LLM-as-MMR: Stepwise Diversification for Redundancy-Aware Document Reranking

Anonymous ACL submission

Abstract

Open-domain QA systems often select a small evidence set from a long retrieval pool. Existing rerankers usually focus on individual passage relevance, which can produce evidence sets that are relevant but redundant. This is especially limiting for ambiguous and multi-answer QA, where selected evidence should cover distinct answer facets. We propose *LLM-as-MMR*, an LLM-based reranking method that extends Maximal Marginal Relevance by replacing fixed similarity-based redundancy control with LLM-based assessment of whether a candidate passage adds answer evidence beyond the current selected set. LLM-as-MMR constructs answer-covering evidence sets through scalable set refinement, and is trained with supervised trajectories and further aligned with Group Relative Policy Optimization (GRPO). We also apply latent reasoning compression after RL training to reduce the cost of explicit reasoning. Experiments on ambiguous and multi-answer QA benchmarks show that *LLM-as-MMR* improves answer coverage in coverage-sensitive settings while reducing evidence redundancy.

1 Introduction

Retrieval-augmented generation and open-domain question answering systems often retrieve a long list of candidate passages but can only use a small evidence set for answering (Lewis et al., 2021; Karpukhin et al., 2020). For single-answer questions, ranking passages by relevance is often a reasonable proxy for evidence selection. For ambiguous and multi-answer questions, however, the selected evidence must cover distinct answer facets. A top-k set can be highly relevant but still incomplete if it repeatedly supports the same interpretation or answer.

This creates a gap between relevance-oriented reranking and answer-covering evidence selection. Bi-encoders, cross-encoders, and many LLM-based reranking methods primarily estimate query–document relevance (Nogueira and Cho, 2019; Ma et al., 2023; Pradeep et al., 2023a,b). Classical diversification methods such as Maximal Marginal Relevance (MMR) reduce redundancy by penalizing similarity to already selected documents (Carbonell and Goldstein, 1998). However, similarity is only an indirect signal of answer redundancy: similar passages may either repeat the same answer or discuss different valid facets of the same ambiguous query. A fixed lexical or embedding-based penalty cannot reliably distinguish these cases.

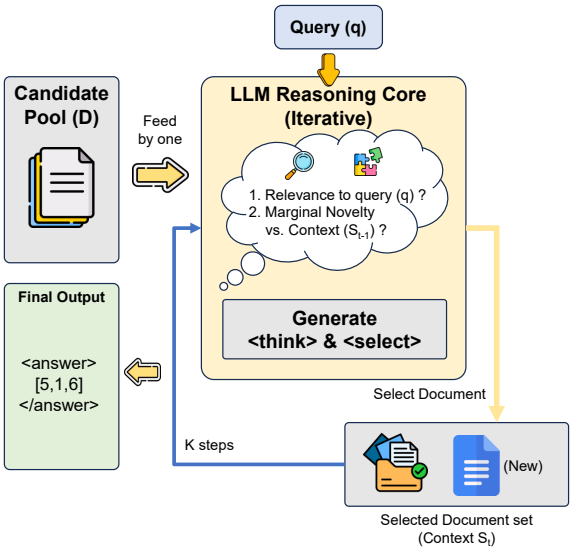

Figure 1: LLM-as-MMR for answer-covering reranking. Relevance-only ranking may allocate the top-k budget to redundant passages, while LLM-as-MMR selects evidence by assessing whether each candidate adds answer evidence beyond the current selected set.

We propose *LLM-as-MMR*, an LLM-based reranking method that extends MMR from fixed similarity-based redundancy control to LLM-based assessment of additional answer evidence. Instead of scoring each passage independently, the model evaluates a candidate with respect to the current selected evidence set. A passage that repeats an

already covered answer should receive low value even if it is relevant, while a passage that covers an underrepresented answer facet should be selected even if its standalone relevance score is not the highest. In this way, reranking is reframed from independent relevance ordering into evidence set construction under a fixed budget.

To scale LLM-as-MMR to candidate lists of arbitrary length, we instantiate it as *sliding-window set refinement*. The model maintains a fixed-size evidence set and scans the candidate list in windows, updating the selected set when newly observed passages add answer evidence beyond the current set. This design supports top-k evidence selection over long retrieval pools while allowing the reranker to revise earlier selections as more candidates become available.

We train and refine the model in three stages. First, we construct supervised set-refinement trajectories from AmbigQA (Min et al., 2020) and QAMPARI (Amouyal et al., 2023) with LLM-assisted trajectory generation, teaching the model how to select complementary evidence, skip redundant passages, and update the evidence set when later candidates improve answer coverage. Second, Group Relative Policy Optimization (GRPO) further aligns the model toward valid and coverage-oriented selection behavior (DeepSeek-AI et al., 2025). Third, after RL training, we introduce CODI-style latent reasoning compression (Shen et al., 2025), inspired by recent work on reasoning in latent space (Hao et al., 2025), to shorten the model's reasoning process and reduce the cost of explicit rationale generation while preserving the same evidence-selection objective.

We evaluate LLM-as-MMR on ambiguous and multi-answer QA reranking benchmarks. The evaluation asks whether the selected evidence covers more answer facets under the same evidence budget, with ranking metrics reported as complementary indicators. Experiments show that LLM-as-MMR improves answer coverage in coverage-sensitive settings, demonstrating the benefit of modeling reranking as answer-covering evidence set refinement rather than independent relevance ordering.

Contributions. Our contributions are:

- We propose LLM-as-MMR, an LLM-based formulation of redundancy-aware reranking that selects evidence by assessing whether a candidate passage adds answer information beyond the current selected set.
- We instantiate this formulation as sliding-window set refinement, enabling scalable top-k evidence selection over candidate lists of arbitrary length.
- We train the model with LLM-assisted supervised set-refinement trajectories and GRPO, and further incorporate latent reasoning compression to reduce inference cost; experiments on ambiguous and multi-answer QA benchmarks verify its effectiveness in answer coverage and redundancy control.

2 Related Work

Relevance-based and LLM reranking. Document reranking is commonly formulated as relevance estimation, where retrieved passages are ordered by their usefulness to a query. Cross-encoder rerankers based on pretrained language models provide strong query–document relevance signals (Nogueira and Cho, 2019), and late-interaction models improve the efficiency–effectiveness trade-off for passage ranking (Khattab and Zaharia, 2020). Recent LLM-based rerankers further perform listwise permutation, ranking generation, or passage-level scoring (Ma et al., 2023; Pradeep et al., 2023a,b; Liu et al., 2025). These methods can compare multiple passages, but their output is still primarily a relevance-oriented ordering. Our work instead targets evidence-set construction, where the model maintains a fixed-size selected set and decides whether later candidates add answer evidence beyond it.

Diversification-aware reranking. Search result diversification addresses the limitation of relevance-only ranking when a query has multiple intents, subtopics, or valid answers. Maximal Marginal Relevance (MMR) is a classical greedy criterion that balances query relevance with novelty relative to already selected items (Carbonell and Goldstein, 1998). Later work studies diversification through subtopic coverage, novelty, and explicit diversity objectives (Zhai et al., 2003; Clarke et al., 2008; Agrawal et al., 2009; Gollapudi and Sharma, 2009; Santos et al., 2010). LLM-as-MMR follows this set-construction view of reranking, but replaces fixed similarity-based novelty with LLM-based assessment of answer coverage. In open-domain QA, redundancy is not merely document similarity; it

is the failure to add answer evidence beyond the current selected set.

3 Method

LLM-as-MMR formulates reranking as evidence-set refinement under a fixed top-k budget. Given a retrieved candidate list, the model maintains a selected evidence set and decides whether newly observed passages add answer evidence beyond what has already been selected. This section first defines answer coverage as the set-level objective, and then introduces the sliding-window refinement process used to apply LLM-as-MMR to long candidate lists.

3.1 Problem Formulation

Let q be a query and let $\mathcal{D} = (d_1, \ldots, d_n)$ be the candidate list returned by a first-stage retriever. Given an evidence budget k, the reranker outputs a selected evidence list $\mathcal{S}_k = (s_1, \ldots, s_k)$, where each $s_i \in \mathcal{D}$ and no candidate is selected more than once. We write $\mathcal{S}_k$ as an ordered list to remain compatible with ranking metrics, while the coverage objective depends only on the passages included in the list.

For datasets with a gold answer set $\mathcal{A}(q) = \{a_1, \ldots, a_m\}$, we define answer coverage as

$$\mathrm{Cov}(\mathcal{S}_k, q) = \frac{1}{m} \sum_{j=1}^{m} \mathbb{I}\left[\mathrm{match}\left(a_j, \mathrm{text}(\mathcal{S}_k)\right)\right], \tag{1}$$

where $\mathrm{text}(\mathcal{S}_k)$ denotes the concatenation of the selected passages, and match denotes normalized answer matching. This coverage score is used to construct supervision, define RL rewards, and evaluate selected evidence sets. At inference time, the reranker observes only the query, candidate passages, and budget.

We further define the incremental answer coverage of a candidate d given a selected set $\mathcal{S}$:

$$\Delta(d \mid \mathcal{S}, q) = \mathrm{Cov}(\mathcal{S} \cup \{d\}, q) - \mathrm{Cov}(\mathcal{S}, q). \tag{2}$$

A candidate with $\Delta(d \mid \mathcal{S}, q) = 0$ may still be relevant to the query, but it does not add a new gold answer under the current evidence set.

The coverage objective differs from pointwise relevance ranking. A relevant document can be redundant if its answer is already covered, while semantically similar documents can both be useful when they support different answer facets. Classical MMR (Carbonell and Goldstein, 1998) captures a related relevance–novelty trade-off through greedy set construction:

$$d_t = \arg\max_{d \in \mathcal{D} \setminus \mathcal{S}_{t-1}} \Big[\lambda \, \mathrm{Rel}(q, d) - (1-\lambda) \max_{d' \in \mathcal{S}_{t-1}} \mathrm{Sim}(d, d') \Big]. \tag{3}$$

LLM-as-MMR keeps this set-construction perspective, but does not instantiate novelty as a fixed lexical or embedding-based similarity penalty. Instead, it uses an LLM to assess whether newly observed passages add answer evidence not covered by the current selected set. This changes the redundancy criterion from document-level similarity to answer-level coverage.

3.2 Sliding-Window LLM-as-MMR

To apply LLM-as-MMR to candidate lists of arbitrary length, we use sliding-window set refinement. The model maintains a fixed-size evidence set throughout reranking: it first initializes the set from the first window, and then updates it as later candidates are observed.

We partition the candidate list $\mathcal{D} = (d_1, \ldots, d_n)$ into contiguous windows $\mathcal{W}_1, \ldots, \mathcal{W}_T$ of size w:

$$\mathcal{W}_t = (d_{(t-1)w+1}, \ldots, d_{\min(tw,n)}). \tag{4}$$

Unless otherwise specified, our experiments use $n = 100$ and $w = 20$.

The first window initializes the evidence set:

$$\mathcal{S}_k^{(1)} \sim \pi_\theta(\cdot \mid q, \mathcal{W}_1, k). \tag{5}$$

The model emits a structured `<select>` action containing exactly k document indices from $\mathcal{W}_1$.

For each later window $\mathcal{W}_t$ with $t > 1$, the model receives the query q, the current evidence set $\mathcal{S}_k^{(t-1)}$, and the new candidates in $\mathcal{W}_t$. It emits either a skip action or a replacement action:

$$a_t \in \{\mathrm{Skip}, \mathrm{Rep}(d_t^+, d_t^-)\}, \tag{6}$$

where $d_t^+ \in \mathcal{W}_t$ is the newly selected document and $d_t^- \in \mathcal{S}_k^{(t-1)}$ is the document to remove. In the serialized output, these actions are written as `<skip>` and `<replace>` $d_t^+ \rightarrow d_t^-$ `</replace>`.

The evidence set is updated by

$$\mathcal{S}_k^{(t)} = \begin{cases} \mathcal{S}_k^{(t-1)}, & a_t = \mathrm{Skip}, \\ \mathcal{S}_k^{(t-1)} \setminus \{d_t^-\} \cup \{d_t^+\}, & a_t = \mathrm{Rep}. \end{cases} \tag{7}$$

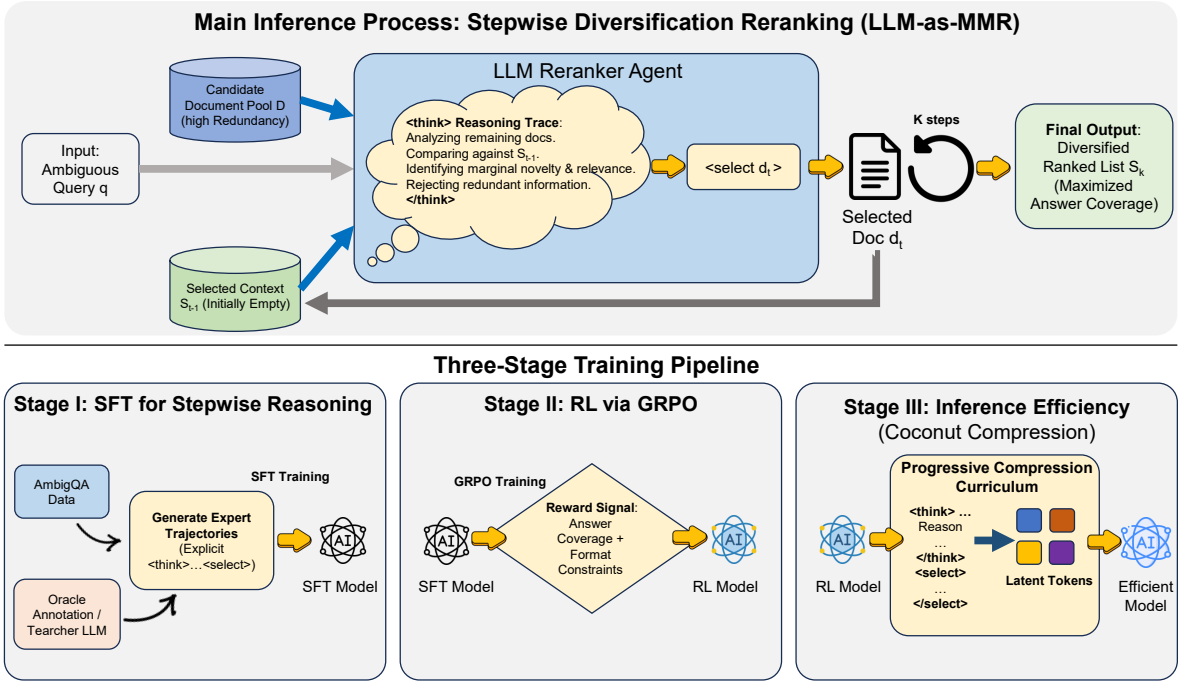

Figure 2: Overview of LLM-as-MMR. A first-stage retriever returns a long candidate list. The reranker maintains a fixed-size selected set, initializes it from the first window, and then scans later windows with skip or replacement actions. Supervised trajectories teach stepwise diversification; GRPO optimizes validity and answer coverage; latent reasoning compression reduces the cost of visible intermediate reasoning.

This update preserves the invariant $|\mathcal{S}_k^{(t)}| = k$. After all windows are processed, the final evidence set is $\mathcal{S}_k^{(T)}$.

This constrained action space keeps the top-k budget fixed and makes each later decision depend on the relation between the new candidates and the current evidence set. It also gives the model a structured output format, making invalid actions easy to detect during supervision and RL training.

3.3 Supervised Set-Refinement Trajectories

We first train the model to follow the sliding-window action format using supervised set-refinement trajectories. These trajectories are constructed offline from QA data with gold answer annotations. In our main data construction pipeline, queries come from AmbigQA (Min et al., 2020) and QAMPARI (Amouyal et al., 2023). For each query, we retrieve a top-100 candidate pool with Pyserini BM25 (Lin et al., 2021) over the Wikipedia prebuilt index `wikipedia-dpr-100w`, using the same retrieval setting for training and evaluation. We generate trajectories for budgets $k \in \{3, 5, 10\}$. Gold answers are used only for offline annotation, trajectory construction, reward computation, and evaluation; at inference time, the reranker observes only the query, candidate passages, and the evidence budget.

Trajectory construction has two stages. First, we use DeepSeek-V4-Pro (DeepSeek-AI, 2026) as an offline teacher model to annotate candidate passages with respect to the query and gold answers. For each passage, the teacher identifies whether it directly covers a gold answer, partially supports an answer, provides useful supplementary evidence, or is only weakly related or irrelevant. These annotations provide evidence signals for trajectory construction, but neither gold answers nor oracle labels are included in the student model input.

Second, the teacher generates sliding-window refinement trajectories consistent with the action space in Section 3.2. For the first window, it selects k passages that jointly cover as many distinct answers as possible and outputs the initial `<answer>` set. For each later window, the teacher compares the new candidates with the current selected set and decides whether the set should be kept or updated. A candidate is skipped if it is irrelevant, weakly useful, or redundant with already covered evidence. A replacement is generated if the candidate adds a missing answer facet, provides stronger support for a weakly covered answer, or improves the selected evidence without reducing answer coverage. The updated `<answer>` field is then used as the selected

set for the next window.

Each supervised example contains a short explanation of the comparison, the structured action sequence for the current window, and the updated selected indices. The explanation teaches the model how the new candidates relate to the current selected set, including whether they add uncovered answer evidence or repeat already covered evidence; it is not evaluated as a separate target. We retain trajectories with valid structured outputs and coverage-consistent updates. The resulting data teaches behaviors that independent relevance labels cannot express: allocating a fixed budget across answer facets, skipping relevant but redundant passages, and revising earlier selections when later candidates provide stronger or more complementary evidence.

3.4 Coverage-Oriented GRPO

After supervised fine-tuning, the model can follow the sliding-window action format and perform basic set refinement. We further apply Group Relative Policy Optimization (GRPO) (DeepSeek-AI et al., 2025) to strengthen output validity and answer coverage under model sampling.

RL training data construction. We construct RL training data by running the SFT model offline on the same BM25 top-100 candidate pools used for reranking. For each query, the SFT model executes the sliding-window procedure and produces intermediate window states. Each state contains the query, the current selected set before the decision, the candidate window, the evidence budget k, and the gold answers used only for reward computation. These window-level states are not directly available from the original QA data, because the input to each later window depends on the selected set produced by previous steps. We therefore collect them from SFT rollouts and serialize them as RL training prompts.

Reward design. The reward combines format validity and answer coverage. Let $\mathcal{C}(y)$ denote the active reward components for output y: it always contains $\{\mathrm{tag}, \mathrm{count}, \mathrm{final}\}$, and additionally contains delta when y includes a replacement action. The normalized score is

$$V(y) = \frac{\sum_{c \in \mathcal{C}(y)} w_c R_c(y)}{\sum_{c \in \mathcal{C}(y)} w_c}, \tag{8}$$

and the final reward is

$$R(y) = 2V(y) - 1. \tag{9}$$

Here R_{tag} checks whether the output follows the required XML-like schema, and R_{count} checks whether the selected indices are parseable, valid, unique, and satisfy the budget k. R_{final} is the answer coverage of the updated selected set. Coverage is computed over gold answers reachable from the retrieved candidate pool; if no gold answer is reachable, we assign a neutral coverage score.

For outputs containing a replacement action, the local coverage-gain term is

$$\begin{aligned} R_{\mathrm{delta}} = \max\big(&\mathrm{Cov}(\mathcal{S}_k^{(t)}, q) \\ &- \mathrm{Cov}(\mathcal{S}_k^{(t-1)}, q), 0\big). \end{aligned} \tag{10}$$

This term rewards a replacement only when it improves coverage over the pre-window selected set. Thus, a well-formed swap between redundant passages receives no local gain. In the main configuration, we set $(w_{\mathrm{tag}}, w_{\mathrm{count}}, w_{\mathrm{final}}, w_{\mathrm{delta}}) = (0.10, 0.10, 0.60, 0.20)$.

3.5 Latent Reasoning Compression

Visible intermediate reasoning helps the model compare new candidates against the current selected set and learn skip or replacement decisions under a fixed evidence budget. However, generating this reasoning text increases decoding cost, since each window requires additional tokens before the updated selected set is produced. To reduce this cost, we apply CODI (Shen et al., 2025) after GRPO training. CODI does not change the reranking objective, evidence budget, action space, or evaluation metrics. It is used as a compressed variant of LLM-as-MMR that replaces visible intermediate reasoning with continuous latent computation.

We train CODI from rollouts of the GRPO-trained LLM-as-MMR model. For each sliding-window prompt, the explicit model produces a visible reasoning trace, a structured action, and the final `<answer>` field. These traces provide the reference reasoning path for compression. In the compressed path, the model observes the same prompt but does not decode the visible reasoning text. Instead, it performs a fixed number of continuous latent transitions and then decodes the final structured `<answer>`.

The training objective combines final-answer generation with hidden-state distillation from the full-reasoning reference path. The reference path is conditioned on the visible reasoning trace, while the compressed path replaces that trace with latent

states produced by the CODI model itself. The distillation signal aligns the compressed representation with the reference representation near the answer boundary, encouraging CODI to preserve the selection behavior of the explicit GRPO policy without generating the full intermediate reasoning. We use eight latent iterations in the main configuration.

4 Experiments

We evaluate whether LLM-as-MMR improves answer coverage under a fixed top-k evidence budget. The experiments address four questions: whether set-refinement reranking improves answer-covering evidence selection, how the GRPO reward design affects the learned policy, whether the method benefits from longer candidate pools, and how much visible generation cost can be reduced by latent reasoning compression.

4.1 Experimental Setup

Datasets. Training data is constructed from AmbigQA (Min et al., 2020) and QAMPARI (Amouyal et al., 2023). We evaluate on AmbigQA, QAMPARI, and QUEST (Malaviya et al., 2023). AmbigQA typically contains a compact answer space, with about two to three answers per query, while QAMPARI requires selecting evidence for more answer facets. To study how candidate-pool redundancy affects reranking, we stratify evaluation examples by the redundancy of their retrieved passages. Specifically, we encode the top-100 passages using `Qwen3-Embedding-8B` (Zhang et al., 2025), compute the average pairwise cosine similarity among retrieved passages, and split examples into low-, medium-, and high-redundancy groups by tertiles. Higher redundancy indicates that the retrieved pool contains more semantically overlapping passages, making fixed-budget evidence selection more likely to suffer from repeated evidence. In our evaluation pools, QAMPARI has higher average top-100 redundancy than AmbigQA, making it a more direct test of answer-covering selection under redundant retrieval.

Retrieval and budgets. For each query, we retrieve passages with Pyserini BM25 (Lin et al., 2021) from the Wikipedia prebuilt index `wikipedia-dpr-100w`. All methods rerank the same BM25 candidate pool. Unless otherwise specified, the main experiment uses top-100 candidates. LLM-as-MMR uses window size $w = 20$, and the main tables report results under evidence budget $k = 5$.

Metrics. We use answer coverage, Cov@k, as the primary metric, following Eq. 1. Coverage is computed with strict automatic answer matching, which provides a reproducible and conservative measure of answer support under the same criterion for all methods. We also report NDCG@k (Järvelin and Kekäläinen, 2002), using binary relevance: a passage is relevant if it contains at least one gold answer.

Compared methods. We compare LLM-as-MMR with BM25, embedding similarity, MMR, a neural reranker, and an open-source listwise LLM reranker. BM25 keeps the original retrieval order. Embedding similarity and MMR use `Qwen3-Embedding-8B`, with MMR using cosine similarity as the redundancy term. The neural reranker baseline uses `Qwen3-Reranker-8B` (Zhang et al., 2025). We run RankZephyr (Pradeep et al., 2023b) as the listwise reranker baseline, using its zero-shot relevance-oriented listwise reranking setting. All reported baselines use the same candidate pools, redundancy splits, budgets, and evaluation metrics.

Implementation. LLM-as-MMR is initialized from `Qwen3-4B-Base`. We report the supervised model, the GRPO-trained model, and the CODI-compressed model (Shen et al., 2025). The SFT stage uses 14,958 supervised set-refinement examples. GRPO uses 13,500 training examples, and CODI is trained from 13,500 GRPO rollout trajectories. Additional training and decoding details are provided in Appendix C.

4.2 Main Results

Table 1 reports the main results over top-100 candidate pools, grouped by dataset and candidate-pool redundancy level.

The results show that the strongest method depends on the answer structure and redundancy of the retrieved pool. On AmbigQA, the pointwise reranker is strongest on most metrics. This is expected: AmbigQA typically has only two to three answers per query, and its top-100 candidate pools are less redundant than QAMPARI on average. In this setting, accurate relevance scoring can already recover much of the needed evidence, leaving less room for set-level diversification. Even so, LLM-as-MMR remains close to the strongest

Table 1: Main reranking results over top-100 candidate pools. Results are grouped by dataset and candidate-pool redundancy tier. We report Cov@5 and NDCG@5. Best results among displayed methods are in bold and second-best results are underlined. Abbreviations: Emb = Qwen3-Embedding-8B, Rerank = Qwen3-Reranker-8B, CODI = CODI-compressed LLM-as-MMR.

Method	AmbigQA						QAMPARI						QUEST					
	Low		Med.		High		Low		Med.		High		Low		Med.		High	
	Cov	NDCG	Cov	NDCG	Cov	NDCG	Cov	NDCG	Cov	NDCG	Cov	NDCG	Cov	NDCG	Cov	NDCG	Cov	NDCG
Baselines																		
BM25	0.270	0.301	0.294	0.330	0.274	0.307	0.192	0.300	0.179	0.306	0.188	0.424	0.029	0.103	0.047	0.154	0.053	0.180
Emb	**0.395**	0.541	0.465	0.588	0.430	0.582	0.220	0.335	0.219	0.386	0.203	0.496	0.045	0.161	0.061	0.205	0.083	0.329
MMR	0.331	0.352	0.390	0.402	0.359	0.388	0.150	0.185	0.160	0.242	0.177	0.373	0.029	0.083	0.048	0.130	0.082	0.240
Rerank	0.393	**0.567**	**0.466**	**0.625**	**0.440**	**0.621**	0.255	0.447	0.229	**0.450**	0.233	0.567	**0.060**	**0.228**	0.077	0.252	0.099	**0.360**
RankZephyr	0.386	0.532	0.454	0.572	0.429	0.563	0.266	0.438	0.238	0.421	0.248	0.542	0.053	0.172	0.080	0.243	0.101	0.289
Ours																		
SFT	0.382	0.499	0.442	0.543	0.428	0.575	0.274	**0.455**	0.244	0.428	0.256	0.553	0.052	0.176	**0.086**	0.249	0.099	0.306
GRPO	0.386	0.515	0.441	0.552	0.432	0.588	**0.284**	0.452	**0.245**	0.436	**0.269**	**0.585**	0.056	0.186	**0.086**	**0.261**	**0.106**	0.310
CODI	0.384	0.509	0.439	0.547	0.426	0.547	0.271	0.445	0.241	0.428	0.253	0.550	0.055	0.179	0.083	0.252	0.104	0.297

baselines, especially in the high-redundancy group where GRPO is only 0.008 Cov@5 below the best method.

The advantage of LLM-as-MMR is clearer on QAMPARI. QAMPARI requires selecting evidence for more answer facets, and its retrieved pools are more redundant on average, making repeated evidence more costly under a fixed top-k budget. GRPO achieves the best Cov@5 in all redundancy groups and outperforms the strongest non-LLM-as-MMR baseline by 0.018, 0.007, and 0.021 Cov@5 in the low-, medium-, and high-redundancy groups. RankZephyr is a competitive listwise baseline, especially on AmbigQA, but it remains a relevance-oriented reranker and does not explicitly optimize answer coverage or set diversity. On QAMPARI and QUEST, CODI consistently exceeds RankZephyr on Cov@5, showing that the gains are not explained by listwise LLM reranking alone. CODI remains close to GRPO across datasets, indicating that latent compression preserves much of the set-selection behavior while reducing visible generation cost.

4.3 Reward Ablations

We ablate the reward design used in GRPO training while keeping the model, data, rollout setting, and training steps fixed. Table 2 reports Cov@5 for the main reward and three variants. The weight column lists the coefficients for tag validity, count validity, final coverage, and local delta, respectively.

The ablation variants remain close to the main reward, indicating that final answer coverage is the dominant learning signal. The full reward is strongest on QAMPARI high redundancy, where the model must choose among overlapping passages while preserving answer diversity. Increasing the delta weight does not consistently improve results, suggesting that local window-level gains should be balanced with final set coverage.

4.4 Effect of Candidate-Pool Length

We evaluate whether LLM-as-MMR remains effective when the retrieved candidate pool becomes longer.

This setting tests whether later candidates can revise the current selected set when the pool expands from top-50 to top-100 and top-200. Table 3 shows that the effect depends on the dataset and reranking policy. On AmbigQA, the neural reranker remains strongest, matching the main results, while RankZephyr is competitive with embedding and reranker baselines. On QAMPARI, GRPO is strongest for Top-50 and Top-100, while SFT and GRPO tie on Top-200. On QUEST, LLM-as-MMR variants outperform lexical and embedding-based baselines across pool sizes. CODI also stays above RankZephyr on QAMPARI and QUEST, supporting the claim that latent compression retains the coverage-oriented selection behavior rather than merely reproducing generic listwise relevance ranking.

4.5 Efficiency of Latent Reasoning Compression

Latent reasoning compression targets the main cost of generative set refinement: visible intermediate reasoning before the final selected set is emitted.

We compare CODI with the explicit GRPO model and the neural reranker. Opt. denotes each method's efficient inference form: the reranker uses batched scoring, GRPO uses vLLM batched gen-

Table 2: Reward ablations for GRPO training over top-100 candidate pools. We report Cov@5 by dataset and redundancy tier. The weights are ordered as tag/count/final/delta. Best results are in bold and second-best results are underlined.

Reward setting	Weights	AmbigQA			QAMPARI			QUEST		
		Low	Med.	High	Low	Med.	High	Low	Med.	High
Main	.10/.10/.60/.20	0.386	0.441	**0.432**	0.284	0.244	**0.269**	0.056	0.086	0.106
No Delta	.10/.10/.80/.00	**0.387**	0.451	0.417	**0.287**	**0.258**	0.254	**0.068**	0.086	0.102
No Format	.00/.00/.80/.20	0.382	**0.461**	0.382	0.283	0.255	0.253	0.061	**0.091**	0.105
High Delta	.10/.10/.40/.40	0.370	0.445	0.421	0.273	0.254	0.249	0.062	0.088	**0.107**

Table 3: Effect of candidate-pool length under a fixed evidence budget $k = 5$. We report Cov@5 averaged over redundancy tiers within each dataset. Best results among displayed methods are in bold and second-best results are underlined.

Method	AmbigQA			QAMPARI			QUEST		
	Top-50	Top-100	Top-200	Top-50	Top-100	Top-200	Top-50	Top-100	Top-200
BM25	0.266	0.279	0.266	0.170	0.186	0.162	0.052	0.043	0.045
Emb	0.386	0.430	0.440	0.218	0.214	0.205	0.066	0.063	0.066
MMR	0.338	0.360	0.362	0.193	0.162	0.160	0.059	0.053	0.052
Rerank	**0.394**	**0.433**	**0.451**	0.253	0.239	0.256	0.077	0.079	0.081
RankZephyr	0.383	0.423	0.436	0.268	0.251	0.268	0.078	0.078	0.080
SFT	0.374	0.417	0.437	0.276	0.258	**0.288**	**0.086**	0.079	**0.085**
GRPO	0.376	0.420	0.438	**0.283**	**0.266**	**0.288**	**0.086**	**0.083**	0.084
CODI	0.372	0.416	0.427	0.277	0.256	0.274	0.082	0.081	0.081

Table 4: Efficiency comparison of explicit and compressed reranking. Time/query is the effective time per query; for CODI-Parallel, it is measured from three concurrent CODI workers on one GPU. Best results are in bold and second-best results are underlined.

Model	Out. Tok.	Time/query	Memory
Rerank-Opt	**0**	**2.65s**	19.1GB
GRPO-Opt	3831	3.24s	66.5GB
GRPO-HF	3831	151.84s	**9.5GB**
CODI	97	10.94s	10.6GB
CODI-Parallel	97	3.65s	31.8GB

eration (Kwon et al., 2023), and CODI-Parallel runs three concurrent CODI workers on one GPU. GRPO-HF provides a controlled single-worker comparison under the same HF decoding environment as CODI. Table 4 shows that vLLM batching makes explicit GRPO already fast, but it still produces long visible outputs. CODI addresses this bottleneck by reducing output length from 3831 to 97 tokens. Compared with GRPO-HF, CODI reduces time per query from 151.84s to 10.94s. With three concurrent workers, CODI reaches 3.65s per query, approaching GRPO-Opt while using less than half of its memory. This trade-off makes LLM-as-MMR most appropriate for coverage-sensitive QA settings where missing an answer facet is costly, while lightweight neural rerankers remain preferable when latency is the dominant constraint.

5 Conclusion

We introduced LLM-as-MMR, a set-aware reranking framework for ambiguous and multi-answer QA. Instead of scoring each document independently or relying on a fixed similarity penalty, LLM-as-MMR performs sliding-window set refinement with content-aware skip and replacement decisions, allowing the reranker to reduce collectively redundant top-k evidence and improve answer coverage under a limited evidence budget. Experiments show that this formulation is most effective when evidence selection must cover multiple answer facets, especially in more redundant candidate pools. CODI further shows that much of the learned set-selection behavior can be retained while reducing visible generation cost. Overall, our results suggest that reranking for QA should be evaluated not only by individual document relevance, but also by whether the selected evidence set covers the information needed by the downstream task.

Limitations

LLM-as-MMR is designed for second-stage reranking rather than first-stage retrieval. It assumes that the retrieved candidate pool contains evidence for the target answers; if a minority interpretation is absent from the pool, no reranker can recover it. We partially account for this during reward computation by treating examples with no reachable answers neutrally, but answer-covering reranking still depends on sufficient first-stage retrieval recall. The sliding-window policy also uses the first-stage ranking as an input prior. Replacement actions allow later windows to revise earlier selections, but the method is not intended to be invariant to arbitrary or adversarial candidate orders.

The method also introduces higher inference cost than lightweight embedding or cross-encoder rerankers. Sliding-window refinement makes long candidate pools tractable, but the explicit model still generates intermediate reasoning and structured actions across multiple windows. CODI reduces this visible generation cost and preserves most of the learned set-selection behavior, but compression can still lose fine-grained distinctions when passages are highly overlapping. Finally, our coverage metric relies on strict answer-string matching. This makes evaluation reproducible and conservative, but may under-credit passages that express correct answers through paraphrase or aliasing. Stronger semantic answer normalization and judge-based coverage assessment are important directions for future work.

Ethics Statement

We hereby acknowledge that all authors of this work are aware of the ACL Code of Ethics and honor the code of conduct.

Data Use and Privacy. Our experiments use public QA benchmarks and Wikipedia-derived passages. These resources are intended for research use and do not require collecting private user data. Gold answers are used only for offline supervision, reward computation, and evaluation; they are not provided to the reranker at inference time.

Use of AI Assistants. We used large language models as writing and data-construction assistants. The scientific claims, experimental design, implementation, analysis, and final manuscript decisions were made by the authors. Any model-generated training trajectories are filtered and evaluated against task-specific constraints before use.

References

Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14.

Samuel Amouyal, Tomer Wolfson, Ohad Rubin, Ori Yoran, Jonathan Herzig, and Jonathan Berant. 2023. QAMPARI: A benchmark for open-domain questions with many answers. In *Proceedings of the 3rd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 97–110.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336.

Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence. Model card and technical report.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Sreenivas Gollapudi and Aneesh Sharma. 2009. An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World wide web*, pages 381–390.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2025. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

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

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Preprint*, arXiv:2004.12832.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. 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*, pages 2356–2362.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2025. Leveraging passage embeddings for efficient listwise reranking with large language models. *Preprint*, arXiv:2406.14848.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *Preprint*, arXiv:2305.02156.

Chaitanya Malaviya, Peter Shaw, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2023. QUEST: A retrieval dataset of entity-seeking queries with implicit set operations. *Preprint*, arXiv:2305.11694.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *Preprint*, arXiv:2309.15088.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. RankZephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv:2312.02724*.

Rodrygo L. T. Santos, Jie Peng, Craig Macdonald, and Iadh Ounis. 2010. *Explicit Search Result Diversification through Sub-queries*, pages 87–99.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. CODI: Compressing chain-of-thought into continuous space via self-distillation. *Preprint*, arXiv:2502.21074.

Cheng Xiang Zhai, William W. Cohen, and John Lafferty. 2003. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 10–17.

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. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

A Data Construction and Prompt Format

A.1 Candidate Retrieval

For each query, we construct a fixed candidate pool using BM25 retrieval over a Wikipedia passage index. In the main experiments, we retrieve the top-100 passages with Pyserini (Lin et al., 2021) from the `wikipedia-dpr-100w` index. All reranking methods operate on the same retrieved candidate pool; no method calls the retriever during reranking.

We divide the top-100 pool into contiguous windows of 20 passages, yielding five decision windows per query. LLM-as-MMR maintains a selected evidence set under a fixed budget k and updates this set as later windows are observed. This setting isolates the reranking problem from first-stage retrieval and allows us to test whether a model can revise earlier selections when later candidates cover missing answers.

A.2 Training and Evaluation Data

Training data is constructed from AmbigQA (Min et al., 2020) and QAMPARI (Amouyal et al., 2023). We evaluate on AmbigQA, QAMPARI, and QUEST (Malaviya et al., 2023). AmbigQA typically contains a compact answer space, with about two to three answers per query, while QAMPARI requires evidence selection for more answer facets. This distinction motivates our analysis by candidate-pool redundancy in the main experiments.

The main training budgets are $k \in \{3, 5, 10\}$, while the main evaluation tables report $k = 5$ unless otherwise specified. For each query and budget, the top-100 candidate pool is split into five windows. The first window initializes the selected set, and later windows require a skip or replacement decision.

For GRPO training, we construct 15,000 window-level examples from 1,000 queries, three budgets, and five windows per query. We use 13,500 examples for training and 1,500 examples for validation. CODI is trained from 13,500 GRPO rollout trajectories generated by the explicit GRPO-trained LLM-as-MMR model. These trajectories use the same sliding-window decision interface and the same final selected-set format as the explicit model.

Item	Setting
Training sources	AmbigQA and QAMPARI
Evaluation datasets	AmbigQA, QAMPARI, and QUEST
Candidate pool	Top-100 passages per query
Window size	20 passages
Windows per query	5
Training budgets	$k \in \{3, 5, 10\}$
Main evaluation budget	$k = 5$
GRPO examples	15,000 window-level examples
GRPO train / validation split	13,500 / 1,500
CODI training data	13,500 GRPO rollout trajectories

Table 5: Data construction statistics for sliding-window training.

A.3 Gold Answers and Data Usage

Gold answers are used only for offline trajectory construction, reward computation, and evaluation. They are not included in prompts during inference. During training-data construction, gold answers are used to identify which candidate passages contribute to answer coverage and to construct target set-refinement trajectories. During GRPO training, they are used by the reward function to score generated selections. During evaluation, they are used to compute coverage and ranking metrics.

This separation avoids leaking answer supervision into the inference prompt. At test time, the model receives only the query, the retrieved passages, the current selected set when applicable, and the evidence budget.

A.4 Sliding-Window Prompt Format

The prompt contains the information available to the reranker at a given decision step: the query, the passages in the current window, the current selected set for later windows, and the target budget k. The model produces a structured output that can be parsed into a selected evidence set.

For the first window, the model initializes the selected set:

```
<think> ... </think>
<select>[i1,i2,...,ik]</select>
<answer>[i1,i2,...,ik]</answer>
```

The `<select>` action must contain exactly k unique document indices from the current window. The `<answer>` field records the selected set used as the state for subsequent windows.

For later windows, the model observes the current selected set and the next candidate window. It

Field	Description
`query`	Natural-language question.
`docs`	Numbered candidate passages in the current window.
`selected`	Current selected set; present for later refinement windows.
`k`	Target evidence budget.
`window_type`	Initial selection window or later refinement window.
`gold_answers`	Gold answer strings used only for offline construction, reward computation, and evaluation.
`output`	Visible reasoning, structured action, and final selected-set answer.

Table 6: Abridged schema for sliding-window examples.

Stage	Purpose
Retrieve	Construct a fixed top-100 candidate pool.
Normalize	Normalize gold answer variants for matching.
Reachability	Identify answers that appear in the retrieved pool.
Initialize	Select a budget-constrained set from the first window.
Refine	Generate skip or replacement decisions for later windows.
Validate	Remove malformed, invalid, or inconsistent trajectories.

Table 7: Sliding-window trajectory construction pipeline.

either keeps the current set or replaces one selected passage with one newly observed passage:

```
<think> ... </think>
<skip></skip>
<answer>[current set]</answer>
```

or

```
<think> ... </think>
<replace>[new] -> [old]</replace>
<answer>[updated set]</answer>
```

The `<answer>` field is the authoritative representation used by the evaluator. This ensures that the state transition used in training and inference is identical.

A.5 Training Example Schema

Each sliding-window example contains the current decision context and the target structured output. Table 6 summarizes the fields used in SFT, GRPO, and CODI data construction.

A.6 Trajectory Construction

Trajectory construction follows the same state transition used at inference time. For each query, we first normalize gold answer variants and identify the gold answers that are reachable within the retrieved top-100 candidate pool. We use DeepSeek-V4-Pro (DeepSeek-AI, 2026) as the offline annotation model in our implementation. We then construct a budget-constrained selected set from the first window. For each later window, the constructor compares the current selected set with possible one-for-one replacements. If no candidate improves the set, the target action is `<skip>`; otherwise, the constructor records the replacement and updates the selected set.

After construction, we remove malformed or inconsistent trajectories, including examples with invalid indices, duplicated selections, incorrect budget size, or mismatched final selected sets. The resulting trajectories provide supervised targets for SFT. GRPO then starts from the SFT checkpoint and optimizes the same decision interface with coverage-oriented rewards.

B Reward and Evaluation Metrics

B.1 Reachable Answer Coverage

During reward computation, answer coverage is computed over gold answers that are reachable in the retrieved top-100 pool. If no gold answer is reachable, the final coverage component receives a neutral value rather than zero, since the reranker has no positive evidence to select from. This adjustment is used only during training reward computation.

For evaluation, we report coverage over the gold answers of each example. This distinction separates reranking quality from first-stage retrieval failure: LLM-as-MMR is evaluated as a reranker over a fixed retrieved pool and cannot recover evidence absent from that pool.

B.2 GRPO Reward Components

GRPO uses a reward composed of structural validity and answer-coverage terms. For a generated output y, let $\mathcal{C}(y)$ denote the active reward components. The scalar score is computed as a normalized weighted mean:

$$V(y) = \frac{\sum_{c \in \mathcal{C}(y)} w_c R_c(y)}{\sum_{c \in \mathcal{C}(y)} w_c}, \qquad R(y) = 2V(y) - 1.$$

Component	Weight	Role
R_{tag}	0.10	Checks whether the output contains the required XML-like tags for the current decision type.
R_{count}	0.10	Checks parsing, valid indices, uniqueness, and exact budget size.
R_{final}	0.60	Measures answer coverage of the final selected set.
R_{delta}	0.20	Rewards coverage improvement over the previous selected set for replacement decisions.

Table 8: Reward components used in GRPO training.

Hyperparameter	Setting
Base model	`Qwen3-4B-Base`
Fine-tuning	Full-parameter SFT
Training examples	14,958 sliding-window examples
Prompt template	`qwen3_nothink`
Cutoff length	8704 tokens
Maximum samples	20,000
Per-device batch size	2
Gradient accumulation	4
Global batch size	64
Learning rate	5×10^{-5}
Epochs	3
Scheduler / warmup	Cosine / 0.1
Precision	bf16

Table 9: Supervised fine-tuning hyperparameters.

The final reward $R(y)$ lies in $[-1, 1]$. The component set always includes tag validity, count validity, and final coverage. The local delta component is active for replacement decisions, where the generated update can be compared with the previous selected set.

The reward encourages the model to produce parseable actions, maintain the fixed evidence budget, and improve final answer coverage. The delta term supplies a local signal for replacement decisions, while final coverage remains the dominant objective.

B.3 Evaluation Metrics

We report answer coverage and ranking quality over the selected top-k evidence set. Strict answer coverage counts a gold answer as covered if its case-insensitive string appears in the concatenation of the selected passages and titles. This metric is intentionally automatic and reproducible; it may under-credit paraphrases and aliases, so it should be interpreted as conservative automatic coverage rather than exhaustive semantic support. Relaxed coverage additionally counts an answer as covered when all answer words longer than three characters appear in the selected evidence.

Cov@k is the fraction of distinct gold answers covered by the selected evidence set. NDCG@k uses binary relevance: a selected passage is treated as relevant if it contains at least one gold answer,

$$g(d_i) = \mathbb{I}[d_i \text{ contains at least one gold answer}].$$

Given the selected ranking $(d_1, \ldots, d_k)$, we compute

$$\begin{aligned} \text{DCG@}k &= \sum_{i=1}^{k} \frac{g(d_i)}{\log_2(i+1)}, \\ \text{NDCG@}k &= \frac{\text{DCG@}k}{\text{IDCG@}k}. \end{aligned} \tag{11}$$

NDCG therefore measures whether answer-bearing passages are placed early in the selected ranking. It is separate from Cov@k, which measures the fraction of distinct answers covered by the selected evidence set.

C Training Hyperparameters

C.1 Supervised Fine-Tuning

The supervised model is trained to follow the sliding-window set-refinement interface. It learns to produce visible intermediate reasoning, a structured action, and the final selected-set answer for each window.

C.2 GRPO Training

GRPO starts from the SFT checkpoint and optimizes the same structured decision interface. For each prompt, the policy samples a group of responses, which are scored using the reward described in Section B.2. We use a small KL penalty against the reference policy to stabilize optimization.

C.3 CODI Training

CODI (Shen et al., 2025) is trained as a continuous latent compression variant of LLM-as-MMR. It uses GRPO rollout trajectories as reference traces and preserves the same final selected-set output format. CODI changes only how the intermediate reasoning is represented: visible reasoning is

Hyperparameter	Setting
Initial policy	SFT checkpoint
Algorithm	GRPO
Training examples	13,500
Validation examples	1,500
Rollout backend	vLLM
Rollout group size	8
Train batch size	32
PPO mini-batch size	32
PPO micro-batch/GPU	1
Maximum prompt length	16,384 tokens
Maximum response length	2,048 tokens
Learning rate	1×10^{-6}
PPO epochs	1
Total training epochs	3
Clip ratio	0.2
KL coefficient	0.001
KL type	Low-variance KL
Advantage normalization	Enabled
Loss aggregation	Token mean
Precision	bf16

Table 10: GRPO training hyperparameters.

Hyperparameter	Setting
Compression method	CODI continuous latent compression
Training source	GRPO rollout trajectories
Training examples	13,500
Base checkpoint	GRPO reranking checkpoint
Adaptation	LoRA (Hu et al., 2021)
LoRA rank / alpha	128 / 32
Projection layer	Enabled, dimension 2560
Latent transitions	8
Epochs	8
Per-device batch size	1
Gradient accumulation	4
Learning rate	5×10^{-4}
Maximum token number	8000
Precision	bf16
Distillation loss factor	20
Reference loss factor	1.0

Table 11: CODI training hyperparameters.

replaced by a fixed number of continuous latent transitions before the final answer is decoded.

D CODI Latent Reasoning Compression

CODI (Shen et al., 2025) is used to compress the explicit reasoning process of the GRPO-trained LLM-as-MMR model. It does not change the reranking objective, the candidate pool, the evidence budget, or the final output format. Instead, it replaces visible intermediate reasoning with continuous latent computation and then decodes the final selected-set answer.

D.1 Training Traces

CODI training examples are obtained by rolling out the GRPO-trained explicit LLM-as-MMR policy on the same sliding-window prompts. Each rollout contains the query, the current candidate window, the current selected set when applicable, the visible reasoning trace, the structured action, and the final selected-set answer. These rollouts provide the reference reasoning path for compression.

The CODI training set contains 13,500 rollout trajectories. The training target is not a new reranking objective; it is the same selected-set answer produced under the LLM-as-MMR interface, with visible reasoning replaced by latent computation.

D.2 Compressed Reasoning Path

For each example, the reference path conditions on the prompt, the visible reasoning trace, the structured action, and the final answer. The compressed path observes the same prompt but does not decode visible reasoning. Instead, it performs eight continuous latent transitions and then decodes the final `<answer>` field.

This design preserves the decision interface used by the explicit model. For the first window, CODI predicts the initial selected set. For later windows, it predicts the updated selected set conditioned on the current selection and the new candidate window. The evaluator parses the final `<answer>` field using the same validity checks as for SFT and GRPO.

D.3 Training Objective

The CODI objective combines final-answer generation with hidden-state distillation from the full-reasoning reference path. The reference path is conditioned on the visible reasoning trace, while the compressed path replaces that trace with latent states generated by CODI. The distillation signal aligns the compressed representation with the reference representation near the answer boundary. This encourages CODI to preserve the selection behavior of the explicit GRPO policy without generating the full visible reasoning trace.

We use a distillation loss factor of 20 and a reference loss factor of 1.0. The final selected-set format remains unchanged, so CODI can be evaluated with the same parser and metrics as the explicit LLM-as-MMR models.

E Baselines and Evaluation Protocol

E.1 Main Baselines

All methods are evaluated on the same fixed candidate pools and the same evidence budget. The main baselines are:

- **BM25**: keeps the original first-stage retrieval order.
- **Embedding**: reranks candidates by query–passage embedding similarity using `Qwen3-Embedding-8B` (Zhang et al., 2025).
- **MMR**: applies maximum marginal relevance using embedding similarity for relevance and redundancy.
- **Qwen3-Reranker**: reranks candidates with `Qwen3-Reranker-8B` (Zhang et al., 2025).
- **RankZephyr**: applies zero-shot listwise reranking with RankZephyr (Pradeep et al., 2023b).
- **SFT**: supervised LLM-as-MMR trained on sliding-window set-refinement trajectories.
- **GRPO**: SFT model further optimized with coverage-oriented GRPO rewards.
- **CODI**: continuous latent compression variant trained from GRPO rollout trajectories.

We report baselines for which complete runs are available under the same candidate pools, budgets, redundancy splits, and metrics. This keeps the main comparison controlled: differences in performance reflect reranking behavior rather than changes in retrieval inputs or evaluation protocol.

E.2 Redundancy Split

To analyze when set-level refinement is most useful, we split evaluation examples by candidate-pool redundancy. For each query, we encode the top-100 candidate passages with `Qwen3-Embedding-8B` (Zhang et al., 2025) and compute the average pairwise cosine similarity among the retrieved passages. Examples are then divided into low-, medium-, and high-redundancy groups by tertiles within each dataset.

This split is used only for analysis. It does not change the candidate pool, the model input, or the reranking algorithm. Higher redundancy indicates that many retrieved passages are semantically similar, which increases the risk that a fixed top-k set contains repeated evidence rather than complementary answer coverage.

E.3 Efficiency Evaluation

For efficiency analysis, we evaluate reranking under top-100 candidate pools with $k = 5$ and window size 20. We compare the explicit GRPO model, CODI, and the neural reranker. We report output tokens, effective time per query, and memory.

The efficiency rows in the main paper use the following inference settings. Rerank-Opt uses batched scoring. GRPO-Opt uses vLLM batched generation (Kwon et al., 2023) for the explicit GRPO model. GRPO-HF evaluates the same explicit model under a single-worker HF decoding environment. CODI evaluates the compressed model with one worker, and CODI-Parallel uses three concurrent CODI workers on one GPU.

The comparison characterizes the quality–cost trade-off of latent reasoning compression. GRPO-Opt represents a strong optimized explicit-generation setting. GRPO-HF and CODI provide a controlled comparison under the same HF decoding environment, isolating the effect of reducing visible generation. CODI-Parallel reports the measured throughput under concurrent compressed inference.

F Additional Qualitative Analysis

Coverage-improving replacement. In successful cases, the model uses later windows to replace a relevant but redundant passage with one that covers a previously missing answer facet. This behavior differs from pointwise relevance reranking, where several top-ranked passages may support the same answer. Such replacements are most useful in candidate pools with high semantic overlap, where a fixed top-k budget can otherwise be spent on repeated evidence.

Compact answer spaces. When the answer space is compact, as in many AmbigQA examples, accurate relevance scoring can already recover much of the needed evidence. In these cases, the advantage of set-level refinement is smaller, and pointwise reranking can be highly competitive. This explains why the strongest method varies across datasets in the main results.

Minority-facet failures. Failures often occur when the retrieved pool is dominated by one in-

terpretation of an ambiguous query. In such cases, LLM-as-MMR may avoid redundant passages but still fail to recover a rare answer if the first-stage pool contains weak or no evidence for that facet. This limitation reflects the dependence of any reranker on the coverage of the first-stage candidate pool.

Compression behavior. CODI tends to preserve the final selected-set decisions of the explicit GRPO model when the evidence distinction is clear. Its errors are more likely when the replacement decision depends on fine-grained comparison among several partially overlapping passages. This pattern is consistent with the quality–efficiency trade-off observed in the main experiments: continuous latent compression reduces visible generation substantially while retaining most of the set-selection behavior.