outside that top-10. An exact feasibility program shows that rescoring the old top 32 plus at most eight more candidates suffices for 93.7% of these revisions when the extra candidates are chosen with hindsight, with a mean of 2.9. The hindsight result leaves open how to identify those extra candidates before scoring them.

Candidate-pool overlap. Deeper retrieval raises overlap between turns (mean Jaccard 0.10 at depth 64, 0.26 at 256, 0.88 at 1,000 on 32 conversations), but keeping old scores for retained documents in a top-256 pool reproduced the full rerank's top-10 on at most 1 of 25 turns. Rescoring the 16 retained candidates with the highest old Qwen ranks, along with all new candidates, raised exact agreement to 0.90, and old Qwen rank chose retained candidates better than current BM25 rank (0.79). Adaptive budget rules and overlap-based fallback rules either fell back to a full rerank on most turns or accepted updates below fidelity nDCG@10 0.99. In these experiments, overlap supported accurate partial updates only when the top retained candidates were rescored.

Fallback rules. On the conversational development data, we tested three single-threshold rules for falling back to a full rerank: the fraction of new candidates, the old-score gap at the edge of the rescored block, and how close new scores at that edge come to the cutoff. Thresholds were set on one fold and tested on the other. The first and third rules each passed one fold with more than 12% projected savings, but no single threshold passed both folds, which is why the conversational policy uses a fixed depth.

D.4 Sensitivity to m and the Top-k Output

Table 15 reports an offline deterministic replay across the 250 held-out conversations, with m ranging from 8 to 64. The replay reads fixed scalar scores from an exhaustive score bank and selects the retained top-m from the preceding exhaustive ranking. The primary live serving policy instead selects from its online stateful cache and performs BF16 inference with policy-dependent batch sizes. Near-tied top-10 scores can swap order under small numerical differences. Thus the replay's exact-order, BM25-margin, and reliability values describe that replay protocol; at $m = 32$, all 250 conversations pass the replay's 0.98 threshold, while 249 pass in live serving. The primary policy remained fixed at $m = 32$; the other depths describe how compute savings and fidelity change with the prefix size. At $m = 24$, call savings increase to 20.2%, but 239 of 250 conversations meet the 0.99 all-turn threshold, compared with 245 of 250 at $m = 32$.

Table 15: Offline deterministic replay sensitivity to the rescoring depth m on 250 held-out conversations. The primary live evaluation used $m = 32$ without retuning. Brackets give 95% intervals for the replay advantage over equal-call BM25.

m	Savings	Exact order (macro)	Turns ≥ 0.98	Conversations ≥ 0.98	Turns ≥ 0.99	Conversations ≥ 0.99	Advantage vs. equal-call BM25 (pp)
8	31.5%	55.6%	76.0%	10.0% (25/250)	68.0%	5.2% (13/250)	+5.1 [4.0, 6.1]
16	25.9%	93.0%	99.9%	98.4% (246/250)	99.0%	89.6% (224/250)	+39.5 [38.1, 40.8]
24	20.2%	98.3%	100.0%	99.6% (249/250)	99.6%	95.6% (239/250)	+41.1 [39.9, 42.2]
32	14.9%	99.4%	100.0%	100.0% (250/250)	99.8%	98.0% (245/250)	+38.4 [37.3, 39.6]
40	10.1%	99.9%	100.0%	100.0% (250/250)	100.0%	99.6% (249/250)	+34.4 [33.2, 35.6]
48	5.7%	100.0%	100.0%	100.0% (250/250)	100.0%	100.0% (250/250)	+25.7 [24.4, 27.1]
64	0.0%	100.0%	100.0%	100.0% (250/250)	100.0%	100.0% (250/250)	0.0

The conversational policy returns only the top-k by new scores. To see what this leaves out, we merged the new scores with the old scores, from the last full ranking, of every candidate still in the pool that was not rescored. On the development data, across 451 partial $m = 32$ updates, no old-score candidate entered the merged top-10 and the returned top-10 was unchanged, but the merged ranking reproduced the full order of all 64 candidates on only one update. An adaptive policy could use a smaller m when the new scores show a wide margin at the cutoff.

E Retrieval and Conversational Experiments

E.1 Query-Form Study and Overlap Diagnostics

Section 7 and Figure 5 summarize how much of the pool carries over between turns under different retrieval setups. This appendix gives the underlying query-format study and overlap statistics.

Table 16 crosses the query format used for retrieval with the query format used for reranking, on four full-Wikipedia conversations. Each cell has 41 turns: four first turns and 37 later turns. Retrieving with the current question alone made 63.8 of 64 candidates new on an average later turn, whereas the question history reduced evaluations by 14.5%, counting first turns. A second study on the same conversations separately varies the query format used for the previous full ranking and for the current query, holding the question-history retrieval pool fixed and giving every turn a full previous ranking. With both in history format, the policy was exact on 37 of 37 later turns; with a raw previous query and a history current query, on 13; with a history previous query and a raw current query, on 10; and with both raw, on 13. All four settings used 1,644 of 2,368 evaluations on later turns. These four-conversation contrasts show the effect of query format; they do not estimate performance at scale.

Table 16: Query format for retrieval versus for reranking ($m = 32$; four full-Wikipedia conversations).

Retrieval	Reranker	New cand.	Eval.	Savings	Exact	Same-budget BM25
Raw current question	Raw current question	63.8	2,624/2,624	0%	37/37	37/37
Raw current question	History	63.8	2,624/2,624	0%	37/37	37/37
History	Raw current question	14.3	2,244/2,624	14.5%	25/37	21/37
History	History	14.3	2,244/2,624	14.5%	37/37	22/37

In the ten-conversation rationale-passage study, each new pool contains 52.99 new candidates on average, of which 20.92 returned after an absence and 32.07 had never been seen earlier in the conversation. The new top-10 of the full rerank contains 2.12 documents from the previous pool, 3.34 returning documents, and 4.54 never-seen documents. In a four-conversation full-Wikipedia sample with question-history retrieval, the corresponding figures are 8.43, 0.31, and 1.25, and every retained document in the new top-10 had ranked within the top 32 old scores among retained candidates. Among retained candidates, the average Kendall τ between old and new scores is 0.754 in the Wikipedia sample and 0.346 in the rationale study, and after subtracting the median change, 83.8% and 38.4% of retained candidates' changes lie within one logit. In the Wikipedia sample, reusing only the previous turn's scores saved 34.3% of evaluations and reusing all earlier scores in the conversation saved 35.9%, so almost all reuse came from the previous turn.

E.2 Conversational Protocol and Timing

The final evaluation processed each of 250 conversations from its first turn. The first turn was a full rerank, which became the first stored full ranking. On each later turn, the policy scored every new candidate and the retained candidates in the top 32 of the last full ranking, returned the top-10 by new scores, and reranked fully if this would have covered the whole pool. After a partial update, the next turn was a full rerank, which became the new stored full ranking. Evaluation counts include every Qwen evaluation: first turns, new candidates, retained rescores, and full reranks.

The same code ran full Qwen reranking, differential reranking with $m = 32$, and the same-budget BM25 baseline on identical turns and pools. Differential reranking reduced mean end-to-end latency by 14.40% (95% CI 13.84–14.93%) and had a 95th-percentile latency ratio of 0.886 (0.866–0.903). Against the same-budget BM25 baseline, it gained 1.39 points in fidelity nDCG@10 (1.26–1.53) and 13.16 points in conversation-averaged exact ordered top-10 on warm turns (74.15% versus 60.99%).

For live exact-order agreement, we first average warm-turn matches within each conversation, then average the 250 conversation rates. The comparison uses each policy's ranking and full reranking as they were served, including any BF16 score variation caused by different batch composition. Near-tied documents can therefore swap adjacent positions even when the selected top-10 documents are unchanged. In a same-selected-set diagnostic that substitutes the exhaustive reference logits, differential reranking's exact-order rate rises to 99.41%. Its fidelity nDCG@10 advantage over equal-call BM25 is 1.41 points in that diagnostic, compared with 1.39 points in live serving. The diagnostic isolates the effect of scoring the selected set with common reference logits; its exact-order rate is not a live serving result.

Per repetition, full reranking required 186,688 Qwen evaluations across 2,917 turns. Differential reranking and the equal-call BM25 control each used 158,792, saving 27,896 evaluations (14.94%). Of the 2,667 later turns, 1,227 used partial updates, 1,094 were scheduled full resets, and 346 were natural full covers; 250 first turns were cold-start full reranks. Thus 1,690 turns used full rescoring and 1,227 used partial updates. The partial updates used 50,632 Qwen calls, including 11,368 calls for new candidates and 39,264 retained-candidate rescores. Relative to 78,528 calls for full rescoring on those turns, this saved 27,896 calls (35.52%). Across all later turns, 97,096 calls scored new candidates and 45,696 rescored retained candidates, for 142,792 calls versus 170,688 with full reranking.

At fidelity nDCG@10 ≥ 0.98, 249 of 250 conversations passed on every later turn; its one-sided exact 95% lower bound is 98.12%, and the six-comparison adjusted bound is 97.29%. At the stricter 0.99 threshold, 245 passed. Its one-sided exact 95% lower bound is 95.84%, above the 95% target, while the six-comparison adjusted bound is 94.73%. Each of the five conversations below 0.99 failed on a single partial turn with one full-rerank top-10 document left outside the rescored block, and the scheduled full rerank on the next turn returned fidelity to 1.0.

The old/new score merge diagnostic of Appendix D.4 is a post hoc analysis and does not affect the returned ranking or the policy. On development data, merging unrefreshed candidates with their latest stored score rarely admitted an unrefreshed candidate into the top-10. The returned conversational policy instead ranks only candidates scored under the current query.

E.3 Where the Method Applies

A retrieval-only diagnostic on 32 conversations shows that the query format affects both retrieval recall and how many candidates carry over between turns. The current question alone replaces almost the whole pool each turn, the answer-free question history keeps many more candidates, and query rewrites from a released T5-QReCC model (Anantha et al., 2021; Elgohary et al., 2019) retrieve the gold passage in the top 64 more often than the question history while overlapping much less between consecutive pools, so better retrieval and higher overlap can pull in different directions. We use this diagnostic only to show where differential reranking applies. The final conversational evaluation held retrieval fixed and measured fidelity to full Qwen reranking on the retrieved pool; downstream answer generation is outside its scope.

F Statistical Procedures

All reported intervals are 95%. The resampling unit is the source query for the controlled sets and the conversation for conversational runs, so turns within a conversation are never treated as independent. Diagnostic summaries use cluster bootstrap intervals (20,000 resamples unless stated otherwise; seed 20260922). Conversation-level pass rates use one-sided exact Clopper–Pearson lower bounds. We also computed Bonferroni-adjusted lower bounds for six planned comparisons; the 0.99 conversation-reliability bound falls below 95% under this stricter check. Paired fixed-pool binary comparisons use McNemar tests, and conversational comparisons use conversation-

level paired tests with sign-flip checks where reported. Latency changes are computed from per-conversation totals including first turns.

Table 17 summarizes the role each body of evidence plays in the paper.

Table 17: Evidence used in the paper and its role.

Evidence	Scale	Role
Controlled revision sets	2,199 Qwen, 1,380 Mxbai	Oracle, mechanism, controlled policies
Query-format studies	4 conversations each	Mechanism diagnostics
Development data	TopiOCQA development data	Policy selection, sensitivity to m, auxiliary baselines
Final held-out evaluation	250 conversations	Primary conversational fidelity, compute, and end-to-end latency
Same-budget Mxbai selector	Development-only	Cheaper-reranker selector diagnostic
EARM-style adaptation	Development-only	Historical-score reuse comparison

Every completed score table was checked for exactly one score per query-document pair and for agreement between the document order used in scoring and in analysis. The final aggregate counts were reconstructed from per-turn rows, including call counts, new-candidate sets, full-rerank flags, update types, timing components, and the budget assigned to the same-budget BM25 baseline. All fidelity nDCG values use the rank-linear gains described in Appendix A.1, and the Mxbai-guided development control uses new Qwen scores for its selected candidates.