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

Anonymous ACL submission

Abstract

Modern retrieval pipelines often rely on relevance-only ranking, which can return highly redundant passages and fail to cover complementary evidence needed to answer ambiguous queries. We propose an LLM-driven diversification reranker inspired by Maximal Marginal Relevance (MMR). Given a query and a fixed pool of candidate documents, our method constructs the ranked list iteratively: at each step, the LLM selects the next document by jointly considering (i) relevance to the query and (ii) marginal novelty with respect to the already selected set, explicitly penalizing information overlap while rewarding complementary coverage. This "LLM-as-MMR" formulation leverages the model's semantic understanding to estimate redundancy beyond surface-level similarity, without relying on hand-crafted similarity functions. Experiments on ambiguous question answering style reranking benchmarks show that our approach improves answer coverage and ranking quality while producing more diverse, less redundant top-k results compared to similarity-based ranking and heuristic diversification baselines.

1 Introduction

In modern Retrieval-Augmented Generation (RAG) and open-domain Question Answering (QA), the utility of a retrieval system is not defined solely by the relevance of individual documents, but by the *collective information coverage* of the top-k results (Lewis et al., 2021; Karpukhin et al., 2020). This is particularly critical for **ambiguous queries**—queries that encompass multiple valid interpretations or require multifaceted evidence to answer comprehensively (Min et al., 2020).

Current retrieval pipelines, dominated by bi-encoder embeddings and pointwise cross-encoders, typically score documents independently (Karpukhin et al., 2020; Qu et al., 2021; Nogueira and Cho, 2019). In addition to classic dense retrievers, recent open embedding models (e.g., E5 and BGE/FlagEmbedding) have further strengthened this approach (Wang et al., 2024; Xiao et al., 2024). While effective for many queries, independent scoring often leads to the **"Redundancy Trap"**: for ambiguous queries, the top-k list can contain near-duplicates of the dominant interpretation, leaving fewer slots for complementary evidence.

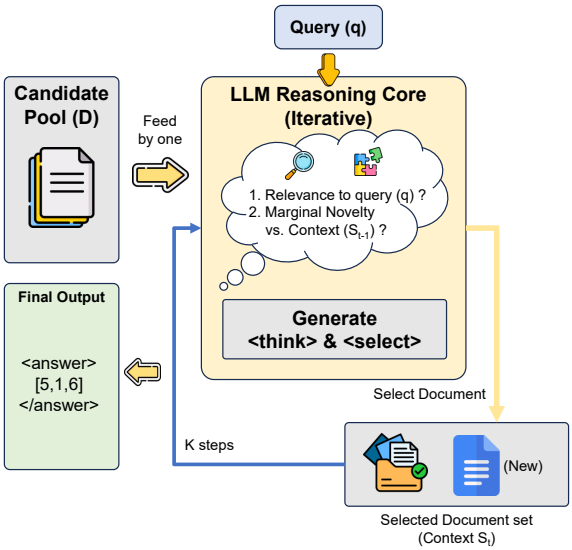

Figure 1: The LLM-as-MMR Framework. The model iteratively selects documents by explicitly reasoning about their relevance to the query (q) and marginal novelty against the selected context ($\mathcal{S}_{t-1}$), effectively performing stepwise diversification.

To address this, the classic *Maximal Marginal Relevance* (MMR) algorithm (Carbonell and Goldstein, 1998) introduced a greedy selection criterion that explicitly balances relevance with novelty. More broadly, diversification and subtopic-aware retrieval have been studied extensively (Zhai et al., 2003; Clarke et al., 2008; Agrawal et al., 2009; Gollapudi and Sharma, 2009; Santos et al., 2010). However, classical MMR relies on heuristic vector similarity (e.g., cosine similarity) to estimate

redundancy. This approach is brittle: it struggles to distinguish between *true semantic redundancy* (which should be penalized) and *lexically similar but complementary details* (which should be preserved).

In this work, we propose **LLM-as-MMR**, an LLM-based reranker that applies the MMR idea in an iterative selection procedure. Instead of using a fixed similarity function as a redundancy proxy, we cast reranking as a **sequential decision process**. At each step, the model conditions on the query and the *already selected* set, produces a short rationale, and selects the next document based on its incremental contribution to the set.

We train the reranker in three stages. First, we perform Supervised Fine-Tuning (SFT) on expert trajectories to teach the model the required tag format and the "compare-and-select" behavior, following standard instruction-tuning practices (Ouyang et al., 2022; Wang et al., 2023; Rafailov et al., 2024). Second, because imitation alone does not directly optimize set-level coverage, we apply **Reinforcement Learning (RL)** via Group Relative Policy Optimization (GRPO) to maximize the *Answer Coverage* objective under format constraints. Third, to reduce inference cost, we adopt **Coconut** compression (Hao et al., 2025), distilling each explicit step into a compact latent representation and achieving an $\approx 8\times$ speedup.

We evaluate our framework on the **AmbigQA** benchmark using a **redundancy-based difficulty stratification**. Our experiments show that in highly redundant pools, RL training improves answer coverage and achieves performance comparable to strong pairwise rerankers, while also outperforming prompted GPT-4.1 baselines.

Contributions. Our contributions are:

- We propose *LLM-as-MMR*, an iterative reranker that implements the relevance–novelty trade-off with an explicit step-wise selection format rather than a hand-crafted similarity heuristic.
- We show that **RL (GRPO)** improves answer coverage in high-redundancy pools compared to supervised baselines.
- We apply **Coconut** compression to reduce inference latency by distilling explicit steps into latent representations.
- We evaluate on AmbigQA with redundancy-based stratification and report robustness gains in the most redundant candidate pools.

2 Related Work

Document reranking has a long history in information retrieval, with most approaches focusing on query–document relevance. Our work is most closely related to (i) diversification-aware reranking, (ii) neural/LLM reranking, and (iii) post-training and efficiency techniques for controllable reranking.

Diversification-aware reranking. Maximal Marginal Relevance (MMR) is a classical greedy criterion that explicitly balances relevance to the query and novelty with respect to already selected items, reducing redundancy in ranked lists (Carbonell and Goldstein, 1998). Beyond MMR, result diversification has been studied through intent/subtopic-aware retrieval and explicit diversification objectives (Zhai et al., 2003; Clarke et al., 2008; Agrawal et al., 2009; Gollapudi and Sharma, 2009; Santos et al., 2010). This line of work motivates our formulation of reranking as iterative set construction; our method differs by using an LLM to make the relevance–novelty decision rather than relying on a fixed similarity function.

Neural and LLM reranking. Neural rerankers based on pretrained language models substantially improved passage ranking by better modeling semantic relevance (Nogueira and Cho, 2019), and late-interaction designs further improve efficiency–effectiveness trade-offs (Khattab and Zaharia, 2020). On the representation side, recent open embedding models such as E5 and BGE/FlagEmbedding have become strong foundations for dense retrieval and reranking pipelines (Wang et al., 2024; Xiao et al., 2024). Beyond these foundations, modern reranking has explored hybrid and collaborative rerankers that leverage multiple retrieval signals (Zhang et al., 2023), and efficiency-oriented reranking for generation pipelines via lattice encoding (Singhal et al., 2023). Related zero-shot reranking approaches based on question generation have also been explored (Sachan et al., 2023). Recent work has also explored listwise reranking with LLMs via prompted permutation generation (Ma et al., 2023; Pradeep et al., 2023; Liu et al., 2025), highlighting both effectiveness and efficiency constraints due to context length and decoding cost. For practical LLM reranking usage and

reproducible evaluation, we also note recent tooling that standardizes LLM-based reranking pipelines (Sharifymoghaddam et al., 2025). Building on this trend, we use an instruction-following LLM as a controllable reranker that performs step-wise selection, making the relevance–diversity trade-off explicit via a structured output format.

Post-training and efficiency. Beyond supervised fine-tuning, reinforcement learning can optimize rankers toward downstream objectives. We use GRPO-style group-relative updates implemented with the VERL training stack (volcengine, 2024) to directly optimize answer coverage under format constraints. Our use of RL is aligned with common policy-optimization recipes used in instruction tuning (Ouyang et al., 2022; Schulman et al., 2017b,a). To reduce the inference cost of explicit step-wise reasoning, we adopt progressive latent-space reasoning and compression following Coconut (Hao et al., 2025).

3 Method

We formulate document reranking as a **sequential decision-making process** driven by a Large Language Model (LLM). Unlike traditional pointwise or pairwise rerankers that score documents independently, our approach, *LLM-as-MMR*, iteratively constructs a ranked list by modeling the conditional probability of the next document given the query and the previously selected context. This section details our problem formulation, the three-stage training pipeline (Supervised Fine-Tuning, Reinforcement Learning, and Inference Compression), and the specific mechanisms designed to balance relevance and redundancy.

3.1 Task Formulation: Reranking as Iterative Selection

Let q be an ambiguous query and $\mathcal{D} = \{d_i\}_{i=1}^{n}$ be a candidate pool of retrieved documents. The objective of reranking is to generate an ordered subset $\mathcal{S}_k = (d_{i_1}, \ldots, d_{i_k})$ of size k (where $k \leq n$) that maximizes the coverage of distinct answers required by q.

We model this process as an autoregressive generation task. At each step t ($1 \leq t \leq k$), the model selects a document $d_{i_t} \in \mathcal{D} \setminus \mathcal{S}_{t-1}$ based on the query q and the history of selected documents $\mathcal{S}_{t-1}$. Inspired by the Maximal Marginal Relevance (MMR) criterion (Carbonell and Goldstein, 1998), the selection at step t aims to maximize the marginal utility:

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

While classical MMR relies on heuristic similarity functions (e.g., cosine similarity) for $\mathrm{Sim}(\cdot, \cdot)$, our method learns to approximate this trade-off implicitly through natural language reasoning. The LLM acts as a policy π_θ that outputs a reasoning path (rationale) followed by a selection decision, effectively "soft-computing" the relevance and novelty of candidates in real-time.

Inference cost. The step-wise procedure selects k documents sequentially, so inference cost scales linearly with k (one generation step per rank). In practice, this makes the approach best suited for reranking a moderate candidate pool (e.g., top-50) rather than producing very deep ranked lists. We therefore focus on $k = 3$ in our main evaluation.

3.2 Stage I: Supervised Fine-Tuning (SFT) for Stepwise Reasoning

To initialize the policy with the capability of comparative reasoning, we construct a high-quality dataset of reranking trajectories derived from the AmbigQA benchmark.

Data Synthesis Pipeline. We develop a five-stage pipeline to synthesize training data that explicitly exposes the "thinking process" of an expert reranker:

1. **Context Extraction**: We extract pairs of $(q, \mathcal{D})$ from AmbigQA, where $\mathcal{D}$ contains both relevant passages answering different sub-intents of q and distracting passages.

2. **Oracle Annotation**: Using ground-truth answers, we label each document in $\mathcal{D}$ with fine-grained tags (e.g., *fully_relevant*, *partially_relevant*, *redundant*). Note that these labels are **hidden** from the student model.

3. **Teacher Self-Decision**: A strong teacher LLM (GPT-4o; Ethics Statement) generates "optimal" reranking trajectories. We prompt the teacher to simulate the MMR process: for each step, it must explain *why* a candidate adds new information compared to $\mathcal{S}_{t-1}$.

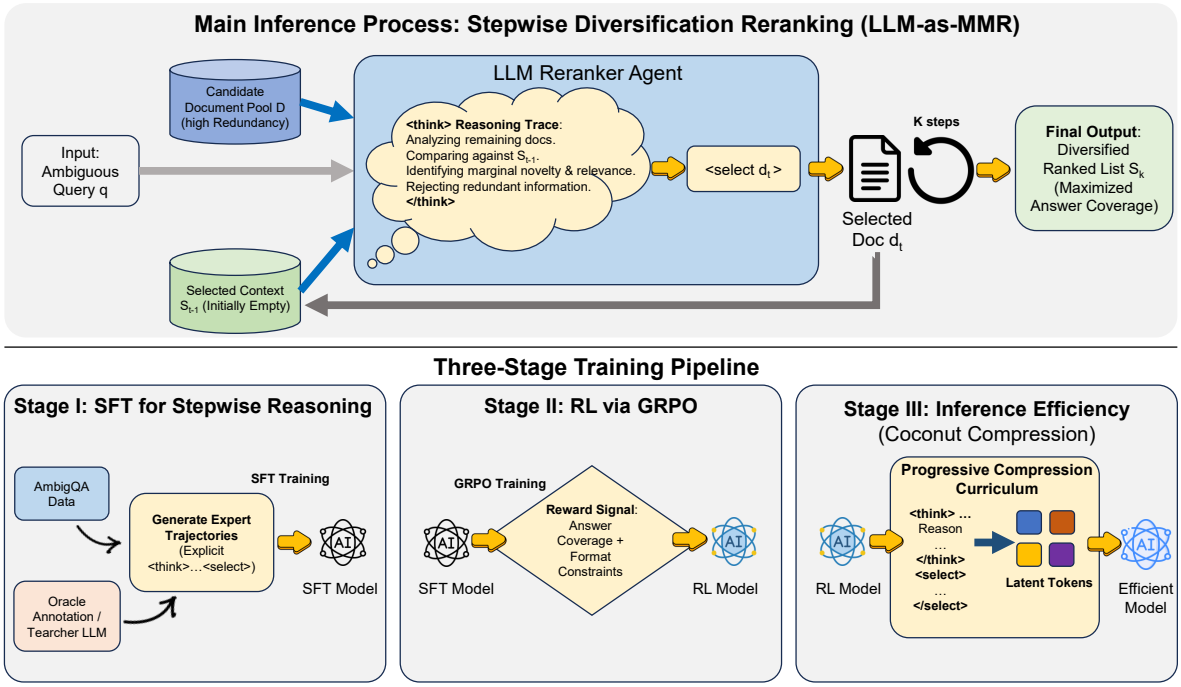

Figure 2: Overview of the LLM-as-MMR framework. (Top) Inference Process: The model functions as a stepwise agent, iteratively selecting documents (d_t) by generating explicit reasoning traces to evaluate relevance and penalize redundancy against the selected context ($\mathcal{S}_{t-1}$). (Bottom) Three-Stage Training Pipeline: The model is optimized through: (I) SFT on expert trajectories to learn the reasoning format; (II) RL via GRPO to directly maximize answer coverage (DeepSeek-AI et al., 2025); and (III) Progressive Compression (Coconut) to distill explicit reasoning into latent tokens for efficient inference (Hao et al., 2025).

4. **Trajectory Formatting**: We curate sequences for multiple target lengths $k \in \{1, 3, 5\}$ and a **dynamic termination mode**. In dynamic mode, the model learns to output `<answer>[]</answer>` when the remaining pool offers no marginal value.

5. **Quality Filtering**: We filter out trajectories where the teacher's selection contradicts the ground-truth evidence.

Structured Input-Output Format. The model takes the query and the indexed candidate pool as input. The target output is a structured chain of thought. We introduce special tokens `<think>`, `<select>`, and `<answer>` to enforce a rigid schema:

Input: `<query>... <docs>[1]...[n]...</docs>`
Output:
`<think> Analyzing... Doc [1] covers X. Doc [2] is redundant. Select [3]. </think>`
`<select> 3 </select>`
`...`
`<answer>[ 1, 3, ... ]</answer>`

This format forces the model to perform **explicit comparison** before selection, grounding the decision in the document content rather than position bias.

3.3 Stage II: Reinforcement Learning via Group Relative Policy Optimization (GRPO)

While SFT teaches the *format* of reasoning, it relies on imitation and may not optimally solve the combinatorial optimization of covering all answers. We employ Reinforcement Learning (RL) to directly optimize the end metric (Answer Coverage).

Reward Engineering. We design a composite reward function $r(y)$ that balances structural validity and semantic utility. For a generated response y:

- **Format Reward** (r_{fmt}): A strict rule-based score that enforces valid XML-like structure and selection constraints.
- **Coverage Reward** (r_{cov}): An answer-level coverage score computed by matching gold short answers against the selected documents.

The total reward is a weighted combination:

$$r(y) = 2\big(\alpha \cdot r_{\text{fmt}}(y) + (1-\alpha) \cdot r_{\text{cov}}(y)\big) - 1 \quad (2)$$

We empirically set $\alpha = 0.3$ (i.e., 30% format and 70% coverage). We also include a sparse bonus for perfect termination in dynamic mode.

Format Reward Details. The format reward r_{fmt} is computed by checking five aspects: (1) **Required tags** (0.25): the output contains `<think>`, `<select>`, and `<answer>`; (2) **Tag pairing** (0.20): tags are properly opened/closed and correctly paired; (3) **Valid indices** (0.25): selected document IDs are within range and contain no duplicates; (4) **Answer format** (0.15): the answer list follows `<answer>[N1,N2,...]</answer>`; (5) **Mode check** (0.15): in fixed mode the list length equals k, and in dynamic mode the model either selects at least one document or terminates with `<answer>[]</answer>`.

Coverage Reward Details. Given gold short answers $\{a_j\}_{j=1}^{m}$ from the dataset and the concatenated text of selected documents, we compute an answer-level match score $s(a_j) \in [0, 1]$: **exact match**: if a_j appears as a case-insensitive substring, $s(a_j) = 1$; **soft match**: otherwise, we compute a lexical overlap ratio $\textit{overlap}(a_j, T) \in [0, 1]$ between the tokenized answer a_j and the tokenized selected text T, defined as the fraction of answer tokens that appear in T:

$$\textit{overlap}(a_j, T) = \frac{|\text{tok}(a_j) \cap \text{tok}(T)|}{|\text{tok}(a_j)|}$$

If $\textit{overlap} \geq 0.5$, we set $s(a_j) = \textit{overlap}$; otherwise, we down-weight it as $s(a_j) = 0.5 \cdot \textit{overlap}$. We then define r_{cov} as the mean score over all gold answers: $r_{\text{cov}} = \frac{1}{m} \sum_{j=1}^{m} s(a_j)$.

Relation to the Reported Metric. The soft-match component is used only for RL reward shaping. The reported Cov@k metric in Section 4 is computed using case-insensitive string matching.

Hard Constraints and Bonuses. We apply additional shaping terms: in fixed mode, if the output length mismatches k, we apply a penalty of -0.3; if the format is perfect and the length matches k, we add a bonus of $+0.1$; if the model outputs an empty selection without a proper dynamic termination, we apply a penalty of -0.2.

Optimization Objective. We use the VERL framework (volcengine, 2024) to implement GRPO (DeepSeek-AI et al., 2025). For a query x, we sample a group of N outputs $\{y_i\}_{i=1}^{N}$ from π_θ. The advantage A_i is computed as $A_i = (r(y_i) - \bar{r})/(\sigma_r + \epsilon)$. The objective maximizes the advantage-weighted log-likelihood with KL regularization:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \Bigg[\frac{1}{N} \sum_{i=1}^{N} \Big(A_i \log \pi_\theta(y_i|x) - \beta \, \text{KL}(\pi_\theta || \pi_{\text{ref}}) \Big) \Bigg] \tag{3}$$

This stage aligns the model's "internal utility function" with the true goal of diverse retrieval.

3.4 Stage III: Inference Efficiency via Progressive Compression (Coconut)

The explicit reasoning steps generated in Stages I and II improve performance but incur high latency costs. To resolve this, we adapt the **Coconut** paradigm (Hao et al., 2025) to compress reasoning into latent space.

Curriculum Learning Schedule. We employ a progressive compression schedule. The training proceeds in stages defined by a schedule function $s(e)$ dependent on the epoch e:

$$s(e) = \left\lfloor \frac{e}{E_{\text{stage}}} \right\rfloor \tag{4}$$

In stage s, the first s explicit reasoning steps are replaced by $s \cdot c_{\text{thought}}$ latent tokens. As training progresses ($s \to k$), the model gradually internalizes the entire MMR reasoning loop.

Latent Token Budget. In our implementation, we set $c_{\text{thought}} = 1$, meaning each step's `<think>...</think><select>...</select>` block is compressed into a single latent token during training. We set $E_{\text{stage}} = 3$, i.e., we train each compression stage for 3 epochs before increasing the number of replaced reasoning steps.

4 Experiments

In this section, we empirically validate the effectiveness of *LLM-as-MMR*. We evaluate our approach on the AmbigQA benchmark, focusing on the model's ability to maximize answer coverage under strict ranking constraints ($k = 3$).

4.1 Experimental Setup

4.1.1 Dataset Stratification

To systematically analyze model robustness against redundancy, we stratify the validation set into three

tiers based on the **Redundancy Score** (ρ) of the candidate pool. We use Qwen3-embedding-8B (Zhang et al., 2025) to encode documents and compute ρ as the average pairwise cosine similarity.

- **Tier I (Low Difficulty)**: Diverse pools ($\rho < Q_{0.33}$).
- **Tier II (Medium Difficulty)**: Moderate overlap ($Q_{0.33} \leq \rho \leq Q_{0.67}$).
- **Tier III (High Difficulty)**: Highly redundant pools ($\rho > Q_{0.67}$).

Candidate Pool Source. We directly rerank the per-query candidate pool provided by the AmbigQA dataset release (i.e., no external retriever is used to form $\mathcal{D}$).

4.1.2 Baselines and Metrics

We compare our method against: (1) **Embedding Sim**: Standard cosine similarity ranking using Qwen3-embedding-8B. (2) **MMR**: Classical diversification with $\lambda = 0.5$. (3) **Qwen3-Reranker**: A state-of-the-art pairwise reranker (8B) (Zhang et al., 2025). (4) **GPT-4.1 (Prompted)**: A prompted LLM baseline using GPT-4.1 (OpenAI et al., 2024), instructed to follow our stepwise reasoning-and-selection format.

We report **Answer Coverage (Cov@3)** and **NDCG@3** (Järvelin and Kekäläinen, 2002). Note on AvgRel: While we monitor Average Relevance, we observe that high AvgRel in Tier III often correlates with low Coverage (the "Redundancy Trap"), as the model selects repetitive passages that are all highly similar to the query.

Answer Matching for Coverage. AmbigQA provides multiple gold short answers per query. We compute coverage by performing case-insensitive string matching of each gold answer against the concatenated text of the selected top-k documents. Cov@3 is the mean match rate over all gold answers for each query, averaged across queries.

Relevance Labels for NDCG. We assign a graded relevance label to each candidate document based on its **answer coverage ratio**, defined as the fraction of gold short answers that appear (case-insensitive string match) in that document. We use this ratio as the gain for computing NDCG@3.

Similarity Function for MMR. For classical MMR, $\mathrm{Sim}(\cdot, \cdot)$ is cosine similarity computed over Qwen3-embedding-8B embeddings.

4.1.3 Implementation Details

Our models are based on `Llama-3-8B` (Grattafiori et al., 2024).

- **SFT**: Fine-tuned on expert MMR trajectories.
- **RL**: Optimized with Group Relative Policy Optimization (GRPO) to maximize coverage reward.
- **Coconut**: Our efficient inference variant. We train the model to compress the explicit reasoning steps into **1 latent token** per selection step. We apply a multi-stage curriculum where each implicit reasoning stage is trained for **3 epochs** to ensure the latent representations effectively capture the rejection logic.

Input/output format. All LLM-based methods use the same numbered candidate format and the tag schema from §3.2; prompt templates are provided in Appendix A. We evaluate on the AmbigQA validation candidate pools using consistent case-insensitive matching for coverage and NDCG labels.

Fixed-k evaluation protocol. Unless stated otherwise, we use fixed-k reranking with k=3 for all methods and report Cov@3 and NDCG@3 on the resulting top-3 list. This isolates set-level coverage behavior under a strict budget and makes comparisons across embedding, heuristic, and LLM-based rerankers consistent.

4.2 Main Results

Table 1 presents the performance breakdown across difficulty tiers.

Overall Performance. As shown in Table 1, our methods consistently outperform the Embedding and MMR baselines. **Ours (Coconut)** achieves the highest overall coverage (42.6%), outperforming GPT-4.1 (Prompted) (41.4%). This demonstrates that specialized fine-tuning on redundancy-aware trajectories is more effective than prompt-only baselines.

Performance by Difficulty. The stratified analysis reveals distinct behaviors:

- **Tier I (Low Difficulty)**: In diverse pools, Coconut performs exceptionally well (Cov 43.0%). When candidates are distinct, the model effectively identifies the optimal subset without "over-thinking."

Table 1: Performance breakdown on AmbigQA ($k = 3$). **Bold** indicates the best performance among methods in that tier. Data is stratified by pool redundancy (Tier I=Low, Tier III=High). Note that in Tier III, while Embedding Sim fails (Cov 36.5%), our RL model maintains high robustness (Cov 41.7%).

Method	Overall (Avg)		Tier I (Low Diff.)		Tier II (Med Diff.)		Tier III (High Diff.)	
	Cov@3	NDCG@3	Cov@3	NDCG@3	Cov@3	NDCG@3	Cov@3	NDCG@3
Baselines								
Embedding Sim	37.1	23.6	39.9	24.6	34.8	22.8	36.5	23.3
MMR ($\lambda = 0.5$)	35.8	20.2	37.2	20.1	35.7	19.6	34.3	20.9
Qwen3-Reranker	41.6	32.7	42.1	30.1	40.9	31.4	41.7	**36.6**
GPT-4.1 (Prompted)	41.4	33.1	39.8	32.2	39.8	31.3	41.0	35.7
Ours								
Ours (SFT)	42.2	32.8	43.0	34.1	41.8	31.2	41.7	33.2
Ours (RL)	41.7	32.9	41.7	**34.6**	41.6	30.8	**41.7**	33.2
Ours (Coconut)	**42.6**	**33.6**	**43.0**	33.6	**42.2**	**33.4**	39.7	33.8

- **Tier III (High Difficulty)**: This tier represents the "Redundancy Trap." Embedding Sim struggles (Cov 36.5%) due to retrieving near-duplicates. **Ours (RL)** maintains robust coverage (41.7%), matching the strong Qwen3-Reranker. This indicates that explicit RL optimization helps the model navigate highly redundant spaces.
- **Coconut in Tier III**: We observe a slight dip in Coconut's performance in Tier III (39.7%) compared to RL. This suggests that extremely subtle redundancy distinctions might be harder to compress into a single latent token.

5 Analysis

In this section, we provide a comprehensive analysis of the model's reasoning process, efficiency, and limitations.

5.1 Reasoning Process and Efficiency

Reasoning Form. Our method employs an iterative selection process where the model explicitly reasons about document relevance and redundancy. Unlike black-box scorers (e.g., Qwen3-Reranker) or vector heuristics (MMR), our model generates a structured `<think>` trace before each selection. This trace evaluates the candidate's marginal utility against the already selected set $\mathcal{S}_{t-1}$, enabling interpretable and controllable diversification.

Efficiency Comparison. While explicit reasoning improves performance, it incurs latency. Table 2 compares the average inference time per query (batch size=1) on a single A100 (80GB) GPU. **Coconut** compresses the reasoning chain into 1 latent token per step, achieving a 7.7× **speedup** (106.9 ms) compared to the explicit models (810 ms), even without vLLM optimization.

Method	Latency (ms)
Ours (SFT/RL; vLLM)	≈ 810
Ours (Coconut; no vLLM)	**106.9**

Table 2: Efficiency comparison (A100 80GB, batch size=1). vLLM uses PagedAttention (Kwon et al., 2023). Coconut runs with a vanilla Transformer backend (no vLLM). Coconut achieves an $\approx 8\times$ speedup over explicit reasoning models.

5.2 Successful Case Study: Disentangling Ambiguity

We present a real-world example from the validation set to illustrate how our model handles high-redundancy scenarios.

> **Query**: *Who played george washington in the john adams series?*
> **Candidate Pool**: Contains multiple documents about the "John Adams" miniseries (Docs [1], [5], [8]), the biography book (Doc [2]), and actors (David Morse [9], William Daniels [3]).

The baseline Embedding Sim model tends to select Docs [1], [2], and [5] because they all strongly match the phrase "John Adams series", failing to pinpoint the specific actor. In contrast, our model produces the following stepwise trace:

- **Step 1 Selection**: **[9]** (David Morse).
  Reasoning: "The most important aspect... is identifying the actor... Document [9] is the only candidate that directly provides this information... compared to background documents, [9] delivers the answer in the clearest way."

- **Step 2 Selection**: **[1]** (Miniseries Info). *Reasoning*: "To build a comprehensive answer, it's important to include... contextual details... While the first selected document clearly states who played George Washington, it does not explain what the 'John Adams' miniseries is... Among remaining options, [1] stands out by providing a concise overview..."
- **Step 3 Selection**: **[8]** (Broader Context). *Reasoning*: "The selected documents so far cover the basics... To create a more comprehensive answer, it's important to add broader historical context. Document [8] achieves this by situating the miniseries within the wider landscape... This information is not present in the previously selected documents..."

Outcome: The model successfully retrieves the direct answer (David Morse) first, then supplements it with essential context (Miniseries info) and broader context (Genre info), achieving 100% coverage and a perfect ranking order. This demonstrates the model's ability to plan a cohesive information package rather than just picking keywords.

5.3 Failure Analysis

Despite the success, our analysis of Tier III failure cases reveals specific limitations.

Case 1: Temporal Ambiguity (The Simpsons). Query: *"When did the simpsons first air on television?"*. Valid answers include both 1987 (shorts) and 1989 (series). Our model failed to retrieve the 1987 date. *Reasoning*: The candidate pool was dominated by documents discussing the 1989 premiere. The model's redundancy detection mechanism correctly identified that multiple 1989 documents were redundant, but the semantic signal for the 1987 short was too weak compared to the overwhelming consensus of the 1989 date. The model prioritized the "authoritative" 1989 date and filled remaining slots with context about that specific event, missing the secondary interpretation.

Case 2: Abstract Concepts (Consubstantial). Query: *"What does consubstantial mean?"*. Valid answers span theological and philosophical definitions. *Reasoning*: The model successfully retrieved the common theological definition ("of the same being") but missed the broader philosophical definition ("common humanity"). *Analysis*: The model exhibits a bias towards the "dominant" interpretation present in the training data. While it avoids verbatim redundancy (choosing different phrasings of the theological definition), it fails to recognize that the philosophical definition represents a *distinct valid answer category* rather than just a less relevant distractor.

5.4 Discussion

The qualitative analysis highlights a critical distinction: **Redundancy Avoidance vs. Answer Diversity**. Our model excels at *Redundancy Avoidance* (e.g., rejecting two documents that say the exact same thing). However, in cases like "The Simpsons" or "Consubstantial", the challenge is *Answer Diversity* (recognizing that two factually different statements are both valid answers). Future work should focus on training objectives that explicitly reward retrieving *conflicting* or *multi-faceted* information when the query is inherently ambiguous.

6 Conclusion

In this paper, we introduced *LLM-as-MMR*, an iterative reranking method inspired by Maximal Marginal Relevance. By casting document selection as a sequential prediction problem, our approach replaces a fixed similarity-based redundancy proxy with step-wise selection conditioned on the already selected set, allowing the model to trade off relevance and marginal novelty explicitly.

Our empirical evaluation on the AmbigQA benchmark demonstrates that our approach effectively addresses the "Redundancy Trap" inherent in traditional retrieval systems. Specifically, our RL-optimized model maintains robust answer coverage in high-difficulty scenarios where standard embedding methods collapse due to severe information overlap. We further established that this reasoning capability can be efficiently deployed: by integrating the Coconut paradigm, we compressed the explicit reasoning chain into compact latent tokens, achieving an $\approx 8\times$ speedup over standard Chain-of-Thought models with minimal performance degradation.

This work suggests that improving diversity can benefit from modeling set construction explicitly, rather than relying only on stronger embeddings. Future directions include extending the objective to multi-criteria reranking (e.g., coverage, faithfulness, coherence) and exploring adaptive compression strategies that adjust the latent reasoning length based on query complexity.

Limitations

Despite the promising results, our approach has inherent constraints regarding efficiency and scalability. First, while **Coconut** compression significantly reduces latency to ≈ 100 ms, it remains an order of magnitude slower than lightweight bi-encoders, restricting its utility to second-stage reranking on small candidate pools (e.g., top-50). Furthermore, the fundamental design of *stepwise selection* is autoregressive, meaning inference cost scales linearly with the target list size k. This prevents parallelization across ranks, making the method unsuitable for generating very deep ranked lists.

Second, we observe a trade-off between reasoning compression and resolution. In our high-difficulty experiments (Tier III), the compressed model exhibited a slight performance degradation compared to the explicit Chain-of-Thought model, suggesting that a single latent token may encounter an information bottleneck when resolving extremely subtle semantic redundancies. Additionally, our empirical validation focused exclusively on AmbigQA to target ambiguity resolution; the generalization of this redundancy-aware reasoning to other domains, such as legal or medical retrieval where "redundancy" may have different definitions, remains to be verified in 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 are primarily conducted on the public **AmbigQA** benchmark (Min et al., 2020), which is derived from Wikipedia and does not contain private or personally identifiable information (PII). For the construction of our instruction-tuning dataset, we utilized a synthetic data generation pipeline where a teacher LLM (GPT-4o) generated reasoning traces based on public passages. We have manually verified a subset of these trajectories to ensure they do not propagate toxic or harmful content. All data usage complies with the licenses of the respective source datasets.

Use of AI Assistants. We employed large language models (specifically GPT-4.1) as a **writing assistant** to refine the grammatical structure and clarity of the manuscript. We also used a teacher LLM (GPT-4o) to synthesize the reasoning trajectories for Supervised Fine-Tuning, as described in Section 3.2. The core scientific ideas, experimental design, and analysis were conducted solely by the authors.

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.

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, 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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 545 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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.

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.

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.

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*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2010.08191.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2023. Improving passage retrieval with zero-shot question generation. *Preprint*, arXiv:2204.07496.

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

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017a. Trust region policy optimization. *Preprint*, arXiv:1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017b. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Sahel Sharifymoghaddam, Ronak Pradeep, Andre Slavescu, Ryan Nguyen, Andrew Xu, Zijian Chen, Yilin Zhang, Yidi Chen, Jasper Xian, and Jimmy Lin. 2025. Rankllm: A python package for reranking with llms. *Preprint*, arXiv:2505.19284.

Prasann Singhal, Jiacheng Xu, Xi Ye, and Greg Durrett. 2023. Eel: Efficiently encoding lattices for reranking. *Preprint*, arXiv:2306.00947.

volcengine. 2024. verl: Volcano engine reinforcement learning for LLMs. `https://github.com/volcengine/verl`. GitHub repository.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. *Preprint*, arXiv:2309.07597.

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.

Zongmeng Zhang, Wengang Zhou, Jiaxin Shi, and Houqiang Li. 2023. Hybrid and collaborative passage reranking. *Preprint*, arXiv:2305.09313.

A Prompting and Data Construction

This appendix documents our implementation details. Unless stated otherwise, evaluation follows the AmbigQA reranking setup (§4) and our method definition in §3. Importantly, **at inference time our reranker only observes the query and candidate documents**; any gold answers or oracle hints are used **only** during offline data synthesis/analysis.

A.1 Data Construction Pipeline (SFT Trajectories)

Overview. We synthesize step-wise "expert" reranking trajectories to initialize the student model with (i) the rigid output schema and (ii) comparative reasoning behaviors (redundancy checks, complementarity, and multi-intent coverage). Concretely, each training sample corresponds to a query q and its AmbigQA candidate pool $\mathcal{D}$, and produces a sequence of step decisions ending with a ranked list of indices.

Implementation notes. Two details are important for reproducibility and to avoid confusion with evaluation: (i) we always rerank the **dataset-provided** candidate pool (no external retriever); (ii) answer matching for both offline analysis and reported Cov@k is **case-insensitive substring matching** (§4); any soft matching is used only for RL reward shaping (§3.3).

Pipeline summary. The full five-stage procedure is summarized in Table 3.

A.2 Prompt Templates

Design goals. All prompts enforce a rigid output contract compatible with our training schema (§3.2): each step emits a `<think>` rationale and a `<select>` decision; after completing the process, the model outputs a final `<answer>` list of selected indices.

Why so many templates? We use different templates for (i) trajectory synthesis (teacher), (ii) offline auditing/analysis (oracle-style), and (iii) prompted baselines. To keep the appendix compact and readable, we present the prompts as structured tables that list the key blocks and constraints; this avoids long verbatim blocks that tend to overflow in a two-column layout.

A.2.1 Teacher Prompt for Trajectory Synthesis (GPT-4o)

Structure. The teacher prompt is composed of a *context block* (query and documents), a *selection rule block* (how to trade off relevance and redundancy), and a strict *output contract*. In synthesis-only mode, we may include oracle hints to improve coverage; these are never shown to the student at test time. See Table 4.

Selection principles. We explicitly instruct the teacher to reason about marginal utility relative to the already-selected set $\mathcal{S}_{t-1}$, rather than scoring each document in isolation. This yields trajectories that expose the redundancy-aware decision rule to the student. See Table 5.

Fixed vs. dynamic mode. Fixed-k is used for standard top-k reranking (e.g., $k \in \{1, 3, 5\}$). Dynamic mode teaches the model to stop when the remaining pool provides no additional value, which is useful for ablations and for understanding termination behavior. See Table 6.

A.2.2 Oracle-Style Labeling Prompt (Quality Control)

Purpose. This prompt is used only for **offline** data auditing and debugging during dataset construction. Given $(q, \mathcal{D})$ and the gold short answers, it asks an LLM to summarize each document's query relevance and which gold answers appear in the document text. These outputs help us sanity-check synthetic trajectories (e.g., whether answer-containing documents exist but were missed), but they are **never** exposed to the student reranker at inference time.

Template summary. We summarize the input/output structure and label set in Table 7.

A.2.3 Prompted GPT-4.1 Baseline

Purpose. This baseline evaluates how far a strong general LLM can go with prompting alone, without any fine-tuning. We prompt GPT-4.1 to follow the same step-wise selection contract as our models (`<think>`/`<select>`/`<answer>`) so that comparisons reflect **ranking behavior** rather than formatting differences.

Template summary. The baseline prompt specifies the fixed-k constraint, the required tags, and the numbered candidate pool; see Table 7.

Stage	Details
1. Context extraction	Extract $(q, \mathcal{D})$ from AmbigQA, where $\mathcal{D}$ is the per-query candidate pool released with the dataset (no external retriever used).
2. Gold normalization	Normalize gold short answers (lowercasing for matching; whitespace/punctuation normalization) and deduplicate answer variants.
3. Oracle analysis (offline)	Using gold answers, compute per-document answer coverage ratio $g(d)$ (case-insensitive substring match) to support quality control and difficulty analysis.
4. Teacher trajectory synthesis	Use a teacher LLM (GPT-4o; Ethics Statement) to produce step-wise selection traces that follow the <think>/<select>/<answer> schema. In synthesis-only mode, oracle hints (e.g., gold answers) may be provided to improve coverage, but are never exposed at test time.
5. Formatting & filtering	Serialize trajectories into instruction-tuning format; discard samples that violate tag format, select invalid indices, or contradict gold evidence (e.g., selecting a clearly irrelevant set when answer-containing docs exist).

Table 3: Five-stage data construction pipeline for SFT trajectories.

A.2.4 SFT/RL Student Model Prompts (Llama-3-8B)

Prompt variants. We use two prompt variants for the student model (§4): a **standard trace** prompt that elicits interleaved <think> and <select> tags (useful for interpretability and RL format checking), and an optional **answer-only** prompt that requests only the final <answer> list when intermediate traces are not needed.

Template summary. Both variants share the same numbered <docs> input format; see Table 7.

Non-teacher prompts (summary). To reduce the number of separate floats (and the whitespace they can create in a two-column layout), we summarize the oracle-style prompt, the prompted GPT baseline, and the student prompt variants in a single table below. See Table 7.

Reference Tables (Appendix A)

We list the corresponding reference tables below.

Field	Content (abridged)
Role	`You are an expert document ranker selecting a comprehensive, non-redundant set for an ambiguous query.`
Input blocks	`Query: {q}`; `Oracle hints (optional, synthesis-only): {h}`; `Selected: {S}`; `Remaining: {D\S}`.
Task objective	Maximize collective answer coverage of the top-k set while penalizing redundancy relative to the already-selected set.
Required behavior	Each step must explicitly compare multiple candidates and state whether each is redundant/non-redundant with respect to $\mathcal{S}_{t-1}$.
Output contract	Repeat per step: <think> ... </think> then <select>N</select>. End with <answer>[N1,N2,...]</answer>.

Table 4: Teacher prompt skeleton for trajectory synthesis (GPT-4o). Oracle hints are used *only* to synthesize higher-coverage trajectories and are never provided at test time.

Selection principles	Checklist (used in synthesis prompts)
Directness	Prefer documents that directly contain missing answer strings or unambiguous facts early (especially in the first step).
Complementarity	Prefer candidates that add new facets (different intent/aspect/definition/date) not covered by $\mathcal{S}_{t-1}$.
Redundancy control	Mark near-duplicates as redundant when they repeat the same key facts as already selected docs; avoid selecting paraphrases of the same answer unless needed for disambiguation.
Comparative reasoning	Compare at least 3–4 candidates each round and justify why the chosen candidate has the highest marginal utility.

Table 5: Selection checklist embedded in the teacher prompt to encourage explicit marginal-utility reasoning.

Mode	Additional instructions
Fixed-k	Must select exactly k documents in total. Each round must output a valid `<select>` with an in-range, non-duplicate index until $\|S\| = k$.
Dynamic	May terminate early only after checking all remaining candidates and concluding that they are irrelevant or redundant. Termination is represented by emitting `<answer>[]</answer>` (no further selections).

Table 6: Mode-specific constraints used for teacher trajectory synthesis.

Stage	Prompt summary (abridged)
Oracle-style auditing	**Input:** `Query`, `Gold answers`, and each candidate document. **Task:** summarize relevance and which answers appear; output structured labels (Core-Complete/Core-Partial/Supplementary/Related/Irrelevant). Used only for offline debugging/quality control.
GPT-4.1 (Prompted)	**System:** select exactly k documents and output only required tags. **User:** `<query>{q}</query>` and numbered `<docs>`. **Output:** interleaved `<think>` and `<select>` repeated k times, ending with `<answer>[...]</answer>`.
Student (Llama-3-8B)	**Standard (trace):** same tag format for interpretability and RL format checking. **Answer-only:** optional fast path that outputs only `<answer>[...]</answer>` when intermediate traces are not needed.

Table 7: Summary of non-teacher prompts used in our pipeline and baselines.

B Dataset Statistics and Data Formats

B.1 Dataset Statistics

Summary. Key dataset statistics are summarized in Table 8.

How the statistics are computed. We report counts over the evaluation split used in our experiments: the number of queries after exclusion for leakage prevention, the per-query candidate pool size $|\mathcal{D}|$ provided by AmbigQA, and the number of gold short answers per query as released by the dataset. The averages are computed across queries in this evaluation split.

B.2 Data Formats

Summary. The final serialized schema is shown in Table 9.

Serialization details. Each training sample contains an `instruction` (task description, including fixed-k vs. dynamic mode), an `input` that concatenates `<query>` and a numbered `<docs>` block,

Statistic (AmbigQA validation)	Value
Total queries (evaluated)	500
Avg. candidate pool size (docs/query)	22.932
Avg. number of gold short answers (answers/query)	3.754
Training queries excluded (leakage prevention)	3,000

Table 8: Dataset statistics from our preprocessing pipeline (seed=42).

Field	Example (abridged)
`instruction`	`... select exactly 3 documents step by step ...`
`input`	`<query>... </query> \n<docs> \n[1] ... \n[2] ... \n</docs>`
`output`	`<think>...</think> \n<select>9</select> ... \n<answer>[9,1,8]</answer>`

Table 9: Training sample schema (single-example, abridged) used for SFT and RL data.

and an `output` containing interleaved `<think>` and `<select>` tags followed by the final `<answer>` list. Document indices in `<select>` and `<answer>` refer to the numbering of candidates in the `<docs>` block.

Reference Tables (Appendix B)

We list the corresponding reference tables below. Tables are allowed to float for better packing in the two-column layout.

C Metrics and Reward Details

C.1 Evaluation Metrics (Expanded)

Answer Coverage (Cov@k). As defined in §4, AmbigQA provides multiple gold short answers per query. We compute Cov@k via **case-insensitive substring matching** of each gold answer against the concatenated text of the selected top-k documents, then average the match rate across answers and queries.

NDCG@k. We compute NDCG@k using each document's **answer coverage ratio** (fraction of gold short answers matched inside that document) as the graded gain (§4). Let $g(d) \in [0, 1]$ be the per-document coverage ratio and $(d_1, \ldots, d_k)$ be the ranked list:

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{g(d_i)}{\log_2(i+1)}, \qquad \mathrm{NDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{IDCG@}k}. \tag{5}$$

Component	Definition (summary)
Format reward (r_{fmt})	Rule-based checks of (i) required tags, (ii) tag pairing, (iii) valid non-duplicate indices, (iv) answer-list format, and (v) fixed/dynamic mode validity. Weight: 30%.
Coverage reward (r_{cov})	Answer-level scoring using exact case-insensitive substring match; otherwise soft lexical-overlap shaping (threshold 0.5) to provide denser reward. Weight: 70%.
Hard constraints/bonuses	Fixed mode length mismatch: -0.3; perfect format & correct length: $+0.1$; empty selection without proper termination: -0.2.

Table 10: Reward function summary (detailed definitions are in §3.3).

Redundancy Score. To stratify difficulty (§4), we embed each document with Qwen3-embedding-8B and compute the mean pairwise cosine similarity within the candidate pool.

C.2 RL Reward vs. Reported Metrics

For GRPO training (§3), we use a shaped reward that can include soft lexical-overlap components to provide denser learning signals. However, **all reported Cov@k results in the paper are computed using strict case-insensitive substring matching**, without soft matching. A compact summary is provided in Table 10.

Reference Tables (Appendix C)

D Additional Case Analysis

We provide additional qualitative examples complementing §5. In general, our model succeeds when (i) at least one candidate contains a direct answer string and (ii) remaining documents provide complementary facets; failures typically arise when one interpretation dominates the pool and the alternative answer is weakly expressed or paraphrased, making substring-based coverage hard to satisfy.

What we analyze. For each query, we examine (i) whether a selected document contains a gold short-answer string under case-insensitive matching, and (ii) whether each additional selected document contributes *new* answer facets rather than repeating the dominant interpretation. We qualitatively diagnose failures by checking whether the missing answer is absent from the candidate pool, present only via paraphrase (making substring matching hard), or dominated by pool bias where one interpretation overwhelms the alternatives.

D.1 Detailed Examples (from the AmbigQA Validation Set)

D.1.1 Successful Example: Disentangling Ambiguity via Step-wise Selection

Query: *Who played george washington in the john adams series?*
Candidate pool (summary): multiple documents about the *John Adams* miniseries (Docs [1], [5], [8]), a biography book (Doc [2]), and actor pages (David Morse [9], William Daniels [3]).

This example illustrates why independent similarity scoring is vulnerable to the redundancy trap. The Embedding Sim baseline tends to select Docs [1], [2], and [5] because they all match the surface phrase "John Adams series" while missing the core intent (the actor identity). In contrast, our model explicitly reasons about marginal utility and produces the following step-wise selections (same trace as §5):

- **Step 1**: select **[9]** (David Morse).
  Rationale: the only candidate that directly states the actor who portrayed George Washington, satisfying the primary information need.
- **Step 2**: select **[1]** (miniseries overview).
  Rationale: adds non-redundant context about what the *John Adams* series is; complements the casting fact without repeating it.
- **Step 3**: select **[8]** (broader contextual/genre list).
  Rationale: provides broader context that is not present in the first two selections, improving the completeness of the package while avoiding near-duplicates.

Overall, the ranked list `[9, 1, 8]` demonstrates the intended behavior of *LLM-as-MMR*: prioritize the direct answer first, then allocate remaining slots to complementary facets rather than redundant paraphrases.

Additional failure modes (brief). Beyond the examples in §5, we repeatedly observe two practical failure modes: (i) **paraphrase-heavy pools**, where the correct answer is present but expressed without the exact short-answer string, which hurts substring-based coverage; and (ii) **pool bias**, where one interpretation dominates the candidate set and the minority interpretation appears only in a weak or noisy document, making it hard for the model to justify allocating a top-k slot.

D.1.2 Failure Example 1: Temporal Ambiguity (The Simpsons)

Query. *When did the simpsons first air on television?*

Multiple valid answers. This query has two common interpretations: (i) the debut of *The Simpsons* shorts (1987) and (ii) the premiere of the full TV series (1989). A coverage-oriented reranker should ideally surface evidence for both.

Observed failure pattern. In our inspection, the candidate pool is often dominated by documents discussing the 1989 premiere. The model correctly identifies many 1989-focused documents as redundant with each other, but the evidence for the 1987 shorts is comparatively weak and easy to miss. As a result, the model prioritizes the "authoritative" 1989 date and uses remaining slots on context about that event, leaving the secondary interpretation uncovered (as described in §5).

D.1.3 Failure Example 2: Abstract Concepts (Consubstantial)

Query. *What does consubstantial mean?*

Multiple valid answer facets. Gold short answers span both theological and philosophical definitions (e.g., "of the same being" vs. a broader philosophical notion such as "common humanity").

Observed failure pattern. The model tends to retrieve the dominant theological definition first. While it avoids verbatim redundancy by selecting different phrasings of that interpretation, it may fail to recognize that a philosophical definition is a *distinct valid answer facet* rather than a weaker distractor. This reflects a gap between redundancy avoidance and answer diversity (as discussed in §5).