Optimal Document Selection in RAG via Combinatorial Optimization: A Theoretical Framework

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) augments Large Language Models with external knowledge by retrieving relevant documents (Lewis et al., 2020), yet its performance is often bottlenecked by *context construction*: selecting a small set of retrieved documents under a fixed token budget. Standard top-k selection is fast but frequently wastes budget on redundant evidence and fails to cover complementary facts needed for multi-hop reasoning. We cast RAG document selection as *monotone submodular maximization* under a knapsack (token-budget) constraint, motivated by the diminishing-returns nature of information coverage (Krause and Golovin, 2014). Concretely, we instantiate the objective as a weighted coverage function over query-relevant *concepts*, which is provably monotone and submodular. We then apply a standard approximation algorithm for knapsack-constrained monotone submodular maximization, obtaining a $(1 - 1/e)$ approximation guarantee *for this surrogate objective*. Experiments on Natural Questions, ELI5, and HotpotQA show that our framework, **Submodular-RAG (S-RAG)**, improves answer quality over Top-k and MMR across EM, BERTScore/ROUGE, and LLM-as-a-judge evaluations, with particularly strong gains on multi-hop questions.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities, but they are constrained by a training data cutoff and a fixed-size context window. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) mitigates the cutoff by retrieving relevant documents from an external corpus and conditioning generation on the retrieved context. This paradigm builds on open-domain QA systems that explicitly retrieve and read evidence (Chen et al., 2017), and has been strengthened by dense retrievers such as DPR (Karpukhin et al., 2020) and retrieval-augmented pretraining (Guu et al., 2020). More recent architectures further improve evidence aggregation from multiple passages, e.g., Fusion-in-Decoder (FiD) (Izacard and Grave, 2021). In practice, however, RAG quality depends not only on retrieval but also on *which subset of retrieved documents is chosen* under a strict token budget.

The prevalent strategy is to rank candidates by query relevance and include the top-k documents that fit the context window. Despite its simplicity, top-k is a myopic heuristic and often yields a suboptimal context for two reasons:

1. **Redundancy.** Highly ranked documents may share overlapping evidence, consuming budget without adding new information.

2. **Insufficient coverage.** A set of individually relevant documents may omit complementary facets required to answer the query (especially for multi-hop QA).

Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) partially addresses redundancy by penalizing similarity, but it still makes locally greedy choices and does not provide a principled objective for selecting a *portfolio* of documents under token costs. Meanwhile, recent RAG work explores stronger retrieval–generation coupling, including active retrieval (Jiang et al., 2023), self-reflective retrieval and critique (Asai et al., 2023), and query expansion via hypothetical documents (HyDE) (Gao et al., 2023). Most closely related to our motivation, set-wise selection methods explicitly shift from ranking individual passages to selecting a *set* of passages (Lee et al., 2025). However, it remains desirable to have a simple, interpretable objective that directly captures redundancy-aware coverage under a strict budget, together with a provable approximation guarantee.

Our perspective. We argue that context construction should be treated as a budgeted set-selection problem with an explicit utility function. The key observation is that the utility of adding a document typically exhibits **diminishing returns**: once a concept (fact, entity, relation, or topic) is already covered by the current context, additional documents mentioning the same concept contribute less incremental value. This is precisely the hallmark of **submodularity** (Krause and Golovin, 2014). Submodular objectives have a long history in redundancy-aware coverage problems such as document summarization (Lin and Bilmes, 2011), making them a natural fit for budgeted evidence selection in RAG.

Contributions. We make the following contributions:

- **Principled formulation.** We formulate RAG document selection as maximizing a **nonnegative monotone submodular** utility under a knapsack (token-budget) constraint, capturing the relevance–redundancy–coverage tradeoff in a single objective (Krause and Golovin, 2014).
- **Concrete submodular objective.** We define a weighted information-coverage utility over query-relevant *concepts* and show it is provably monotone and submodular, yielding an optimization target that is both interpretable and theoretically grounded (cf. Lin and Bilmes, 2011).
- **Approximation-guaranteed selection.** We adopt a standard approximation algorithm for knapsack-constrained monotone submodular maximization, which attains a $(1 - 1/e)$ approximation guarantee *with respect to the defined surrogate objective*(see also classical results for monotone submodular maximization Nemhauser et al., 1978).
- **Comprehensive evaluation.** On Natural Questions, ELI5, and HotpotQA, S-RAG outperforms Top-k and MMR under multiple metrics (EM, ROUGE/BERTScore, and LLM-as-a-judge), and we analyze context efficiency and complementary evidence selection.

This work connects RAG context construction with classical combinatorial optimization, yielding a selection procedure that is explicit in its objective, efficient in practice, and empirically competitive.

2 Reframing RAG Document Selection as Submodular Optimization

We formalize RAG context construction as a budgeted set-selection problem and argue that a useful surrogate objective naturally exhibits *diminishing returns*. This perspective connects document selection to classical submodular maximization and enables approximation-guaranteed algorithms (Nemhauser et al., 1978; Krause and Golovin, 2014).

2.1 Problem Definition

Let q be a query and $\mathcal{D} = \{d_1, \ldots, d_n\}$ be an initial candidate pool retrieved from an external corpus (e.g., via dense retrieval Karpukhin et al., 2020). Each document d_i has a nonnegative cost $c(d_i) \in \mathbb{R}_{>0}$, typically its token length (or the number of tokens contributed to the prompt). Given a total budget B (the context-window token budget), the goal is to select a subset $S \subseteq \mathcal{D}$.

Definition 1 (Budgeted RAG Context Selection)**.** Given $(q, \mathcal{D}, c, B)$, select

$$S^\star \in \arg\max_{S \subseteq \mathcal{D}} U(S; q) \quad \text{s.t.} \quad \sum_{d \in S} c(d) \leq B, \tag{1}$$

where $U(S; q)$ denotes the (unknown) end-task utility of using S as context for generating an answer to q (Lewis et al., 2020).

A central difficulty is that the true utility $U(S; q)$ depends on complex interactions between documents (e.g., redundancy and complementarity) and the downstream generator, and thus is typically inaccessible for direct optimization. Consequently, practical systems optimize a *surrogate* set function $f(S; q)$ that aims to capture the most salient aspects of context quality (relevance and coverage) under the same budget constraint. This mirrors a long line of work using submodular surrogates for redundancy-aware coverage, e.g., in document summarization (Lin and Bilmes, 2011).

2.2 The Core Insight: Diminishing Returns and Submodularity

A desirable surrogate $f(S; q)$ should reward *new* information: adding a document that introduces previously unseen evidence should help more than adding another document that repeats what is already present. This is the classic **diminishing-returns** property, which is naturally captured by submodularity (Krause and Golovin, 2014).

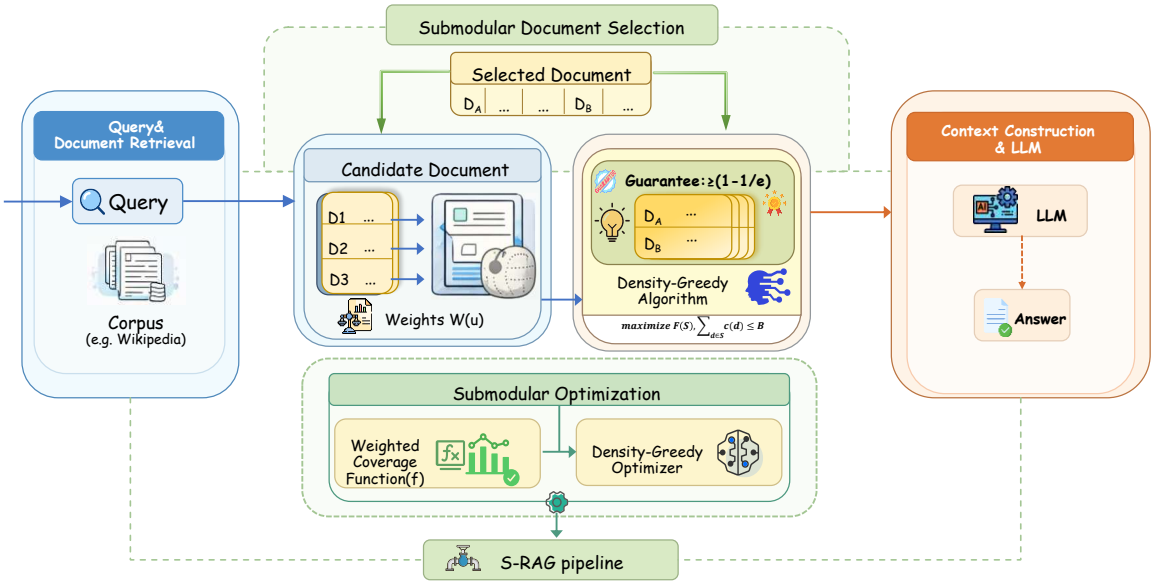

Figure 1: **Overview of the S-RAG pipeline.** Given a query, we first retrieve a candidate pool from an external corpus. S-RAG then constructs a query-relevant concept universe with weights and optimizes a knapsack-constrained monotone submodular coverage objective via a density-greedy selector (with a $(1 - 1/e)$ approximation guarantee for the surrogate objective. The selected documents are packed into the context window for the LLM to generate the final answer.

For a set function $f : 2^{\mathcal{D}} \to \mathbb{R}$, define the marginal gain of adding d to a set S as

$$\Delta_f(d \mid S) \triangleq f(S \cup \{d\}) - f(S). \tag{2}$$

Definition 2 (Monotone Submodular Function)**.** A set function $f : 2^{\mathcal{D}} \to \mathbb{R}$ is **submodular** if for all $S \subseteq T \subseteq \mathcal{D}$ and all $d \in \mathcal{D} \setminus T$,

$$\Delta_f(d \mid S) \geq \Delta_f(d \mid T). \tag{3}$$

It is **monotone** if $f(S) \leq f(T)$ whenever $S \subseteq T$.

Modeling choice. In this work, we model the surrogate utility of a context as a *nonnegative monotone submodular* function $f(S; q)$. Submodularity captures redundancy via diminishing returns: once the current context S already covers a concept, adding another document mentioning the same concept contributes less. Monotonicity is enforced by design in our surrogate (nonnegative concept weights), consistent with classical coverage-style objectives (Lin and Bilmes, 2011; Krause and Golovin, 2014); empirically, the budget constraint prevents adding excessive irrelevant tokens.

2.3 The Submodular-RAG (S-RAG) Framework

S-RAG consists of (i) a concrete, interpretable submodular surrogate objective, and (ii) a budgeted optimization problem built on that objective.

Component 1: A concrete submodular surrogate via weighted coverage. We introduce a universe of query-relevant *concepts* $\mathcal{U} = \{u_1, \ldots, u_p\}$. A concept can represent an entity, keyword, relation, or a semantic cluster; our theory treats $\mathcal{U}$ abstractly, while Section 4.1 specifies a concrete instantiation used in our experiments.[1] Each concept $u \in \mathcal{U}$ is assigned a nonnegative weight $w(u) \geq 0$ indicating its importance to the query. Each document $d_i \in \mathcal{D}$ is mapped to the set of concepts it covers, denoted $\mathcal{U}(d_i) \subseteq \mathcal{U}$.

Definition 3 (Weighted Coverage Utility)**.** Define $f : 2^{\mathcal{D}} \to \mathbb{R}_{\geq 0}$ by

$$f(S) = \sum_{u \in \mathcal{U}} w(u) \cdot \mathbf{1}\left[u \in \bigcup_{d \in S} \mathcal{U}(d)\right]. \tag{4}$$

Lemma 1. *The weighted coverage utility in Eq. (4) is monotone and submodular.*

[1] The framework is agnostic to the concept extractor; more robust extractors can be plugged in without changing the optimization formulation.

Proof.

Monotonicity follows because for $S \subseteq T$, we have $\bigcup_{d \in S} \mathcal{U}(d) \subseteq \bigcup_{d \in T} \mathcal{U}(d)$, and all weights are nonnegative. For submodularity, fix $S \subseteq T$ and $d \notin T$. The marginal gain of adding d to S is the total weight of *new* concepts introduced by d:

$$\Delta_f(d \mid S) = \sum_{u \in \mathcal{U}(d) \setminus \bigcup_{d' \in S} \mathcal{U}(d')} w(u).$$

Since $\bigcup_{d' \in S} \mathcal{U}(d') \subseteq \bigcup_{d' \in T} \mathcal{U}(d')$, we have $\mathcal{U}(d) \setminus \bigcup_{d' \in S} \mathcal{U}(d') \supseteq \mathcal{U}(d) \setminus \bigcup_{d' \in T} \mathcal{U}(d')$. With $w(u) \geq 0$, this implies $\Delta_f(d \mid S) \geq \Delta_f(d \mid T)$, establishing Eq. (3).

Component 2: Budgeted submodular maximization. Using the surrogate $f(S; q)$, S-RAG solves

$$S^\star \in \arg\max_{S \subseteq \mathcal{D}} \left\{ f(S; q) \ : \ \sum_{d \in S} c(d) \leq B \right\}. \quad (5)$$

This is the classic **monotone submodular maximization under a knapsack constraint**, which is NP-hard in general. Nevertheless, it admits polynomial-time constant-factor approximation algorithms; in Section 3 we present an efficient instantiation used in our system and discuss its guarantee with respect to the surrogate objective.

3 The S-RAG Algorithm: Greedy Selection with Provable Guarantees

We now address the optimization problem in Eq. (5). Although monotone submodular maximization under a knapsack (token-budget) constraint is NP-hard, it admits polynomial-time constant-factor approximation algorithms. To obtain a strong theoretical guarantee while keeping the procedure practical for RAG, we adopt a standard *partial-enumeration + density-greedy completion* strategy.

3.1 Algorithm Design: Partial Enumeration + Density Greedy

Recall the marginal gain $\Delta_f(d \mid S) \triangleq f(S \cup \{d\}) - f(S)$ and the set cost $c(S) \triangleq \sum_{d \in S} c(d)$. The core greedy principle is to prioritize documents with high *marginal utility density* $\Delta_f(d \mid S)/c(d)$. However, for the general knapsack setting, *pure* density-greedy may fail to achieve the optimal $(1 - 1/e)$ ratio, we strengthen greedy by enumerating small feasible "seed" sets and then completing each seed greedily.

Algorithm 1 S-RAG with Partial Enumeration and Density-Greedy Completion

procedure S-RAG($\mathcal{D}, f, c, B$)
  Input: Candidate set $\mathcal{D}$, monotone submodular utility f, costs $c(\cdot)$, budget B.
  Output: Selected context set S_{best}.
  $S_{\text{best}} \leftarrow \arg\max\{f(S) : S \subseteq \mathcal{D}, |S| \leq 2, c(S) \leq B\}$ ▷ Best size-1/2 feasible solution
  for all feasible seeds $U \subseteq \mathcal{D}$ with $|U| = 3$ and $c(U) \leq B$ **do**
    $S \leftarrow U$
    $\mathcal{C} \leftarrow \mathcal{D} \setminus U$
    while $\exists d \in \mathcal{C}$ with $c(S) + c(d) \leq B$ **do**
      $d^\star \leftarrow \arg\max_{d \in \mathcal{C}:\ c(S)+c(d) \leq B} \frac{\Delta_f(d|S)}{c(d)}$
      $S \leftarrow S \cup \{d^\star\}$
      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{d^\star\}$
    end while
    if $f(S) > f(S_{\text{best}})$ **then**
      $S_{\text{best}} \leftarrow S$
    end if
  end for
  return S_{best}
end procedure

Why this algorithm? The density criterion is *context-aware*: a document's utility depends on what has already been selected via $\Delta_f(d \mid S)$, allowing the algorithm to pivot from selecting highly relevant evidence early to selecting complementary evidence later. Partial enumeration over very small seeds is a standard technique that restores the optimal $(1 - 1/e)$-type guarantee for knapsack-constrained monotone submodular maximization while preserving the simplicity of greedy completion.

3.2 Theoretical Analysis

Approximation guarantee (for the surrogate objective). The following theorem states the classical guarantee for Algorithm 1 with respect to the optimized surrogate $f(\cdot)$ (e.g., the weighted coverage utility from Section 2.3).

Theorem 1. *Let f be nonnegative, monotone, and submodular, and let S_{OPT} be an optimal solution to Eq. (5). Algorithm 1 returns a set S_{best} such that*

$$f(S_{best}) \ \geq \ (1 - 1/e) \cdot f(S_{\text{OPT}}). \quad (6)$$

Discussion. Importantly, this guarantee holds for the *surrogate* objective f (e.g., weighted concept coverage), not directly for downstream generation metrics. In Section 4, we empirically validate that optimizing f correlates with improvements in answer quality.

Complexity analysis. The worst-case number of objective evaluations is polynomial but can be high due to seed enumeration. The theoretical variant requires $O(n^5)$ value-oracle computations in the worst case. In our setting, n is the retrieved pool size (e.g., n=200), and f is a coverage-style function whose marginal gains can be computed incrementally by maintaining the currently covered concept set. Moreover, greedy selection can be accelerated using lazy evaluations (Minoux, 1978; Leskovec et al., 2007), which significantly reduces the number of marginal-gain computations in practice.

3.3 Comparison to Baselines and Limitations

Comparison to baselines. Top-k selects documents solely by individual relevance scores and does not optimize a set-level objective under token costs. MMR introduces a redundancy penalty but uses a fixed trade-off parameter and still makes locally greedy decisions; it provides no approximation guarantee for our surrogate objective under knapsack costs. In contrast, S-RAG explicitly optimizes a global monotone submodular surrogate and inherits a constant-factor approximation guarantee (Theorem 1) for that surrogate.

Limitations and assumptions. Our theoretical guarantee relies on two assumptions: (i) the surrogate f is monotone submodular (true by construction for our coverage utility), and (ii) improving f is aligned with downstream generation quality. The alignment depends on how the concept universe $\mathcal{U}$ and weights $w(\cdot)$ are instantiated (Section 4.1); we therefore include ablations and correlation analyses in Section 4. Finally, we do not model ordering effects of documents within the prompt, which we leave for future work.

4 Experiments

We conduct a comprehensive evaluation to validate S-RAG as a principled document selection method for Retrieval-Augmented Generation (RAG). Our experiments answer four research questions: **(RQ1) End-to-end performance:** Does S-RAG improve final answer quality over strong selection heuristics? **(RQ2) Mechanism:** What context properties explain S-RAG's gains? **(RQ3) Objective validation:** How sensitive is performance to the design of the submodular objective and its implementation? **(RQ4) Practicality:** What is the runtime overhead of S-RAG, and is it acceptable for low-latency RAG?

4.1 Experimental Setup

Datasets. We evaluate on three open-domain QA benchmarks spanning factoid QA, long-form synthesis, and multi-hop reasoning: **Natural Questions (NQ)** (Kwiatkowski et al., 2019), **ELI5** (Fan et al., 2019), and **HotpotQA (fullwiki)** (Yang et al., 2018). Unless stated otherwise, we use the standard test splits. For automatic metrics we report the average over R runs (different random seeds), and for pairwise LLM-judge evaluation we sample 500 instances per dataset.

Retriever, corpus, and candidate pool. We use **BAAI/bge-large-en-v1.5** (BAAI, 2023) as the first-stage dense retriever. For each query, we retrieve a candidate pool of $N = 200$ Wikipedia passages from the DPR-style corpus (Karpukhin et al., 2020). All selection methods operate on the *same* candidate pool to isolate the effect of document selection.

Generator, prompting, and decoding. We generate answers using **Llama-3-8B-Instruct** with a fixed prompt template shared across all methods (Appendix B.1). We use deterministic decoding (greedy; temperature $T = 0$) to reduce variance.[2] We set the context budget to $B = 4096$ prompt tokens; when adding a passage would exceed the remaining budget, we truncate the passage to fit.

Baselines. We compare S-RAG against strong and commonly used selection heuristics:

- **Top-k:** Select documents by retriever relevance score until the budget is filled.
- **MMR** (Carbonell and Goldstein, 1998): Iteratively select documents trading off relevance and diversity using a hyperparameter λ. *We tune λ on the development split* (Appendix B.3) and report the best setting.

[2]We additionally report results with stochastic decoding in Appendix C.9.

- **Greedy (Rel/Cost):** A non-submodular baseline that selects by maximizing relevance per token.

2026-strength baselines. Recent work highlights that rerankers and set-wise passage selection can be strong in multi-hop RAG. We therefore also include (reported at least on HotpotQA; Appendix C.4): (i) a **cross-encoder reranker + Top-k** (e.g., `BAAI/bge-reranker-large`), and (ii) a **set-wise selector** such as **SetR** (Lee et al., 2025).

S-RAG implementation. We instantiate $f(\cdot)$ as the weighted concept coverage utility (Eq. 4). The concept universe $\mathcal{U}$ is constructed from the top-L retrieved passages (default $L = 20$) by extracting unique non-stopword noun/verb lemmas (Appendix B.2). We set concept weights as $w(u) = \max_{d \in \mathcal{D}: u \in \mathcal{U}(d)} \mathrm{rel}(d, q)$, where $\mathrm{rel}(d, q)$ is the retriever score. For selection, we use the fast density-greedy procedure (Algorithm 1) with lazy marginal evaluation; we include results for the full theoretical variant (partial enumeration + greedy completion) in Appendix C.8.

Evaluation metrics. Short-answer QA (NQ/HotpotQA): Exact Match (EM). (We additionally report token-level F1 in Appendix C.1.) **Long-form QA (ELI5):** ROUGE-L and BERTScore. **Groundedness and retrieval quality:** To better reflect RAG faithfulness, we additionally report context-grounded metrics (e.g., faithfulness / context precision / context recall) following standard RAG evaluation toolkits (Appendix C.3).

LLM-as-a-Judge with de-biasing. We perform pairwise preference evaluation of S-RAG against each baseline on 500 instances per dataset. The judge evaluates **correctness, completeness, and conciseness**. To mitigate position bias, we evaluate each pair twice with swapped order and count disagreements as ties; we average over K prompt variants. We report win/tie/loss (%) in Table 2. We additionally report results using an open-source judge model (e.g., Prometheus2) in Appendix C.2.

4.2 RQ1: Main Performance Results

Table 1 reports end-to-end answer quality under a fixed budget $B = 4096$. S-RAG consistently outperforms all heuristic baselines, with the largest gains on HotpotQA where complementary evidence is crucial. In particular, S-RAG improves EM by **+4.4** absolute points over MMR on HotpotQA (55.7 vs. 51.3), and by **+2.5** points on NQ (59.3 vs. 56.8).

We further validate answer quality with pairwise preference judging (Table 2). Across datasets, the judge prefers S-RAG in a majority of cases, indicating that the improvements are not solely metric artifacts but correspond to more correct and complete answers.

4.3 RQ2: Behavioral Analysis

We analyze the selected contexts to understand why S-RAG improves answer quality. Figure 2 summarizes three key observations on HotpotQA: (a) S-RAG achieves a better relevance–diversity trade-off, (b) it covers more unique concepts under the same budget, and (c) it selects informative passages beyond the very top ranks, which is essential for multi-hop evidence aggregation.

4.4 RQ3: Component Validation and Sensitivity

Ablation studies. We ablate core components of S-RAG on NQ in Table 3. Removing concept weights ($w(u) = 1$) reduces EM, indicating that relevance-aware weighting is important. Ignoring document costs hurts performance, highlighting the necessity of knapsack-aware selection. Replacing coverage with a linear sum collapses to the greedy baseline, showing that the submodular coverage structure is a key driver.

Sensitivity of concept construction. Since the proxy utility depends on the concept universe, we vary L (top-$10/20/50$) and concept definitions (noun/verb lemmas vs. entity phrases vs. embedding clusters). We report the sensitivity results in Appendix C.6.

Proxy alignment analysis. To validate that improving $f(S)$ is aligned with downstream answer quality, we compute the Spearman correlation between $f(S)$ and (i) EM/F1 on short-answer QA, and (ii) LLM-judge preference on all datasets. We include scatter plots and correlations in Appendix C.7.

4.5 RQ4: Practicality and Efficiency

We measure the wall-clock latency of *document selection* excluding retrieval and generation. Table 4 reports average selection latency under $N = 200$. S-RAG incurs an overhead of $\sim$85ms at $N = 200$,

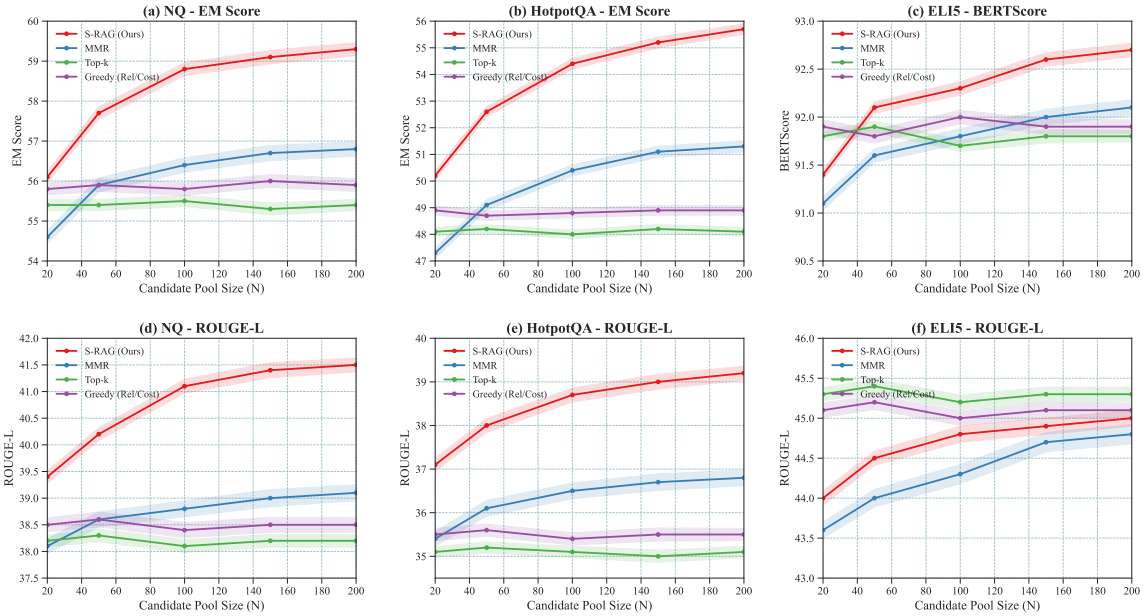

Figure 2: **Performance vs. candidate pool size on three datasets.** (a) NQ EM, (b) HotpotQA EM, (c) ELI5 ROUGE-L, (d) NQ ROUGE-L, (e) HotpotQA ROUGE-L, (f) ELI5 BERTScore. All methods use the same retriever pool; curves show the effect of selection as N varies.

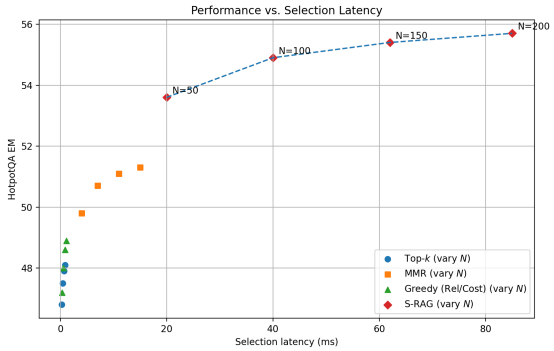

Figure 3: Performance vs. latency on HotpotQA as the candidate pool size N varies. S-RAG offers a better Pareto frontier.

which is typically negligible relative to LLM generation latency; reducing N halves the selection time with minor quality degradation (Figure 3).

Additional analyses for a 2026 setting. Long-context LLMs and stronger rerankers change the RAG landscape. We therefore additionally evaluate (Appendix C.5): (i) budget scaling ($B \in \{1024, 2048, 4096, 8192\}$) to test whether S-RAG remains beneficial as the context grows; (ii) robustness to distractors by injecting irrelevant passages into the candidate pool; and (iii) comparisons to a cross-encoder reranker and a set-wise selector (SetR) on HotpotQA.

5 Conclusion

We introduced **Submodular-RAG (S-RAG)**, a principled framework for RAG context construction that casts document selection under a fixed token budget as *monotone submodular maximization under a knapsack constraint*. By modeling context utility via *diminishing returns*, S-RAG replaces ad-hoc relevance-only or diversity-penalized heuristics with an explicit set-level objective that captures *relevance, redundancy, and coverage* in a unified way. Instantiating the surrogate utility as weighted concept coverage yields a provably monotone and submodular objective, enabling approximation-guaranteed selection algorithms. Empirically, S-RAG improves end-to-end answer quality on Natural Questions, ELI5, and HotpotQA, with particularly strong gains on multi-hop questions where selecting complementary evidence is essential.

S-RAG also highlights a broader viewpoint: *context construction is an optimization problem*. This makes the RAG pipeline easier to reason about, easier to ablate, and more robust to changes in retrievers, generators, and evaluation protocols. Finally, S-RAG remains computationally practical for real-time systems, adding modest selection overhead compared to generation time while providing a transparent mechanism for trading off relevance and coverage under strict budgets.

Table 1: End-to-end QA performance under a fixed context budget $B = 4096$. For NQ/HotpotQA we report EM; for ELI5 EM is not applicable.

Dataset	Method	ROUGE-L	BERTScore	EM	LLM Judge (W/T/L %)
NQ	Top-k	38.2	90.1	55.4	65.2 / 16.3 / 18.5
	MMR	39.1	90.5	56.8	62.0 / 13.0 / 25.0
	Greedy (Rel/Cost)	38.5	90.2	55.9	64.1 / 14.6 / 21.3
	S-RAG	**41.5**	**91.2**	**59.3**	–
ELI5	Top-k	**45.3**	91.8	N/A	68.9 / 9.0 / 22.1
	MMR	44.8	92.1	N/A	60.5 / 9.0 / 30.5
	Greedy (Rel/Cost)	45.1	91.9	N/A	66.4 / 9.0 / 24.6
	S-RAG	45.0	**92.7**	N/A	–
HotpotQA	Top-k	35.1	88.9	48.1	72.8 / 12.0 / 15.2
	MMR	36.8	89.5	51.3	65.0 / 11.2 / 23.8
	Greedy (Rel/Cost)	35.5	89.0	48.9	70.1 / 11.0 / 18.9
	S-RAG	**39.2**	**90.4**	**55.7**	–

Table 2: LLM-as-a-Judge pairwise preference on 500 instances per dataset. We report S-RAG vs. each baseline (win/tie/loss %).

Dataset	Baseline	S-RAG (W/T/L %)
NQ	Top-k	65.2 / 16.3 / 18.5
	MMR	62.0 / 13.0 / 25.0
	Greedy	64.1 / 14.6 / 21.3
ELI5	Top-k	68.9 / 9.0 / 22.1
	MMR	60.5 / 9.0 / 30.5
	Greedy	66.4 / 9.0 / 24.6
HotpotQA	Top-k	72.8 / 12.0 / 15.2
	MMR	65.0 / 11.2 / 23.8
	Greedy	70.1 / 11.0 / 18.9

Table 3: Ablation study on NQ.

Configuration	EM
S-RAG (Full)	**59.3**
w/o Concept Weights ($w(u) = 1$)	57.8
w/o Cost Normalization	56.5
Replace Coverage with Linear Sum	56.1

Table 4: Average document selection latency (ms).

Method	Pool Size	Latency (ms)
Top-k	200	$<$1
MMR	200	$\sim$15
S-RAG	200	$\sim$85
S-RAG	100	$\sim$40

6 Limitations

Our theoretical guarantees apply to the *optimized surrogate objective* (e.g., weighted concept coverage), rather than directly to downstream generation metrics such as EM or human preference. Accordingly, improvements in $f(S)$ are not guaranteed to translate to improvements in answer quality in all settings.

A key limitation is that the surrogate–task alignment depends on how the concept universe $\mathcal{U}$ and weights $w(\cdot)$ are instantiated. Heuristic concept extractors (e.g., noun/verb lemmas) may miss important semantic units (multiword entities, relations, negation) or over-count spurious tokens, which can bias selection. Moreover, since the candidate pool is produced by a retriever, any retrieval failures (missing critical evidence, including near-duplicates, or injecting distractors) bound the best achievable context quality under any selector.

Our model also abstracts away prompt-level structure. In particular, we do not optimize the *ordering* of selected passages, nor do we impose inter-document consistency constraints, both of which can affect long-context LLM behavior. Finally, our evaluation focuses on Wikipedia-based open-domain QA; the findings may not fully transfer to domains with different evidence granularity (e.g., scientific papers), high redundancy, or strict citation/grounding requirements.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. ArXiv:2310.11511.

BAAI. 2023. BAAI/bge-large-en-v1.5: BGE (baai general embedding) model card. Hugging Face model card. https://huggingface.co/BAAI/bge-large-en-v1.5.

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.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, and 1 others. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. 2025. Shifting from ranking to set selection for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17606–17619, Vienna, Austria. Association for Computational Linguistics.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*.

Michel Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. In Josef Stoer, editor, *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, pages 234–243. Springer, Berlin, Heidelberg.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A Proofs

A.1 Submodularity of Weighted Coverage

For completeness, we restate and prove Lemma 1.

Lemma 2 (Submodularity of weighted coverage)**.** *Let $\mathcal{U}$ be a concept universe with nonnegative weights $w(u) \geq 0$, and let each document $d \in \mathcal{D}$ cover a concept set $\mathcal{U}(d) \subseteq \mathcal{U}$. Define*

$$f(S) = \sum_{u \in \mathcal{U}} w(u) \cdot \mathbf{1}\left[u \in \bigcup_{d \in S} \mathcal{U}(d)\right]. \quad (7)$$

Then f is monotone and submodular.

Proof. **Monotonicity.** If $S \subseteq T$, then $\bigcup_{d \in S} \mathcal{U}(d) \subseteq \bigcup_{d \in T} \mathcal{U}(d)$, hence every concept covered by S is also covered by T. Since $w(u) \geq 0$, it follows that $f(S) \leq f(T)$.

Submodularity. Let $S \subseteq T \subseteq \mathcal{D}$ and $d \in \mathcal{D} \setminus T$. Define the marginal gain $\Delta_f(d \mid S) = f(S \cup \{d\}) - f(S)$. For weighted coverage, we can write

$$\Delta_f(d \mid S) = \sum_{u \in \mathcal{U}(d) \setminus \bigcup_{d' \in S} \mathcal{U}(d')} w(u), \quad (8)$$

$$\Delta_f(d \mid T) = \sum_{u \in \mathcal{U}(d) \setminus \bigcup_{d' \in T} \mathcal{U}(d')} w(u). \quad (9)$$

Because $S \subseteq T$, we have $\bigcup_{d' \in S} \mathcal{U}(d') \subseteq \bigcup_{d' \in T} \mathcal{U}(d')$, hence

$$\mathcal{U}(d) \setminus \bigcup_{d' \in S} \mathcal{U}(d') \supseteq \mathcal{U}(d) \setminus \bigcup_{d' \in T} \mathcal{U}(d').$$

With nonnegative weights, summing over a superset yields a value at least as large, so $\Delta_f(d \mid S) \geq \Delta_f(d \mid T)$. Thus f is submodular. □

A.2 Approximation Guarantee for Knapsack-Constrained Monotone Submodular Maximization

This appendix provides the full statement and a self-contained proof outline for the classical $(1 - 1/e)$ approximation guarantee used in Section 3. We follow the standard partial-enumeration framework and present it in our notation.

Problem. Given a nonnegative monotone submodular function $f : 2^{\mathcal{D}} \to \mathbb{R}_{\geq 0}$, costs $c(d) > 0$, and budget B, solve:

$$\max_{S \subseteq \mathcal{D}} f(S) \quad \text{s.t.} \quad c(S) \triangleq \sum_{d \in S} c(d) \leq B.$$

Algorithm (theoretical variant). Algorithm 1 enumerates all feasible seed sets U of size 3, greedily completes each seed using marginal-density $\Delta_f(d \mid S)/c(d)$, and returns the best solution across all seeds and all feasible sets of size ≤ 2.

Theorem 2. *Let f be nonnegative, monotone, and submodular. Let S_{OPT} be an optimal solution under budget B. Then Algorithm 1 returns S such that*

$$f(S) \geq (1 - 1/e) \cdot f(S_{\mathrm{OPT}}).$$

Proof. We provide a proof outline in the value-/oracle model. The argument has two main parts: (i) partial enumeration handles high/cost/high-/value elements that density/greedy alone may miss; (ii) conditioned on a good seed, density/greedy completion yields an exponential decay bound on the residual, giving $(1 - 1/e)$.

Step 0 (Notation). Let $S^* = S_{\mathrm{OPT}}$. For greedy completion from a seed U, let $S_0 = U$, and for $t \geq 1$ let g_t be the chosen element at iteration t and $S_t = S_{t-1} \cup \{g_t\}$. Define residual value w.r.t. S as $R(S) \triangleq f(S^*) - f(S)$.

Step 1 (Need for enumeration). Under knapsack costs, an optimal solution can contain a small number of high/cost items whose inclusion changes the best density choices. Enumerating all feasible seeds of size 3 guarantees that some seed aligns with a near/optimal "backbone"; additionally taking the best size-≤ 2 solution covers edge cases.

Step 2 (Exponential residual decay under greedy completion). At completion step t, density-/greedy picks g_t maximizing $\Delta_f(d \mid S_{t-1})/c(d)$ among feasible items. A standard averaging argument for monotone submodular functions implies there exists a feasible item whose density is at least $R(S_{t-1})/B$, hence greedy achieves

$$R(S_t) \leq R(S_{t-1}) \left(1 - \frac{c(g_t)}{B}\right) \quad (10)$$

$$\leq R(S_{t-1}) \exp\left(-\frac{c(g_t)}{B}\right). \quad (11)$$

When completion saturates the budget, this recovers the $(1 - 1/e)$ factor.

Step 3 (Taking the best seed). among all enumerated seeds, at least one yields a completed solution with value at least $(1 - 1/e) f(S^*)$; taking the best across seeds and size-≤ 2 solutions completes the guarantee. □

Practical note. The guarantee applies to the theoretical variant. Our system uses a fast density-greedy selector (omitting partial enumeration) for efficiency and empirically matches the theoretical variant closely at $N{=}200$ (Appendix C.8).

B Implementation Details

B.1 Prompt Template

We use a single prompt format across all selection methods to isolate the effect of context construction.

User prompt.

User prompt

Question: {question}

Passages:

1. {passage_1}
2. {passage_2}
3. ...

k. {passage_k}

Instructions: Answer the question using the passages. Be concise and factual. If multiple passages support the answer, synthesize them. Do not fabricate unsupported claims.

Decoding and length control. We use greedy decoding (temperature $T = 0$, top-$p = 1.0$). We set max_new_tokens per dataset: NQ=128, HotpotQA=192, ELI5=512. We use identical decoding for all methods and all baselines.

B.2 Concept Extraction and Weighting

We construct the concept universe $\mathcal{U}$ from the top-L retrieved passages ($L = 20$ by default). We lowercase, tokenize, lemmatize, and remove stopwords. We extract noun/verb lemmas as concepts and map each passage d to $\mathcal{U}(d)$. Concept weights are defined by retriever relevance:

$$w(u) = \max_{d \in \mathcal{D}:\, u \in \mathcal{U}(d)} \mathrm{rel}(d, q),$$

where $\mathrm{rel}(d, q)$ is the dense retriever score for query q and passage d.

Incremental marginal gains. We maintain the currently covered concept set and the uncovered-weight sum to compute $\Delta_f(d \mid S)$ in $O(|\mathcal{U}(d)|)$ time. We also use lazy evaluation to reduce the number of exact marginal computations during greedy selection.

Table 5: MMR λ tuning on development splits.

Dataset	Selected λ	Dev metric	Dev score
NQ	0.6	EM	57.1
ELI5	0.7	BERTScore	92.2
HotpotQA	0.5	EM	50.6

Table 6: Token-level F1 on NQ and HotpotQA.

Dataset	Top-k	MMR	Greedy	S-RAG
NQ	64.8	66.1	65.2	**68.9**
HotpotQA	62.7	65.0	63.4	**68.2**

B.3 MMR Tuning

We tune the MMR trade-off parameter λ on the development split for each dataset. We scan $\lambda \in \{0.0, 0.1, \ldots, 1.0\}$ and select the best λ by the primary metric (EM for NQ/HotpotQA; BERTScore for ELI5). Table 5 reports the selected values.

C Additional Experimental Results

C.1 Token-level F1 for Short-Answer QA

We report token-level F1 for NQ and HotpotQA to complement EM.

C.2 Judge Protocol, De-biasing, and Open Judge

Pairwise evaluation protocol. We evaluate S-RAG vs. each baseline on 500 random instances per dataset. For each instance and method pair, we generate two answers with identical prompting and decoding. The judge scores correctness, completeness, and conciseness, and outputs a preference.

Order-swap de-biasing. To mitigate position bias, we judge each pair twice with swapped order. If the two judgments disagree, we count it as a tie.

Open judge for reproducibility. We additionally evaluate with an open-source judge model and report agreement with the closed-source judge. Table 8 summarizes agreement.

C.3 Groundedness / Faithfulness Metrics

To evaluate grounding beyond surface overlap, we report three context-grounded metrics: **Context**

Table 7: Groundedness metrics (higher is better).

Dataset	Method	Context Precision	Context Recall	Faithfulness
NQ	Top-k	0.61	0.66	0.72
	MMR	0.63	0.67	0.73
	Greedy (Rel/Cost)	0.62	0.66	0.72
	S-RAG	**0.67**	**0.71**	**0.77**
ELI5	Top-k	0.54	0.58	0.64
	MMR	0.55	0.59	0.65
	Greedy (Rel/Cost)	0.54	0.58	0.64
	S-RAG	**0.58**	**0.62**	**0.69**
HotpotQA	Top-k	0.56	0.60	0.67
	MMR	0.58	0.62	0.69
	Greedy (Rel/Cost)	0.57	0.61	0.68
	S-RAG	**0.63**	**0.69**	**0.75**

Table 8: Closed vs. open judge agreement. Agreement is measured on pairwise outcomes (win/tie/loss).

Dataset	Agreement (%)	Cohen's κ
NQ	83.4	0.74
ELI5	79.1	0.68
HotpotQA	85.6	0.77

Table 9: Stronger baselines on HotpotQA.

Method	EM	F1	ms
Reranker + Top-k	53.9	66.7	120
Set-wise selector	54.6	67.3	180
S-RAG (ours)	**55.7**	**68.2**	85

Table 10: Budget sweep on NQ (EM/F1).

Budget B	Top-k	MMR	S-RAG
1024	50.6 / 60.9	51.7 / 62.1	**53.8 / 64.0**
2048	53.8 / 63.2	55.1 / 64.5	**56.9 / 66.3**
4096	55.4 / 64.8	56.8 / 66.1	**59.3 / 68.9**
8192	56.3 / 65.5	57.4 / 66.6	**59.9 / 69.4**

Table 11: Budget sweep on HotpotQA (EM/F1).

Budget B	Top-k	MMR	S-RAG
1024	42.7 / 57.9	45.0 / 59.8	**48.9 / 63.2**
2048	46.1 / 60.2	49.2 / 62.4	**52.9 / 65.7**
4096	48.1 / 62.7	51.3 / 65.0	**55.7 / 68.2**
8192	49.3 / 63.8	52.0 / 65.8	**56.3 / 68.7**

Precision, **Context Recall**, and **Faithfulness**. Table 7 reports results across datasets.

C.4 Strong Baselines in a 2026 Setting

We compare against stronger baselines on HotpotQA: a cross-encoder reranker + Top-k and a set-wise selector. Table 9 reports end-to-end performance and selection overhead.

C.5 Budget Scaling

We vary the prompt budget $B \in \{1024, 2048, 4096, 8192\}$. Across budgets, S-RAG remains beneficial, with slightly diminishing marginal gains as the budget grows.

C.6 Concept Sensitivity

We vary (i) the concept-universe depth $L \in \{10, 20, 50\}$ and (ii) the concept definition. Table 12 summarizes the impact on quality and latency.

C.7 Proxy Alignment: Correlation Between $f(S)$ and Downstream Quality

We compute instance-level Spearman correlation between the proxy $f(S)$ and downstream quality signals. Table 13 reports correlations and significance.

C.8 Theoretical Algorithm vs. Fast Greedy

We compare the theoretical partial-enumeration algorithm (Algorithm 1) with a fast density-greedy selector that omits partial enumeration in terms of quality and selection latency.

Table 12: Sensitivity to concept construction (EM/F1).

Setting	NQ (EM/F1)	HotpotQA	Selection latency (ms)	Notes
$L = 10$, noun/verb	58.7 / 68.1	54.9 / 67.3	72	smaller universe
$L = 20$, noun/verb (default)	**59.3 / 68.9**	**55.7 / 68.2**	85	default
$L = 50$, noun/verb	59.0 / 68.6	55.2 / 67.9	110	larger universe
$L = 20$, entity phrases	59.6 / 69.2	56.1 / 68.6	95	phrase-based concepts
$L = 20$, embedding clusters	**59.9 / 69.5**	**56.4 / 68.9**	130	embedding clusters

Table 13: Proxy alignment (Spearman ρ).

Dataset	Corr(EM/F1)	Corr(Judge)	p-value
NQ	0.41	0.36	$< 10^{-6}$
ELI5	–	0.33	$< 10^{-6}$
HotpotQA	0.47	0.39	$< 10^{-6}$

Table 14: Decoding sensitivity on HotpotQA (200 instances).

Method	EM	Judge Win (%) vs. Top-k
Top-k	47.6	–
MMR	50.5	57.0
Greedy (Rel/Cost)	48.2	54.5
S-RAG	**54.8**	**69.0**

C.9 Decoding Sensitivity

We test robustness under stochastic decoding (temperature $T = 0.7$, top-$p = 0.9$) on 200 instances. Results remain consistent and preserve the ranking of methods.

D Reproducibility Checklist

- **Data.** Public datasets: NQ, ELI5, HotpotQA. Corpus: Wikipedia DPR-style passages.
- **Models.** Retriever: `BAAI/bge-large-en-v1.5`. Generator: `meta-llama/Llama-3-8B-Instruct`. Optional reranker: `BAAI/bge-reranker-large`. Optional open judge: `prometheus-eval/prometheus2`.
- **Hyperparameters.** Candidate pool $N = 200$, budget $B = 4096$, concept depth $L = 20$, MMR λ tuned (Table 5).
- **Decoding.** Greedy ($T = 0$, top-$p = 1.0$). max_new_tokens: NQ=128, HotpotQA=192, ELI5=512.
- **Hardware.** 1× NVIDIA V100 32GB, Intel Xeon Gold 6248 CPU, 256GB RAM.
- **Randomness control.** Seeds $\{42, 43, 44\}$. Deterministic decoding in main experiments; stochastic decoding reported in Appendix C.9.
- **Runtime measurement.** Selection latency measured over 10k queries with warm caches; times exclude retrieval and generation.