GENERATIVE INFORMATION RETRIEVAL FOR THE REAL WORLD

A Dissertation Presented

by

Hansi Zeng

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy in Computer Science

September 2026

Graduate Program in Computer Science

© Copyright by Hansi Zeng 2026
All Rights Reserved.

GENERATIVE INFORMATION RETRIEVAL FOR THE REAL WORLD

A Dissertation Presented
By
Hansi Zeng

Approved as to style and content by:

_____
Hamed Zamani, Chair

_____
James Allan, Member

_____
Razieh Rahimi, Member

_____
Chen Luo, Outside Member

_____
Neena Thota, Associate Dean for
Educational Programs and Teaching
Manning College of Information and Computer
Sciences

ACKNOWLEDGMENTS

I owe my deepest gratitude to my advisor, Hamed Zamani. When I began my Ph.D., I was still learning what it means to do research well: how to recognize a meaningful problem, how to turn an idea into a careful study, and how to explain why a result matters. Hamed taught me these things patiently and by example. From him, I learned to value research taste, honest experimentation, clear writing, and the discipline of asking the right question before rushing toward an answer. He set a high standard, but he also gave me room to explore, disagree, fail, and choose my own path. That combination of guidance, trust, and freedom shaped the researcher I became. I feel fortunate to have had him as my advisor, and even more fortunate to have learned from someone who is both an excellent mentor and a genuinely good person.

I am also grateful to my dissertation committee, James Allan, Razieh Rahimi, and Chen Luo. Their questions, suggestions, and careful feedback helped me think more clearly about this dissertation and about the broader research problems behind it. I appreciate the time they spent reading my work and the guidance they provided throughout this process.

My internships were an important part of my growth as a researcher. At Lowe's, Surya Kallumadi and Zaid Alibadi guided me through my first industry research internship and helped me understand how research ideas are shaped by real systems, users, and practical constraints. At Amazon, Chen Luo and Sheikh Muhammad Sarwar gave me generous research freedom and helped me learn how to develop a project in an applied research environment. At Google DeepMind, Kai Hui, Zhen Qin, and Honglei Zhuang exposed me to the pace and standards of research in a frontier lab; working with them helped me see how strong researchers sharpen problems, design experiments, and judge impact. During my second internship at Amazon, Zoey Li and Yifan Gao taught me a great deal about foundation-model research, especially reinforcement-learning training and how to make these methods work in practice. At Snap, Neil, Liam, and Bhuvesh gave me the space to continue doing research as a part-time researcher, together with thoughtful feedback that

helped shape the work.

I thank my parents for their unconditional love, support, and trust. Their encouragement has been quiet, steady, and essential. This journey would not have been possible without the confidence and stability they gave me.

Finally, I am grateful to my labmates, internship friends, and friends at UMass and beyond. The meals, trips, basketball games, conversations, and many small moments we shared made these years much happier and more memorable. They gave me a life outside research and helped me get through difficult periods with more balance and perspective. They reminded me that a Ph.D. is not only a research journey, but also a part of life, and that friendship, balance, and joy matter along the way.

This work was supported in part by the Center for Intelligent Information Retrieval, in part by the Amazon Alexa Prize Competition, in part by Lowes, in part by Adobe Research, in part by NSF grant #2402873, and in part by the Office of Naval Research contract #N000142412612. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

ABSTRACT

GENERATIVE INFORMATION RETRIEVAL FOR THE REAL WORLD

SEPTEMBER 2026

HANSI ZENG

B.S., NANKAI UNIVERSITY

M.A., UNIVERSITY OF WISCONSIN MADISON

M.S., UNIVERSITY OF UTAH

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Hamed Zamani

Information retrieval (IR) offers external knowledge for search systems and applications built with large language models (LLMs). This dissertation studies generative information retrieval (GIR), where each document is assigned a unique document identifier (DocID) and retrieval is performed by generating DocIDs from a query. GIR connects retrieval with sequence generation. To be useful, it must scale to large corpora and decode identifiers efficiently.

The first part studies GIR as a standalone retriever. Chapter 3 introduces RIPOR, which constructs relevance-aware DocIDs through residual quantization and trains the generative model with prefix-oriented ranking optimization. On MSMARCO, a benchmark with 8.8 million passages, RIPOR substantially improves over previous GIR models and narrows the gap with strong dense retrievers. Chapter 4 introduces PAG, which combines set-based lexical identifiers with sequential identifiers. PAG uses simultaneous decoding to estimate document-level relevance before autoregressive decoding, improving MRR@10 by 15.6% over RIPOR with a 10$\times$ smaller beam and 22$\times$ lower query latency on a single A100 GPU.

The second part studies retrieval in agentic search. In this setting, an LLM-based search agent repeatedly issues search queries, reads retrieved evidence, and updates its reasoning before answering. Good retrieval is therefore not just relevant in isolation; it must help the agent solve the task. Chapter 5 presents CoSearch, a reinforcement learning framework that

jointly trains a reasoning agent and a generative document ranker. CoSearch uses semantic grouping and a composite reward to train the ranker from multi-turn rollouts. Experiments on seven QA benchmarks show consistent improvements over strong baselines, achieving +6.6% relative F1 over Search-R1 with a 7B agent and +10.0% with a 3B agent.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS iv

ABSTRACT vi

LIST OF TABLES xi

LIST OF FIGURES xii

CHAPTER

1. INTRODUCTION 1

    1.1 Optimization in Generative Information Retrieval 2
    1.2 Decoding in Generative Information Retrieval 4
    1.3 Unifying Reasoning and Generative Information Retrieval 5
    1.4 Contributions 7

2. BACKGROUND AND RELATED WORK 9

    2.1 Neural Embedding Models 9

        2.1.1 Dense Retrieval 11
        2.1.2 Sparse Retrieval 12
        2.1.3 Fundamental Components for Improving Embedding Models 14

    2.2 Generative Retrieval Models 15

        2.2.1 Document Identifier Construction 16
        2.2.2 Training Objectives 17
        2.2.3 Inference and Search Algorithms 17
        2.2.4 Scalability and Limitations 17

    2.3 Neural Reranking Models 18
    2.4 Retrieval-Augmented Generation and Search Agents 19

        2.4.1 Retrieval-Augmented Generation 20
        2.4.2 Search Agents and Reinforcement Learning 20

3. OPTIMIZATION IN GENERATIVE INFORMATION RETRIEVAL 22

    3.1 Introduction to Generative IR 25
    3.2 Methodology 27

3.2.1 Prefix-Oriented Ranking Optimization . . . 28
3.2.2 Relevance-Based DocID Construction . . . 30
3.2.3 Optimization Details . . . 32

3.3 Experiments . . . 34

3.3.1 Experiments Settings . . . 34
3.3.2 Experiment Results . . . 35
3.3.3 Analysis and Discussion . . . 39

3.4 Summary . . . 40

4. DECODING IN GENERATIVE INFORMATION RETRIEVAL . . . 42

4.1 Constrained Beam Search in GIR . . . 44
4.2 Methodology . . . 46

4.2.1 Planning-Ahead Constrained Beam Search . . . 46
4.2.2 DocID Construction . . . 49
4.2.3 PAG Optimization . . . 52

4.3 Experiments . . . 55

4.3.1 Experiment Settings . . . 55
4.3.2 Main Results . . . 58
4.3.3 Analysis and Discussion . . . 59

4.4 Summary . . . 62

5. UNIFYING REASONING AND GENERATIVE INFORMATION RETRIEVAL . . . 64

5.1 Methodology . . . 66

5.1.1 Problem Formulation . . . 67
5.1.2 Main Agent Optimization . . . 68
5.1.3 Ranker Optimization . . . 68
5.1.4 Ranker Reward Design . . . 70

5.2 Experiments . . . 72

5.2.1 Experimental Setup . . . 72
5.2.2 Main Results . . . 73
5.2.3 Ablation Study . . . 74
5.2.4 Analysis . . . 75

5.3 Summary . . . 77

6. CONCLUSION 78

6.1 Summary 78
6.2 Limitations and Future Directions 79

A. SUPPLEMENTARY MATERIAL FOR CHAPTER 3 83

B. SUPPLEMENTARY MATERIAL FOR CHAPTER 5 88

BIBLIOGRAPHY 108

LIST OF TABLES

Table Page

1. Experimental results on MSMARCO and TREC Deep Learning Track Data. 34

2. Ablation study results on MSMARCO Dev. 36

3. The retrieval performance for various DocID combinations on MSMARCO Dev set. 37

4. Experimental results on MSMARCO and TREC Deep Learning Track Data. 56

5. Ablation study results on MSMARCO Dev. 59

6. Effectiveness and efficiency with different m and k on MSMARCO Dev. 60

7. Main results (token-level F1) on seven QA benchmarks. 73

8. Ablation study. 74

9. Effect of the number of ranked documents K on Avg F1. 76

10. Average unique number of prefixes for relevant documents. 83

11. The training time of each round in the rank-oriented fine-tuning phase. 85

12. The training time of each round when removing the progressive training in the prefix-oriented fine-tuning round. 86

13. Experimental results on the MSMARCO-1M subset. 87

14. Sub-queries after semantic grouping and filtering (Part 1 of 2). 98

15. Sub-queries after semantic grouping and filtering (Part 2 of 2). 102

16. Oracle retrieval gap (F1 score). 106

17. Search turn distribution (% of questions) at validation step 100. 107

LIST OF FIGURES

Figure Page

1. A length-three semantic DocID example. 23

2. An illustration of generative retrieval models. 26

3. An overview of the RIPOR framework. 27

4. Clusters of the relevant documents to 20 queries sampled from TREC DL. The color indicates the query ID. 37

5. The retrieval performance for different prefix lengths in MSMARO Dev. 38

6. RIPOR effectiveness and latency across beam sizes on MS MARCO Dev. 45

7. A visualization of the PAG framework. 46

8. Results on MS MARCO Dev with different beam sizes. 61

9. Per-query analysis of PAG on TREC-19/20 and MS MARCO Dev. 63

10. Both panels share the same architecture: Main Agent $\rightarrow$ Retriever $\rightarrow$ Ranker. 66

11. Training and evaluation dynamics across 100 steps. 75

12. Ranker training dynamics across 100 steps. 76

13. Unique number of codes per position for documents relevant to the same query. . . 84

14. Semantic grouping for GRPO training. 96

CHAPTER 1

INTRODUCTION

Information Retrieval (IR) is the mechanism through which modern AI systems access information beyond what is stored in their parameters or immediately available in their input. It has long supported large-scale search, recommendation, and enterprise information access. The rise of large language models (LLMs) has made this role even more central. LLMs can generate fluent and useful text, but they cannot reliably memorize all current, domain-specific, or long-tail knowledge. For many knowledge-intensive tasks, they must retrieve external evidence before they can reason or answer accurately. In this sense, retrieval is no longer only a user-facing search technology; it is also the knowledge interface that grounds generative models in external information.

In most LLM systems, this connection between retrieval and generation is implemented by placing a conventional retriever outside the generative model: documents are stored in sparse or dense indexes, and the model receives retrieved text only after search has been performed. Generative Information Retrieval (GIR) explores a different formulation. It keeps the goal of retrieval the same, but changes how documents are represented and selected. Each document is assigned a unique document identifier (DocID), and a generative model is trained to produce the DocIDs of relevant documents conditioned on a query.

The appeal of GIR comes from bringing retrieval into the same generative modeling framework that underlies modern foundation models. A generative retriever estimates relevance by modeling the probability of a DocID conditioned on the query, rather than by applying a fixed similarity function to independently encoded query and document vectors. This gives the model a more expressive way to learn query-document relevance. More importantly, retrieval and natural-language generation can share the same model backbone, making it easier to unify retrieval with language tasks such as reasoning and answering.

Recent reasoning models provide a useful example: generating reasoning before the final answer often improves answer quality; similarly, a generative retriever can reason about the information need before generating the DocIDs to retrieve.

However, the conceptual appeal of GIR does not by itself make it practical. The first question addressed in this dissertation is whether GIR can work as a standalone retriever at realistic corpus scale. Existing GIR models often struggle on large-scale IR benchmarks containing millions of documents, where strong dense and lexical retrievers remain difficult to match. They also rely on autoregressive constrained decoding, making inference sensitive to beam size and therefore expensive at query time. Chapters 3 and 4 address these two practical bottlenecks: Chapter 3 focuses on DocID construction and training objectives, while Chapter 4 focuses on efficient decoding.

A separate but related question arises when retrieval is used inside LLM-based reasoning systems. In particular, a search agent does not simply consume a one-shot ranked list for a fixed query; it invokes retrieval repeatedly while reasoning, searching, and updating its intermediate state. In this setting, the usefulness of retrieved documents is ultimately measured by whether they help the agent answer the question. Chapter 5 studies this setting by jointly optimizing a reasoning agent and a generative document ranker.

1.1 Optimization in Generative Information Retrieval

At the level of optimization, GIR turns retrieval quality into a question of what identifiers the model learns to generate and how it is trained to rank them. Each document is represented as a unique DocID, and the model retrieves by generating the DocIDs that are likely to be relevant to a query [Tay et al., 2022, Wang et al., 2022, Zhuang et al., 2022b]. During inference, constrained beam search is typically used to decode the top-K valid identifiers, which are then mapped back to documents.

Existing GIR methods mainly differ along two design dimensions. First, DocID designs generally fall into two types: semantic-based DocIDs [Tay et al., 2022, Mehta et al., 2023,

Wang et al., 2022, Zhou et al., 2022, Zeng et al., 2024], which are derived from clustering or quantization over document representations, and token-based DocIDs [Chen et al., 2022a, Bevilacqua et al., 2022, Li et al., 2023b, Zhang et al., 2024], which are constructed directly from lexical features of the document. Second, training objectives vary between standard cross-entropy loss [Tay et al., 2022, Mehta et al., 2023, Wang et al., 2022] and ranking-oriented objectives [Li et al., 2023a, Zeng et al., 2024], including pairwise or listwise losses that aim to better align generation probabilities with retrieval relevance.

The central practical obstacle is scalability. Empirical studies show that GIR models often underperform strong dense retrieval baselines and, in some cases, even traditional lexical methods such as BM25 [Robertson and Zaragoza, 2009, Robertson et al., 1994] when evaluated on datasets containing millions of documents, such as the MSMARCO 8.8 million passage ranking benchmark [Campos et al., 2016, Pradeep et al., 2023a]. These observations indicate that naively reformulating retrieval as generation does not guarantee competitive performance at scale.

Chapter 3 addresses this scalability challenge. Through analysis, we identify two main sources of performance degradation: suboptimal DocID construction and misalignment between ranking objectives and autoregressive decoding dynamics. First, existing DocID construction methods are typically independent of retrieval relevance. Semantic-based identifiers derived from general-purpose embeddings capture linguistic similarity rather than query-dependent relevance, which constrains the representational capacity of GIR models once DocIDs are fixed. Second, commonly used ranking losses optimize full-length DocID scores but do not explicitly account for the prefix-level pruning behavior induced by constrained beam search. Since decoding is autoregressive, relevant documents can be irreversibly discarded if their prefixes are assigned low scores in early decoding steps. This mismatch between training objectives and inference dynamics further limits scalability.

To address these challenges, we propose two complementary techniques. We introduce relevance-based DocID construction, in which document representations are learned with

retrieval-aware objectives before being decomposed into structured identifiers. In addition, we propose prefix-oriented ranking optimization, a training strategy that explicitly enforces relevance discrimination at the prefix level to better align model learning with autoregressive decoding.

Together, these methods significantly improve the effectiveness of generative retrieval on large-scale benchmarks and establish a scalable foundation for GIR.

1.2 Decoding in Generative Information Retrieval

While relevance-aware DocID construction and prefix-oriented ranking optimization substantially improve the effectiveness of GIR, they do not remove the decoding bottleneck. The core challenge arises from the autoregressive nature of DocID generation [Tay et al., 2022, Zeng et al., 2024].

During inference, constrained beam search [Tay et al., 2022, Zeng et al., 2024, Wang et al., 2022] is used to decode the top-K DocIDs. At each decoding step, prefixes are expanded according to next-token probabilities, and low-scoring prefixes are pruned. Unlike natural language generation, where multiple alternative sequences may be semantically acceptable, retrieval admits no such redundancy. Each relevant document corresponds to exactly one identifier. If the prefix of that identifier is pruned during beam search, the document cannot be recovered.

Retrieval effectiveness is highly sensitive to beam size [Zeng et al., 2024]. Increasing the beam size reduces the risk of pruning relevant prefixes and improves ranking quality. However, larger beam sizes require substantially more forward passes through the generative model, leading to notable increases in query latency. Even with large beams, autoregressive decoding may still fail to match brute-force scoring where all documents are evaluated. This leads to an effectiveness–latency dilemma: small beams make inference fast but sacrifice ranking quality, while large beams improve quality only by making retrieval much slower.

To address this limitation, Chapter 4 revisits the interaction between decoding and

representation design. Instead of relying only on local next-token scores over sequential DocIDs, we introduce PAG, a hybrid decoding framework that brings lexical-style document scoring into GIR. The key idea is to estimate which documents are promising before the model commits to a sequence of DocID tokens. Inspired by the bag-of-words assumption in classical retrieval models, PAG augments each document's sequential DocID with a set-based lexical identifier whose token order does not matter. The set-based identifier supports simultaneous decoding, producing an approximate document-level relevance score in one model pass. This score is then used to guide autoregressive generation of the sequential DocID, helping beam search retain prefixes that lead to promising documents instead of relying only on local next-token probabilities.

Experimental results show that integrating lexical retrieval signals into GIR improves effectiveness and reduces query latency. These findings suggest that lexical and generative retrieval can be unified within a coherent framework in which lexical signals provide global guidance and generative modeling provides compact and differentiable indexing.

1.3 Unifying Reasoning and Generative Information Retrieval

The previous sections focus on improving the effectiveness and efficiency of GIR as a standalone retrieval paradigm. Chapter 3 studies optimization strategies that improve the scalability of GIR on large-scale benchmarks, while Chapter 4 addresses the decoding bottleneck by integrating lexical retrieval signals to improve inference efficiency. Together, these advances establish GIR as a practical retrieval framework that can operate on realistic corpora containing millions of documents.

While these results demonstrate that generative retrieval can serve as a competitive search engine, modern LLM systems increasingly rely on retrieval in a different role. In many knowledge-intensive tasks, LLM-based search agents interact with external search systems during problem solving [Trivedi et al., 2023a, Jin et al., 2025, Li et al., 2025b]. Instead of issuing a single query and retrieving documents once, the agent generates queries

iteratively, observes retrieved information, and updates its reasoning before producing a final answer. This multi-step search-and-reasoning loop is often referred to as *agentic search*. In such settings, the retrieval system functions as an interactive tool that supports multi-step reasoning and information gathering.

Most existing approaches treat the retrieval component in such systems as a fixed module and focus primarily on optimizing the LLM (the main agent) [Jin et al., 2025, Li et al., 2025a, Sun et al., 2025a]. In these pipelines, the search engine is typically pre-trained on retrieval datasets and remains unchanged during interaction with the agent. As a result, the retrieval system is optimized for general document relevance rather than for the downstream objective of the agent. This mismatch can lead to suboptimal behavior in agentic search tasks: at a given reasoning step, the agent may issue a sub-query that targets the information needed next, but an imperfect fixed retriever may fail to surface the evidence that would support that step.

These observations motivate a different perspective in which the search engine is treated as a trainable agent that can be optimized jointly with the main agent. Through repeated interactions between the two agents, the system generates diverse multi-turn exploration trajectories that provide rich training signals for both components.

Chapter 5 studies this setting by jointly optimizing a reasoning agent and a generative retrieval component for agentic search tasks. The main agent performs multi-turn reasoning and query generation in a ReAct-style loop [Yao et al., 2023b] before producing the final answer. For each issued sub-query, the retrieval component returns documents that serve as observations for the main agent to continue its reasoning process. The resulting interaction trajectories provide training signals for both reasoning and retrieval. The main agent is trained on the correctness of the final answer, while the retrieval component is optimized using signals derived from both trajectory success and document ranking quality. Through this joint optimization process, retrieval is no longer treated as a fixed tool, but as an adaptive component of the reasoning system. On seven QA benchmarks spanning single-hop and

multi-hop questions, the most direct comparison is with a Fixed Ranker baseline that uses the same generative ranker but keeps it frozen during RL training. Joint optimization improves average token-level F1 from 0.553 to 0.568 with a 7B agent and from 0.460 to 0.471 with a 3B agent, showing the benefit of adapting the retrieval component during agent training.

The broader goal of this dissertation is to make GIR useful under the conditions in which retrieval systems are actually used in the real world. This begins with standalone retrieval. A generative retriever must search over large corpora and return results fast enough for query-time use. Chapters 3 and 4 address these two practical constraints: first by improving DocID construction and optimization so that GIR becomes effective on large-scale IR benchmarks, and then by reducing its dependence on large beam sizes so that decoding becomes substantially faster. Yet real information seeking is often not finished once a ranked list is returned. For many complex questions, the user ultimately wants an answer, a synthesis, or a report, while the intermediate search steps should be handled by the system. Chapter 5 studies this broader setting by optimizing GIR as part of a search agent, where retrieval supports the intermediate evidence gathering and reasoning needed to answer the user's question.

1.4 Contributions

The main contributions of this dissertation are as follows:

1. **Relevance-aware DocID construction and prefix-oriented ranking optimization for scalable generative retrieval (Chapter 3).** We introduce RIPOR, a framework that derives document identifiers from retrieval-oriented representations via residual quantization and trains the generative model with a prefix-aware ranking loss that explicitly accounts for beam search pruning dynamics. On MSMARCO Dev, using the full MSMARCO corpus with 8.8M passages, RIPOR improves MRR@10 over

the strongest prior GIR model from 0.255 to 0.333, a 30.6% relative gain. This brings generative retrieval close to competitive dense retrievers such as TAS-B (0.344).

2. **Hybrid simultaneous-autoregressive decoding for efficient generative retrieval (Chapter 4).** We propose PAG, a framework that augments each DocID with a set-based identifier that can be decoded in a single step to approximate document-level scores, then conditions autoregressive decoding on these scores. This reduces the beam size required for competitive retrieval by 10×, yielding a 22× improvement in query latency on a single A100 GPU while achieving a 15.6% relative improvement in MRR@10 over the previous best generative retrieval model. PAG also matches or outperforms strong dense retrievers while requiring 7.7× less memory for corpus indexing.

3. **Joint optimization of a reasoning agent and a generative ranker for agentic search (Chapter 5).** We propose CoSearch, a framework that jointly trains a reasoning LLM and a generative document ranker using GRPO. To enable this, we introduce a semantic grouping strategy that clusters sub-queries across rollouts to form valid GRPO groups without additional sampling, and a composite reward that combines ranking quality with trajectory-level answer correctness. On seven QA benchmarks, CoSearch achieves +6.6% relative F1 over Search-R1 with a 7B agent and +10.0% with a 3B agent, showing that co-optimizing retrieval and reasoning is a promising direction for building more capable agentic search systems.

CHAPTER 2

BACKGROUND AND RELATED WORK

Information retrieval is commonly formulated as a ranking problem. Given a query and a corpus, a retrieval system assigns relevance scores to candidate documents and returns an ordered list. Classical lexical models such as TF-IDF and BM25 estimate relevance from term matching and term statistics. Neural retrieval models instead learn representations for queries and documents, allowing relevance to be computed in a learned vector space. Many modern retrieval systems combine these ideas in a retrieve-then-rerank pipeline: a fast first-stage retriever produces candidates, and a more expressive reranker refines their order.

This chapter uses this standard retrieval pipeline as the starting point. It first reviews neural embedding models and reranking methods, which provide the main reference point for modern retrieval systems. It then turns to Generative Information Retrieval (GIR), where the model retrieves documents by generating document identifiers rather than by comparing query and document representations in an index. In this setting, DocID quality is a key factor. A DocID is not only a name for a document; it is also the target sequence the model must learn and the search path that decoding must follow. The final part reviews retrieval-augmented generation and search agents, where retrieval models are used inside LLM-based systems.

2.1 Neural Embedding Models

Neural embedding models form the foundation of modern information retrieval systems. They are widely used in applications such as recommender systems and ad hoc retrieval, and they also serve as a critical component in emerging search agents. The quality of retrieval directly affects the performance of the entire system and its downstream components. If

relevant information cannot be retrieved at the early stage, subsequent reasoning modules cannot compensate for the missing knowledge.

Embedding-based retrieval models are typically deployed in the first stage of retrieval, where the goal is to efficiently retrieve relevant candidates from collections containing millions or even billions of documents. In this setting, both efficiency and effectiveness must be considered. Although these two dimensions are often in tension, embedding-based methods provide a practical balance between them.

From the perspective of efficiency, embedding-based models pre-compute document representations and store them in an index. With the support of optimized infrastructure such as FAISS [Johnson et al., 2019] and approximate nearest neighbor (ANN) search algorithms [Xiong et al., 2021a], retrieval can be performed with high throughput during inference. As a result, large-scale candidate retrieval typically does not become a system bottleneck.

In terms of effectiveness, neural embedding models have benefited from large-scale language-model pre-training and supervised retrieval data. Transformer-based encoders acquire broad textual representations during pre-training [Devlin et al., 2019], and datasets such as MS MARCO [Campos et al., 2016] and Natural Questions [Kwiatkowski et al., 2019] provide direct supervision for query-document matching. This enables embedding models to capture semantic relatedness beyond exact term overlap [Xiong et al., 2021b], model diverse query intents [Zeng et al., 2023], and reasoning-oriented matching [SU et al., 2025]. This progress builds on a long line of classical retrieval work that made lexical evidence effective at scale. Term weighting and probabilistic retrieval models, including TF-IDF and BM25 [Sparck Jones, 1988, Robertson et al., 1994], use exact query-term evidence and collection statistics to estimate relevance. Stemming [Porter, 1997], query expansion and relevance feedback [Lavrenko and Croft, 2001], and language-modeling approaches with probabilistic smoothing [Ponte and Croft, 1998, Lafferty and Zhai, 2001, Zhai and Lafferty, 2004] further reduce different forms of mismatch between queries and

documents. Neural embedding models extend this tradition by learning query and document representations from data, providing a complementary way to model semantic relatedness while preserving the efficiency required for first-stage retrieval.

Embedding-based retrieval models can be broadly categorized into two types. The first is dense embedding models, including single-vector and multi-vector approaches [Xiong et al., 2021b, Karpukhin et al., 2020c, Khattab and Zaharia, 2020]. The second is sparse embedding models, which represent texts using high-dimensional sparse vectors aligned with the vocabulary space [Formal et al., 2021b, Zeng et al., 2025].

2.1.1 Dense Retrieval

Dense retrieval, also known as dual-encoder or bi-encoder retrieval, encodes queries and documents independently into a shared low-dimensional embedding space using pre-trained large language models, such as BERT or more advanced models including Qwen [Yang et al., 2025] and LLaMA [Dubey et al., 2024].

Dense retrieval systems are fine-tuned using supervised retrieval data, which typically consists of a query paired with positive documents and a set of negative documents. The training objectives can include pairwise losses [Xiong et al., 2021b], listwise losses [Khattab and Zaharia, 2020], or contrastive losses [Gao and Callan, 2022]. Contrastive loss can be viewed as a special case of listwise loss. Due to its strong empirical performance and ease of implementation, it has become the most widely adopted objective in practice.

When teacher models are available, dense retrievers can be further improved through knowledge distillation. Teacher models may include cross-encoder rankers [Zeng et al., 2022], larger dense retrieval models, or complementary retrieval systems such as sparse retrievers [Zhang et al., 2022]. These teachers provide soft supervision signals to guide the training of the student retriever. Prior work shows that knowledge distillation can substantially improve small dense models, enabling them to approach the performance of large models or cross-encoder rerankers on in-domain benchmarks [Hofstätter et al., 2021,

Zeng et al., 2022]. However, it has also been observed that performance gains obtained through distillation on in-domain data do not always transfer well to out-of-domain or zero-shot retrieval settings [Zeng et al., 2025].

Formally, single-vector dense retrieval can be defined as

$$\text{score}(q, d) = \text{sim}(E_\psi(q), E_\phi(d)), \tag{1}$$

where E_ψ and E_ϕ are the query and document encoders, respectively, and sim is commonly implemented as inner product.

In this formulation, queries and documents do not interact before encoding. Relevance is computed only after both are projected into the embedding space. This architectural property allows document embeddings to be pre-computed and stored in an index. During inference, ANN search can be used to efficiently retrieve the top-ranked documents for each query.

Despite their efficiency and strong empirical performance, single-vector dense retrieval models exhibit representational limitations. Certain queries may not be well represented by a single embedding vector. Multi-vector dense retrieval models such as ColBERT [Khattab and Zaharia, 2020] are an alternative with more representation power. ColBERT represents queries and documents using multiple token-level embeddings and computes relevance through a late interaction mechanism based on a max-sum operator. This operator still allows document-side embeddings to be pre-indexed, preserving efficient retrieval. Still, multi-vector models require substantially more memory, incur higher computational cost, and introduce additional implementation complexity than single-vector models.

2.1.2 Sparse Retrieval

Sparse retrieval models originate from classical lexical retrieval methods such as TF-IDF [Sparck Jones, 1988] and BM25 [Robertson and Zaragoza, 2009, Robertson et al., 1994]. The underlying principle of lexical retrieval is that relevance between a query and

a document can be estimated by measuring the overlap between query terms and important terms in the document.

Before the emergence of BERT-based models, BM25 was widely considered the state-of-the-art retrieval method. However, lexical retrieval models suffer from the semantic mismatch problem. For example, semantically related terms such as "vehicle" and "car" are treated as independent tokens in vanilla BM25, leading to zero overlap and therefore zero contribution to the relevance score. Although techniques such as pseudo relevance feedback (PRF) [Lavrenko and Croft, 2001, Abdul-Jaleel et al., 2004, Lv and Zhai, 2009] have been proposed to mitigate this issue, these approaches often rely on strong assumptions or introduce additional complexity, and they do not fully resolve the fundamental limitations of lexical matching.

Earlier work on SNRM [Zamani et al., 2018] showed that a neural network can learn high-dimensional sparse representations for queries and documents, and that the sparsity of these representations makes them compatible with inverted-index retrieval. SPLADE extends this idea by deriving sparse representations from transformer token logits [Formal et al., 2021b]. In SPLADE, documents and queries are encoded using a transformer model, and the output token logits are transformed and aggregated into a single vocabulary-aligned vector. A log-based transformation stabilizes the magnitude of the logits, and max pooling produces a sparse representation whose dimensionality typically matches the vocabulary size of the underlying language model. Each dimension therefore stores the weight of a specific token.

To ensure efficiency, sparsity is enforced using regularization techniques such as ℓ_1 regularization [Zamani et al., 2018] or FLOPs-based regularizers [Formal et al., 2021b]. As a result, the high-dimensional representation becomes highly sparse. During inference, retrieval can be performed using an inverted index [Formal et al., 2021b], leveraging decades of optimization in index-based search systems while maintaining high efficiency.

Because the sparse representations are learned from supervised training data and operate

at the subtoken level, they alleviate the semantic mismatch issue more effectively than traditional lexical retrieval. Compared to dense retrieval, neural sparse retrieval achieves comparable or even superior performance in IR benchmarks. In particular, sparse models often demonstrate stronger zero-shot generalization [Zeng et al., 2025] and provide better interpretability due to their explicit token-level weighting mechanism.

2.1.3 Fundamental Components for Improving Embedding Models

Neural embedding models are commonly trained with supervised contrastive learning [Gao and Callan, 2022, Ma et al., 2024]. The objective encourages query representations to be close to relevant documents while separating them from non-relevant ones in the embedding space. A key factor that determines the effectiveness of contrastive training is the choice of negative examples.

Prior work shows that hard negatives [Xiong et al., 2021b, Zhan et al., 2021, Zeng et al., 2022], especially those retrieved by the model itself, substantially improve retrieval performance. Yet, relevance annotations in large-scale retrieval benchmarks are usually incomplete. Many relevant documents remain unlabeled due to annotation cost and coverage limitations. As a result, some sampled negatives may in fact be relevant, leading to the false negative problem. This issue can introduce biased gradients and hinder model optimization. To mitigate this effect, Qu et al. [2020] attempt to improve negative sampling by filtering potentially mislabeled negatives using stronger ranking models or more reliable relevance estimators, or by using larger in-batch negatives.

Knowledge distillation (KD) [Ren et al., 2021, Hofstätter et al., 2020, 2021, Zeng et al., 2022, Hinton et al., 2015] has also become a standard technique for improving embedding models. In this setting, a stronger teacher model, such as a cross-encoder reranker or a larger retrieval model, provides supervision to a more efficient student retriever. Instead of relying solely on human-annotated relevance labels, KD transfers soft relevance signals produced by the teacher. This partially alleviates the incomplete labeling problem and

enables the student model to benefit from the teacher relevance distribution. Common objectives include minimizing KL divergence between their score distributions [Santhanam et al., 2022, Zeng et al., 2025], as well as pairwise or listwise ranking losses that approximate the teacher's ranking behavior.

Beyond training objectives, strengthening the representational capacity of retrievers through mid-training or pre-training has shown promising results. One line of work creates large-scale pseudo query–document pairs to mid-train embedding models with retrieval-oriented supervision [Gao and Callan, 2022, Shen et al., 2023]. Another direction introduces reconstruction-based objectives, where a single embedding is trained to recover the original text, encouraging it to encode richer semantic information. A third approach adapts decoder-only language models into embedding-based retrieval models with lightweight architectural changes, such as replacing causal attention with bidirectional attention. Recent work in this direction [BehnamGhader et al., 2024, Zeng et al., 2025] shows that retrieval models can benefit from the scale and representation strength of large pretrained language models.

Taken together, embedding-based methods define the practical reference point for this dissertation. They are effective, efficient, and supported by mature indexing infrastructure. GIR must therefore offer more than a different formulation; it must show that generation can support retrieval at realistic scale and with competitive efficiency.

2.2 Generative Retrieval Models

Generative Information Retrieval (GIR) departs from the conventional "index-then-retrieve" paradigm adopted by sparse and dense retrieval models. Instead of encoding documents into embedding spaces and performing ANN search, GIR reformulates retrieval as a sequence generation problem. Each document is assigned a unique document identifier (DocID), and a sequence-to-sequence model is trained to directly generate relevant DocIDs conditioned on a query.

This formulation eliminates the need for explicit similarity computation over an index

during inference and enables retrieval to be modeled entirely within a generative framework. It also makes DocID design one of the core problems in GIR. Good identifiers preserve useful structure from the corpus, so that related documents can share generation cues. Poorly organized identifiers instead force the model to learn retrieval through labels that are difficult to predict and difficult to search.

2.2.1 Document Identifier Construction

Because DocIDs define the output space of the model, their construction directly affects retrieval performance. In most existing approaches, DocIDs are fixed during fine-tuning, thereby serving as a representational bottleneck that determines how easily the model can associate a query with the right documents.

Existing DocID construction strategies can be broadly categorized into two groups:

Semantic-based DocIDs. They are constructed by organizing documents according to their semantic representations. Typical approaches apply quantization [Zhou et al., 2022, Rajput et al., 2023, Zeng et al., 2024, Chen et al., 2023a] or hierarchical clustering [Tay et al., 2022, Mehta et al., 2023, Wang et al., 2022, Sun et al., 2023a, Jin et al., 2024] on document embeddings. For example, hierarchical clustering methods build a tree structure over document representations, and each document is assigned a root-to-leaf path encoded as a sequence of discrete tokens. Such identifiers aim to preserve semantic locality, so that documents with similar meanings share common prefix structures.

Token-based DocIDs. They are directly derived from document content, which may incorporate document titles [Chen et al., 2022a,b, Lee et al., 2022, De Cao et al., 2021], n-grams [Bevilacqua et al., 2022, Li et al., 2023b,a, Wang et al., 2023, Chen et al., 2023b], URLs [Zhou et al., 2022, Ren et al., 2023], or salient terms [Zhang et al., 2024]. Compared to semantic-based identifiers, token-based DocIDs often preserve lexical signals but may lack global structural organization across the corpus.

2.2.2 Training Objectives

Early generative retrieval models [Tay et al., 2022, Mehta et al., 2023, Bevilacqua et al., 2022, Wang et al., 2022] optimize the sequence-to-sequence model using cross-entropy loss to predict the target DocID tokens. Under this formulation, retrieval is treated purely as conditional language modeling.

Subsequent research demonstrates that ranking-oriented objectives can further improve effectiveness. For instance, learning-to-rank losses [Li et al., 2023a, Zeng et al., 2024] are introduced to better align the generation process with retrieval evaluation metrics. Also, data augmentation strategies, such as generating pseudo queries [Zhou et al., 2022, Zhuang et al., 2022b, Wang et al., 2022], have been shown to mitigate distribution mismatch between index construction and query-time generation.

2.2.3 Inference and Search Algorithms

At inference time, generative retrieval requires constrained decoding to generate valid DocIDs. Methods such as constrained beam search or FM-index-based decoding [Tay et al., 2022, Zeng et al., 2024, Bevilacqua et al., 2022, Li et al., 2023b] are employed to restrict the output space to valid identifiers. The decoding strategy plays a critical role in balancing efficiency and effectiveness, particularly when the identifier space is large.

2.2.4 Scalability and Limitations

While generative retrieval has promising performance on relatively small-scale datasets, such as NQ-320K [Tay et al., 2022] and MSMARCO-100K [Pradeep et al., 2023a], its scalability to large collections remains challenging. Recent studies report performance degradation when scaling to full-sized benchmarks [Pradeep et al., 2023a], raising concerns regarding identifier bottlenecks and decoding inefficiency.

RIPOR [Zeng et al., 2024] addresses some of these limitations by introducing relevance-based DocID and prefix-oriented ranking optimization. RIPOR demonstrates competitive

performance against strong dense retrieval baselines on the full MSMARCO benchmark with 8.8 million passages.

Despite these advances, generative retrieval still faces open challenges in scalability, decoding efficiency, and representation flexibility. These challenges motivate the methods proposed in the subsequent chapters. Chapter 3 focuses on identifier construction and training objectives, while Chapter 4 focuses on the decoding procedure that turns model scores into ranked retrieval results.

2.3 Neural Reranking Models

The embedding retrieval models reviewed earlier form the first stage of most practical search systems. Their effectiveness is tied to an efficient scoring interface: documents are encoded in advance, queries are encoded at run time, and relevance is computed through inexpensive operations such as inner products between query and document representations. This design makes large-scale search feasible. Yet it constrains the expressiveness of first-stage relevance scoring: the model must estimate relevance from independently constructed, indexable representations and a simple similarity operation. This constraint leads to a practical limitation: first-stage retrieval is efficient enough to search the full corpus, but its simplified scoring function can limit ranking accuracy and weaken zero-shot generalization across heterogeneous domains [Thakur et al., 2021].

This tradeoff motivates the multi-stage retrieval pipeline. The first-stage retriever is responsible for high-recall candidate generation over the full corpus, while a reranker applies a more expressive model to a much smaller candidate set. Cross-encoder rerankers based on BERT [Nogueira and Cho, 2019] jointly encode the query and document, allowing attention to operate directly across their tokens. This richer interaction is expensive to apply to every document, but it is effective once candidate generation has narrowed the search space. Sequence-to-sequence rankers such as monoT5 [Nogueira et al., 2020] and RankT5 [Zhuang et al., 2022a] further cast ranking as a text generation or sequence modeling

problem.

More recently, LLMs have been used as listwise rankers that consider multiple candidate documents together and produce an ordered list [Sun et al., 2023b, Pradeep et al., 2023b]. These models extend the same retrieve-then-rerank logic: the first-stage retriever supplies the candidate set, and the reranker uses a stronger model to make more fine-grained ranking decisions. This background also prepares Chapter 5, where a search agent begins from candidates returned by a fixed retriever and trains a document ranker to select evidence that improves downstream answering.

2.4 Retrieval-Augmented Generation and Search Agents

The previous sections mainly discuss retrieval models in the standard single-step setting: given a query, the system returns documents that are relevant to that query. This abstraction is central to information retrieval, and it is the setting studied by Chapters 3 and 4. However, many information needs require more than one retrieval step. Complex questions often require decomposition into sub-questions, evidence from multiple sources, and synthesis of intermediate findings before producing an answer. This need is reflected in multi-hop QA benchmarks such as HotpotQA and 2WikiMultiHopQA [Yang et al., 2018, Ho et al., 2020].

In these settings, the user's need is broader than finding a relevant document. The desired output may be a direct answer or a concise summary; when the question requires multiple steps, the system may also need relevant information for each intermediate sub-question. Retrieval therefore begins to function as a tool for reasoning: a search agent can invoke it step by step, use the returned evidence to update its state, and decide what to search for next.

2.4.1 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) captures the answer-oriented side of this shift. A traditional search system returns a ranked list of documents for a user. In RAG, retrieved documents serve as evidence for a language model, which generates a response conditioned on that evidence [Lewis et al., 2020, Guu et al., 2020, Borgeaud et al., 2022, Karpukhin et al., 2020b]. The user-facing output is therefore an answer or summary, even though the system still relies on document retrieval internally.

RAG changes what retrieval is used for, but it does not necessarily change the search process itself. Standard RAG pipelines usually retrieve a fixed set of documents for the original question and then generate from that fixed context. As a result, standard RAG has limited ability to handle multi-step questions: it can condition generation on retrieved evidence, but it does not by itself decide how to decompose the question, when to search again, or what information is needed at a later step [Jiang et al., 2023].

2.4.2 Search Agents and Reinforcement Learning

Search agents address this limitation by making retrieval an action inside a multi-step reasoning process. Instead of retrieving once and then answering, a search agent alternates between reasoning, issuing queries, reading observations, and deciding what to do next. Methods such as IRCoT [Trivedi et al., 2023b] and ReAct [Yao et al., 2023a] show that interleaving reasoning with search can help models solve questions that require multiple evidence-gathering steps. Work on self-reflective retrieval and iterative query generation further develops this view of retrieval as an interactive tool rather than a static preprocessing step [Asai et al., 2024, Shi et al., 2024b].

This interaction loop is often referred to as *agentic search*. It becomes useful when the original question is too complex to be answered by a single retrieval call. The agent may need to decompose the question into smaller information needs, some of which can be pursued independently and others that depend on evidence found in earlier steps. Compared

with a one-shot retrieval pipeline, agentic search gives the system greater flexibility: it can decide when to search, what to search for, and how to use each observation as the reasoning process unfolds. Each step can refine the agent's intermediate state and guide the next search until the system has gathered enough evidence to answer the user's information need.

Training search agents has become an important direction because the desired behavior is complex, sequential, and difficult to supervise directly. A successful trajectory may involve long reasoning chains and many tool interactions, making it expensive to collect human labels for every intermediate search decision. Reinforcement learning [Kaelbling et al., 1996, Sutton et al., 1999] provides a natural way to optimize such behavior from outcome-level signals, such as final-answer correctness, without token-level annotations for each intermediate decision [Jin et al., 2025, DeepSeek-AI et al., 2025]. Recent RL-based systems use verifiable rewards to teach search agents when to search and what to search for [Jin et al., 2025, Li et al., 2025a,b, Sun et al., 2025b, Zeng et al., 2026]. Related tool-use frameworks similarly train LLMs to decide when and how to invoke external tools [Schick et al., 2023, Li et al., 2025c, Feng et al., 2026, Wang et al., 2025].

Most of these systems, however, focus on training the reasoning agent while leaving the retrieval system fixed. A separate line of work adapts retrievers or rankers for downstream generation, including LM-supervised retriever training [Guu et al., 2020, Shi et al., 2024a, Salemi and Zamani, 2024], LLM-based listwise ranking [Sun et al., 2023b, Pradeep et al., 2023b], and retrieval training for agentic search [Liu et al., 2026]. These directions show that retrieval can be shaped by downstream use, but many agentic systems still assume that the search engine or first-stage retriever is a fixed environment.

These developments motivate the final technical step in this dissertation. If retrieval is repeatedly invoked by a search agent, retrieval quality should be judged not only by standalone relevance but also by whether the returned evidence helps the agent solve the task. Chapter 5 studies this setting by jointly optimizing the reasoning agent and the retrieval component it uses.

CHAPTER 3

OPTIMIZATION IN GENERATIVE INFORMATION RETRIEVAL

Generative information retrieval (GIR) has recently emerged as a new paradigm that differs from embedding-based neural retrieval models. Instead of relying on an external index built from dense or sparse document representations, GIR assigns each document a unique identifier (DocID) and trains a sequence-to-sequence model to generate relevant DocIDs autoregressively for a given query.

Intuitively, a DocID is a compact, generatable representation of a document. It turns a document into a short sequence of discrete tokens that a sequence-to-sequence model can produce one token at a time. The design of this sequence matters: the tokens should retain information useful for identifying the document, and the prefixes should organize related documents so that left-to-right generation can narrow the search space step by step. For semantic DocIDs, this organization can be understood as a coarse-to-fine structure over the collection: each generated token moves the model to a smaller region of documents, and a useful identifier space places related documents on nearby paths. Figure 1 illustrates this intuition with three length-three DocIDs. The two footwear documents, *Nike football shoes* and *Adidas basketball shoes*, share the prefix 4-2, while the unrelated *rice cooker* document follows a separate path, 1-7-3.

This prefix structure helps decoding by turning document selection into a coarse-to-fine search. Once the model commits to a prefix such as 4-2, both footwear documents remain reachable, and the final token only needs to distinguish between them. At the same time, unrelated regions of the collection can be pruned early. The figure shows only three length-three paths for illustration; in practice, a DocID space may contain millions or billions of valid paths, and each DocID may contain four, eight, or more tokens. The goal is still to make related documents easier to reach through related generation paths.

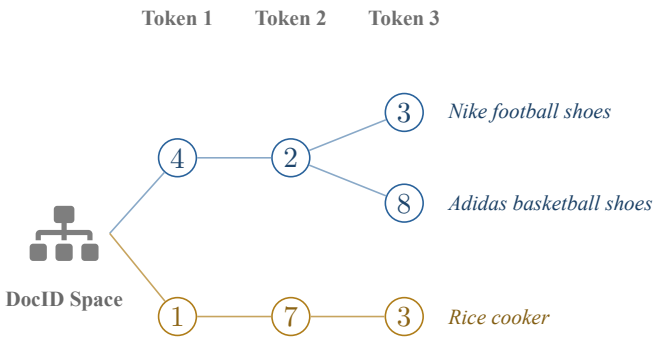


Figure 1: A length-three semantic DocID example. Related documents can share a prefix and diverge only at later tokens, while unrelated documents follow different generation paths.

Existing DocID designs instantiate this idea in two broad ways. Semantic-based DocIDs first place documents in a learned representation space and then convert that space into identifiers through hierarchical clustering or quantization, so documents close in that space tend to receive related token sequences [Tay et al., 2022, Mehta et al., 2023, Wang et al., 2022, Zhou et al., 2022, Rajput et al., 2023]. Token-based DocIDs derive the sequence from observable document content, such as titles, n-grams, URLs, or salient terms, preserving lexical cues that can anchor generation and search [Bevilacqua et al., 2022, Li et al., 2023b,a, Wang et al., 2023, Zhang et al., 2024].

Tay et al. [2022] introduced the Differentiable Search Index (DSI), the pioneering work showing that a generative model can retrieve by generating document identifiers. This result established the basic GIR formulation: assign documents discrete identifiers, train a sequence-to-sequence model to generate relevant identifiers from queries, and map the generated identifiers back to documents. Later work extends this framework with new DocIDs, pseudo-query augmentation, and retrieval-oriented training [Wang et al., 2022, Zhuang et al., 2022b, Mehta et al., 2023].

Despite its conceptual elegance, GIR still faces clear scalability limits on large retrieval benchmarks. Prior studies show that GIR often trails strong dense retrievers and, in some cases, even classic lexical models such as BM25 [Pradeep et al., 2023a]. On the full MSMARCO passage corpus with 8.8 million passages, for example, NCI reaches only about

half the MRR@10 of competitive dense retrieval systems [Pradeep et al., 2023a]. Scaling model size and augmenting training data with synthesized pseudo-queries partially reduce this gap. However, these strategies do not fundamentally address the underlying limitations of the generative retrieval paradigm. On the MSMARCO Dev set, a 110M-parameter BERT-based dense retriever achieves MRR@10 above 0.333, whereas a substantially larger T5-3B generative retrieval model attains only 0.267 MRR@10 [Pradeep et al., 2023a]. This persistent performance gap calls into question the practicality of GIR for large-scale, real-world retrieval scenarios.

This chapter studies the first practical obstacle for GIR: whether a generative retriever can remain effective when the corpus grows to millions of documents. We introduce RIPOR, a framework designed to improve the scalability of generative retrieval on large-scale IR benchmarks. The main idea is to align both what the model generates and how it is trained with the retrieval task. First, we propose a prefix-oriented ranking optimization that aligns the training objective with the autoregressive decoding process used at inference time. Since constrained beam search prunes candidates based on prefix scores, relevant documents can only survive decoding if their intermediate prefixes consistently receive higher scores than competing prefixes. Our optimization explicitly models this requirement, thereby reducing early pruning errors. Second, we introduce a relevance-based DocID construction strategy that derives document identifiers from retrieval-oriented representations learned on the target task. These representations are decomposed using residual quantization (RQ), producing structured DocIDs that better reflect hierarchical relevance patterns. Experiments demonstrate that both the prefix-aware optimization and the relevance-driven identifier construction are critical to achieving strong performance. Together, these components enable RIPOR to substantially narrow the performance gap between generative and dense retrieval models on large-scale benchmarks.

3.1 Introduction to Generative IR

In generative document retrieval, each document is symbolized by a unique identifier, known as document ID or *DocID* for short. Pre-trained encoder-decoder models, such as T5 [Raffel et al., 2019], are employed to generate a list of document IDs in response to a given query. Let M represent a generative retrieval model that represents a document d using the document ID $c_d = [c_1^d, c_2^d, \ldots, c_L^d]$ of length L. Various methods are applied to the DocID construction [Tay et al., 2022, Bevilacqua et al., 2022, Zhou et al., 2022]. For instance, DSI employs the hierarchical k-means over the document embeddings obtained from the pre-trained BERT model [Devlin et al., 2019]. Once the tree is built, each root-to-leaf path is used as a unique document ID [Tay et al., 2022].

As depicted in Figure 2, M is trained to generate document IDs autoregressively for any given query q, meaning that it generates each DocID token c_i^d conditioned on previously generated tokens, denoted by $c_{<i}^d$. Therefore, the model generates a conditional hidden representation for the i^{th} DocID token as follows:

$$\mathbf{h}_i^d = \text{Decoder}(c_{<i}^d; \text{Encoder}(q)) \in \mathbb{R}^D.$$

where $c_{<i}^d = [c_1^d, c_2^d, \ldots, c_{i-1}^d]$ is fed to the decoder as its input and the encoded query vector is used to compute cross-attentions to the decoder. In generative retrieval, each DocID token is associated with a D-dimensional representation. Let $\mathbf{E}_i \in \mathbb{R}^{V \times D}$ denotes a token embedding table for each position i in the DocID sequence, where V is the vocabulary size for DocID tokens, i.e., the number of distinct tokens for representing document IDs. Therefore, the representation associated with each DocID token c_i^d is represented as $\mathbf{E}_i[c_i^d] \in \mathbb{R}^D$. Note that the DocID token embedding matrices are distinct, thus $\mathbf{E}_i \neq \mathbf{E}_j : \forall i \neq j$.

Inspired by seq2seq models[Raffel et al., 2019, Chung et al., 2024, Ouyang et al., 2022], existing generative retrieval models estimate relevance scores based on $\log$-conditional

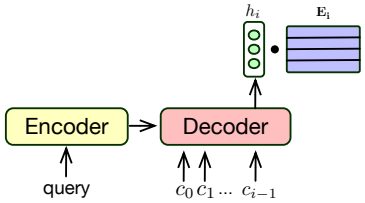


Figure 2: An illustration of generative retrieval models.

probability as follows:

$$
\begin{aligned}
S(q, c_d) &= \log p([c_1^d, c_2^d, \ldots, c_L^d]|q) \\
&= \sum_{i=1}^{L} \log p(c_i^d|q, c_{<i}^d) \\
&= \sum_{i=1}^{L} \left[\text{LogSoftmax}(\mathbf{E_i} \cdot \mathbf{h_i^d})[c_i^d]\right]
\end{aligned}
$$

where $S(q, c_d)$ denotes the scoring function for a query-document pair. In this chapter, we instead adopt a conditional logit approach, due to its less expensive computation cost and better alignment with our margin-based pairwise loss. We will further elaborate this choice in Section 3.2.1. This approach is inspired by dense retrieval models that use dot product similarity between query and document representations, and computes dot product similarity between the token embedding vectors corresponding to the DocID and the hidden vectors learned for each decoding position given the query and past decodings. In more detail, this approach can be formulated as follows:

$$
\begin{aligned}
S(q, c_d) &= \text{concat}(\mathbf{E_1}[c_1^d], \ldots, \mathbf{E_L}[c_L^d]) \cdot \text{concat}(\mathbf{h}_1^d, \ldots, \mathbf{h}_L^d) \\
&= \sum_{i=1}^{L} \mathbf{E_i}[c_i^d] \cdot \mathbf{h_i^d}.
\end{aligned}
$$

Employing these scoring functions, generative retrieval models produce a ranked list of documents using *beam search* with constrained decoding [De Cao et al., 2021], where the top K valid DocIDs are generated according to the scoring function. Each of the DocIDs

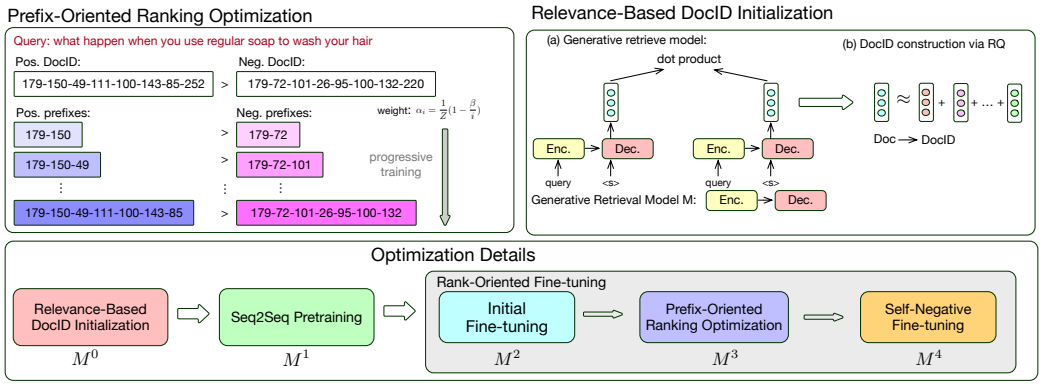


Figure 3: An overview of the RIPOR framework. The top two sub-figures illustrate the novel components in RIPOR framework, detailed in Sections 3.2.1 and 3.2.2. The bottom sub-figure presents the complete optimization pipeline.

is then mapped back to its original document. This results in a ranked list of K documents.

3.2 Methodology

The high-level overview of the RIPOR framework is illustrated in Figure 3. RIPOR is built around two intuitions. First, DocIDs should be initialized from retrieval-oriented representations, because the similarity that matters for retrieval is often query-dependent rather than merely linguistic. Second, training should account for prefix survival, because constrained beam search may falsely prune relevant DocIDs when their prefixes receive low scores. Following these intuitions, the generative model M is first viewed as a dense encoder and fine-tuned with a relevance-based objective. RIPOR then employs Residual Quantization (RQ) [Chen et al., 2010] to derive a unique identifier for each document. Subsequently, following Wang et al. [2022], Zhuang et al. [2022b], Pradeep et al. [2023a], we leverage a seq2seq pre-training method using pseudo queries from the documents. Next, we introduce rank-oriented fine-tuning to refine the parameters of model M. In the next two sections, we describe the two major components in RIPOR: prefix-oriented ranking optimization and relevance-based document ID construction. A detailed description of the entire optimization pipeline is presented in Section 3.2.3.

3.2.1 Prefix-Oriented Ranking Optimization

State-of-the-art generative retrieval models, such as LTRGR [Li et al., 2023a], adopt a learning-to-rank loss for optimization. The objective is to ensure that $S(q, c_{d^+}) > S(q, c_{d^-})$ for a training triplet of query q, relevant document d^+ and irrelevant document d^-. We posit that this modeling is not optimal. A primary oversight is the intrinsic nature of beam search that *sequentially* decodes document ID tokens from left to right.

Solely focusing on pairwise ranking for a full-length document ID does not guarantee that relevant documents can survive the beam search eliminations in earlier decoding steps. Therefore, we aim to develop a model that produces accurate scores at every decoding step. Formally, we desire to satisfy the following criterion: $S^i_{\text{prefix}}(q, c_{d^+}) \geq S^i_{\text{prefix}}(q, c_{d^-}), \ \ \forall i \in [1, L]$, where $S^i_{\text{prefix}}(q, d)$ denotes the relevance score produced by the generative retrieval model for the first i tokens in the document ID: $[c^d_1, c^d_2, \ldots, c^d_i]$.

Margin Decomposed Pairwise Loss: Taking inspiration from MarginMSE [Hofstätter et al., 2020], we use a pairwise loss for knowledge distillation as follows:

$$\mathcal{L}(q, d^+, d^-) = \left(S(q, d^+) - S(q, d^-) - T_{(q,d^+,d^-)}\right)^2,$$

where $T_{(q,d^+,d^-)}$ denotes the golden margin, commonly predicted by a teacher model derived from a cross-encoder [Nogueira and Cho, 2019]. Prior research [Hofstätter et al., 2020, Zeng et al., 2022] reveals that this loss function often outperforms other pairwise losses [Xiong et al., 2021a] by addressing data sparsity issues in large-scale retrieval benchmark [Qu et al., 2020], utilizing pseudo-labels for unlabeled query-document pairs.

For generative retrieval, we extend the MarginMSE loss by modeling pairwise ranking between prefixes of c_{d^+} and c_{d^-} for each decoding step i:

$$\mathcal{L}^i_{\text{rank}}(q, c_{d^+}, c_{d^-}) = \left(S^i_{\text{prefix}}(q, c_{d^+}) - S^i_{\text{prefix}}(q, c_{d^-}) - \alpha_i T_{(q,d^+,d^-)}\right)^2. \tag{2}$$

Here, at each step i we re-weight the golden margin by multiplying with α_i, which is a weight we assign to each prefix position. The reason for this decision is that we emphasize on the early decoding steps of the document IDs. With this motivation, α_i should be a monotonically increasing concave function w.r.t. i. Formally, α_i values should satisfy the following constraint: $\alpha_i - \alpha_{i-1} \geq \alpha_{i+1} - \alpha_i$ for every i. In our experiments, we use $\alpha_i = \frac{1}{Z}(1-\frac{\beta}{i})$, where $Z = 1-\frac{\beta}{L}$ is a normalization factor and β is a constant hyper-parameter. We leave the exploration of other concave functions to future work. For efficiency reasons, we only do prefix-oriented optimization for $i = 4, 8, 16, 32$ and thus set $\beta = 2$. This concave formulation of α_i emphasizes larger sub-margins in early steps, ensuring for any query q that $S^i_{\text{prefix}}(q, c_{d^+})$ surpasses $S^i_{\text{prefix}}(q, c_{d^-})$. Moreover, as $\alpha_L = 1$, the predicted margin for full-length DocID sequences aligns with the real margin, maintaining the fidelity of ranking knowledge.

Progressive Training: To better learn representations aligned with the left-to-right decoding characteristic of the beam search, we draw inspiration from curriculum learning [Zeng et al., 2022, MacAvaney et al., 2020, Bengio et al., 2009, Matiisen et al., 2017] and implement a progressive training strategy. The training process is initialized with the shortest prefix. This allows the model to first focus on basic sequence representations and build adequate capacity for the subsequent stages. As the training advances, the scope is systematically extended to the longer prefixes, culminating in training on the full-length sequence with length L.

During training on longer prefixes, we empirically found that the model tends to overlook previously acquired knowledge related to shorter prefixes. To mitigate this catastrophic forgetting issue, we employ multi-objective learning at each time step to ensure the retention of knowledge from earlier stages. Given the training data $\mathcal{D} = \{(q_j, d_j^+, d_j^-, T_{(q_j, d_j^+, d_j^-)})\}_{j=1}^{|\mathcal{D}|}$,

we use the following multi-objective loss function:

$$\sum_{(q,d_j^+,d_j^-)\in\mathcal{D}} \Big(\underbrace{\mathcal{L}^i_{\text{rank}}(q, d_j^+, d_j^-)}_{(1)} + \underbrace{\sum_{k=1}^{i-1} \mathcal{L}^k_{\text{rank}}(q, d_j^+, d_j^-)}_{(2)} \Big)$$

In this loss function, term (1) is responsible for acquiring the pairwise rankings specific to the current step i, while term (2) ensures the model retains the ranking knowledge from previous prefixes. As mentioned earlier, for efficiency reasons, without loss of generality, we only repeat this training process for $i = 4, 8, 16, 32$.

3.2.2 Relevance-Based DocID Construction

Generative retrieval models predominantly adopt a two-step optimization approach. First, they initialize the document IDs by employing various methods such as hierarchical k-means [Tay et al., 2022, Wang et al., 2022] or discriminative textual descriptions extracted from documents [Bevilacqua et al., 2022, Li et al., 2023b,a]. In the subsequent step, they optimize the model leveraging either cross-entropy loss [Tay et al., 2022, Bevilacqua et al., 2022] or learning-to-rank loss [Li et al., 2023a], with fixed DocIDs obtained in the first step. Given that the DocIDs remain immutable in this phase, they potentially become a significant bottleneck, influencing the overall efficacy of generative retrieval models.

We argue that the design of DocIDs is crucial in two specific ways: First, it must ensure the documents with inherent similarity possess correspondingly similar DocIDs. Second, due to the characteristics of beam search for decoding in generative retrieval, these DocIDs should encapsulate a hierarchical structure. Notably, the conception of similarity in this context is nuanced; it is tied to specific queries and deviates from standard linguistic similarities in natural language processing. Addressing these challenges, we introduce a relevance-based method for initializing DocIDs. This approach is crafted to encapsulate the query-document relevance nuances and the necessary hierarchical structure, ensuring effective performance in generative retrieval tasks.

Generative retrieval model as dense encoder: To capture the documents similarities in terms of relevance, we design an optimization process inspired by dense retrieval models, but by utilizing the encoder-decoder architecture in M. Specifically, we input document content into the encoder and a special start token as input to the decoder. The document representation is then derived from the first contextualized output embedding of the decoder:

$$\mathbf{d} = \text{Decoder}(s_0; \text{Encoder}(d)) \in \mathbb{R}^D.$$

Where s_0 is the start token. Adopting a similar approach for queries, we determine their representations. To optimize model M, we employ the MarginMSE loss [Hofstätter et al., 2020] with multi-stage negative sampling introduced in Sec 3.2.3 in details.

Residual Quantization: Hierarchical k-means, used by several prior GIR models for document ID construction [Tay et al., 2022, Wang et al., 2022, Pradeep et al., 2023a, Zhuang et al., 2022b], does not explicitly minimize the distortion error between original and approximated representations. As highlighted by Ge et al. [2014], there is a notable inverse correlation between information retrieval metrics like MAP and the distortion error, particularly for large-scale datasets. Motivated by this observation, we adopt quantization-based techniques [Ge et al., 2014, Wang et al., 2014, Babenko and Lempitsky, 2014, Chen et al., 2010] explicitly designed to minimize this distortion error. Among a myriad of quantization algorithms, we select Residual Quantization (RQ) [Babenko and Lempitsky, 2014, Chen et al., 2010] due to its inherent advantages. Specifically, (1) its recursive process captures the hierarchical document structure, aligning with the beam search strategy inherent to generative retrieval, and (2) compared to methods like product quantization (PQ) [Ge et al., 2014, Wang et al., 2014], it requires a shorter length of DocID to achieve a strong performance, leading to memory and time savings during inference. Using M as our dense encoder, we calculate the representation $\mathbf{d}$ for each document d. Subsequently, employing RQ, we optimize the token embedding table $\{\mathbf{E}_i\}_{i=1}^{L}$ to determine the optimal DocID $c_d =$

$[c_1^d, \ldots, c_L^d]$ for every document d. Upon optimization, each $\mathbf{d}$ can be approximated using a sequence of token embeddings as:

$$\mathbf{d} \approx \sum_{i=1}^{L} \mathbf{E}_i[c_i^d].$$

The trained model M alongside the embedding tables $\{\mathbf{E}_i\}_{i=1}^{L}$ will serve as the initial weights for subsequent optimization phases within generative retrieval.

3.2.3 Optimization Details

Our optimization process involves three distinct phases: (1) DocID initialization (2) seq2seq Pre-training, and (3) rank-oriented Fine-tuning.

DocID Initialization: As described in Section 3.2.2, we first treat M as a dense encoder. We optimize this encoder with multi-stage hard-negative training [Xiong et al., 2021a]. In the initial stage, BM25 [Robertson and Zaragoza, 2009] retrieves the top K documents for each query, where $K = 100$ in our experiments. We train the model on these candidates using the MarginMSE loss [Hofstätter et al., 2020]. After this stage, we obtain the dense representation $\mathbf{d}$ from M for each document and use nearest-neighbor search to retrieve a new top-K set for each query. We then train the model again with the same loss on the model-retrieved candidates. Finally, we apply residual quantization (RQ) to obtain the DocID for each document. The trained model is denoted as M^0, and the embedding tables $\{\mathbf{E}_i\}_{i=1}^{L}$ will be used as the initial weights for the next phase. Seq2seq Pre-training To equip our model M with a comprehensive understanding of the corpus, we incorporate a seq2seq pre-training phase. Instead of using the document d as input and predicting its corresponding semantic tokens $[c_1^d, \ldots, c_L^d]$, we align with prior work [Wang et al., 2022] and utilize pseudo queries associated with each document as input proxies for DocID prediction. Specifically, by leveraging the doc2query model [Cheriton, 2019], we generate N_{pseudo} pseudo queries for every document. We then optimize the model using a

cross-entropy loss, with the tokens from the relevant DocIDs serving as the ground-truth labels. We denote the trained model in this phase as M^1.

Rank-oriented Fine-tuning: We optimize the model with the pairwise loss described above. Hard negatives are important for dense retrieval [Xiong et al., 2021a, Zhan et al., 2021]. Reliable teacher supervision also improves ranking quality [Hofstätter et al., 2021, 2020, Zeng et al., 2022]. We therefore use multi-stage training. At each stage, the current model mines harder negatives, which are then used to train the next stage.

Initial Fine-tuning: This stage is to warmup the generative retrieval model and provide the high-quality negative samples to the later stages. We utilize the model M^0 from Sec 3.2.3 as a dense encoder, we index each document via its embedded representation and retrieve 100 documents for each query. We use the initialized DocIDs from Sec 3.2.3 to map each retrieved documents to their corresponding DocIDs. The training data $\mathcal{D}^R$ can be constructed based on the negative samples and ground-truth query-DocID positive pairs. Unlike our approach in subsequent stages, here we use the full-sequence $\mathcal{L}^L_{rank}$ defined in Eq 2. Starting from M^1 as an initial model, after training, the model is represented as M^2.

Prefix-Oriented Ranking Optimization: Given a query q, we apply the constrained beam search on the model M^2 to retrieve 100 DocIDs, each of which is mapped back to their corresponding documents. The documents serve as an augmented source of negative samples, and we subsequently construct a training set $\mathcal{D}^B$ in a manner similar to the previous stage. The comprehensive training set for this stage consolidates data both from the nearest neighborhood search and beam search, represented as $\mathcal{D} = \mathcal{D}^R \cup \mathcal{D}^B$. To optimize the model, we utilize the progressive training described in Section 3.2.1. For each optimization step i, we employ the multi-objective loss function described in Section 3.2.1. The trained model in this section is denoted as M^3.

Self-Negative Fine-tuning: To enhance the model's effectiveness, we employ beam search on M^3 to establish a training dataset $\mathcal{D}^B_{self}$. The model is then trained on the same multi-objective loss function in the full-length setting ($i = L$), and denoted as M^4.

Table 1: Experimental results on MSMARCO and TREC Deep Learning Track Data. Highest generative retrieval performances are boldfaced. Superscript $*$ denotes statistically significant improvement compared to all generative retrieval baselines. Superscripts $^{\triangle}$ and $^{\triangledown}$ denote significantly higher and lower performance compared to RIPOR. (t-test with Bonferroni correction, p_value < 0.01). For dense retrieval models, the HNSW [Malkov and Yashunin, 2016] index is used for ANN search.

Model	MSMARCO Dev		TREC DL 2019		TREC DL 2020	
	MRR@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10
Generative Retrieval						
DSI	.045	.138	.163	.076	.150	.070
DSI-QG	.105	.292	.320	.138	.328	.120
Ultron-PQ	.117	.248	.331	.135	.342	.134
NCI-QG	.153	.352	.403	.167	.394	.159
SEAL	.127	-	-	-	-	-
MINDER	.186	.383	.506	.201	.392	.144
LTRGR	.255	.531	-	-	-	-
RIPOR (ours)	**.333***	**.562***	**.628***	**.205***	**.631***	**.191***
Sparse and Dense Retrieval Models (For Reference)						
BM25	.185$^{\triangledown}$	.381$^{\triangledown}$	.512$^{\triangledown}$	.178$^{\triangledown}$	.477$^{\triangledown}$	.164$^{\triangledown}$
DPR	.287$^{\triangledown}$	.539$^{\triangledown}$	.588$^{\triangledown}$	.195$^{\triangledown}$	.581$^{\triangledown}$	.182$^{\triangledown}$
ANCE	.301$^{\triangledown}$	.545$^{\triangledown}$	.600$^{\triangledown}$	.262$^{\triangledown}$	.587$^{\triangledown}$	.174$^{\triangledown}$
MarginMSE	.312$^{\triangledown}$	.552$^{\triangledown}$	.634$^{\triangle}$	.250$^{\triangle}$	.614$^{\triangledown}$	.193
tas-b	.323$^{\triangledown}$	.557$^{\triangledown}$	.629	.200	.633	.227$^{\triangle}$

3.3 Experiments

3.3.1 Experiments Settings

Dataset We assess our retrieval models on the MSMARCO dataset [Campos et al., 2016], comprising 8.8M passages and 532K training queries with incomplete annotations (averaging about 1.1 relevant passages per query). We evaluate our models using three datasets: (1) MSMARCO-Dev, with 7K queries and incomplete annotations, and (2, 3) TREC DL 2019 & 2020: the passage retrieval datasets used in the first and the second iterations of TREC Deep Learning Track [Craswell et al., 2019, 2021] with 43 and 54 queries, respectively. For evaluation, we report recall@10 for all datasets, as well as the official metric for each dataset: MRR@10 for MSMARCO-Dev and NDCG@10 for TREC DL 2019 and 2020.

Implementation Details We employ the pre-trained T5-base model [Raffel et al., 2019]

as the backbone for our generative retrieval model. For DocID initialization, we adopt the residual quantization (RQ) implementation from Faiss [Johnson et al., 2019]. The length of DocID L is 32 and the vocabulary size V is 256. Unlike prior work that adds an extra code for repeated DocIDs [Rajput et al., 2023], we do not do so because we found the possibility of repeated DocIDs to be extremely small. For seq2seq pre-training, the T5-large doc2query model [Cheriton, 2019] generates 10 pseudo queries for each document. We obtain the doc2query model by first train it on the MS MARCO training set using the cross entropy loss described in the doc2query work [Cheriton, 2019]. The optimization is done using Adam optimizer [Kingma and Ba, 2015], featuring linear scheduling and a warmup ratio of 4.5% of total learning steps. For DocID initialization and rank-oriented fine-tuning phases, we set the learning rate to 0.0001 with 120 epochs and batch size of 64. For seq2seq pre-training, we set the learning rate to 0.001 with $250,000$ steps and batch size of 256. We conducted all the experiments using 8 40GB A100 GPUs. The beam size for inference is 1000.

Baselines We use the state-of-the-art generative retrieval models for comparison: DSI [Tay et al., 2022], DSI-QG [Zhuang et al., 2022b], Ultron-PQ [Zhou et al., 2022], NCI-QG [Wang et al., 2022], SEAL [Bevilacqua et al., 2022], [Li et al., 2023b] and LTRGR [Li et al., 2023a]. We also select the competitive sparse and dense retrieval models for reference: BM25 [Robertson and Zaragoza, 2009], DPR [Karpukhin et al., 2020c], ANCE [Xiong et al., 2021a], MarginMSE [Hofstätter et al., 2020], tas-b [Hofstätter et al., 2021].

3.3.2 Experiment Results

Main Results The main experiments test whether the RIPOR can close the performance gap between generative retrieval and strong retrieval baselines in large-scale IR corpora. We report the performance of RIPOR and the baselines in Table 1. First, most generative retrieval models, including DSI, DSI-QG, NCI-QG, SEAL, and MINDER, consistently lag behind BM25 across all three evaluation sets. In contrast, the LTRGR model, which

Table 2: Ablation study results on MSMARCO Dev. Superscript $^{\triangledown}$ denotes significantly lower performance compared to RIPOR (t-test with Bonferroni correction, p_value < 0.01).

	MRR@10	Recall@10
-. RIPOR	.333	.562
1. w/o prefix optimization	.280$^{\triangledown}$	.475$^{\triangledown}$
2. w/o multi-objective learning	.317$^{\triangledown}$	.532$^{\triangledown}$
3. w/o self-neg. fine-tuning	.325$^{\triangledown}$	.543$^{\triangledown}$
4. w/o seq2seq pre-training	.319$^{\triangledown}$	.539$^{\triangledown}$
5. replace with sentence t5	.192$^{\triangledown}$	.287$^{\triangledown}$
6. replace with PQ	.112$^{\triangledown}$	.155$^{\triangledown}$

incorporates a learning-to-rank algorithm, manages to surpass BM25. These observations underscore the importance of integrating learning-to-rank methodologies when designing generative retrieval models. Second, RIPOR outperforms all generative retrieval baselines consistently. When compared to the top-performing baseline LTRGR, RIPOR achieves a 30.6% improvement in MRR@10 on the MSMARCO Dev set and 94% enhancement in NDCG@10 on the TREC-20 test set. Third, our RIPOR can obtain comparable results to dense retrieval models. For instance, compared to ANCE, our model achieves 16% improvement in terms of MRR@10 on MSMARCO Dev, 4.7% and 7.5% improvement on TREC'DL 19 and 20 respectively in terms of NDCG@10. [1] Additionally, we provide the experimental results on the small-scale dataset MSMARCO-1M, in line with previous work [Pradeep et al., 2023a]. These results can be found in Table 13 in Appendix A.4.

Ablation Studies We conduct a thorough ablation study on the MSMARCO dataset to investigate the impact of each component in RIPOR. We report results in Table 2.

Beginning with Row 1, we can see the significance of incorporating prefix-oriented ranking optimization. Its absence results in 19% degradation in MRR@10. Without using the approach, the model fails to ensure every prefix of relevant DocIDs receives higher scores than those of irrelevant DocIDs in response to a query. This increases the risk of discarding the relevant DocIDs in the early steps of beam search, which, in turn, negatively

[1] HNSW index might slightly impact the performance compared to DR models using a flat index [Malkov and Yashunin, 2016, Xiong et al., 2021a].

Table 3: The retrieval performance for various DocID combinations on MSMARCO Dev set.

L × V	MRR@10	Recall@10	Extra Param.(M)
32 × 256	.333	.562	6.29
16 × 512	.307	.520	6.29
8 × 1024	.306	.535	6.29
4 × 2048	.273	.493	6.29
16 × 1024	.324	.554	12.58
8 × 2048	.319	.550	12.58
4 × 4096	.291	.528	12.58

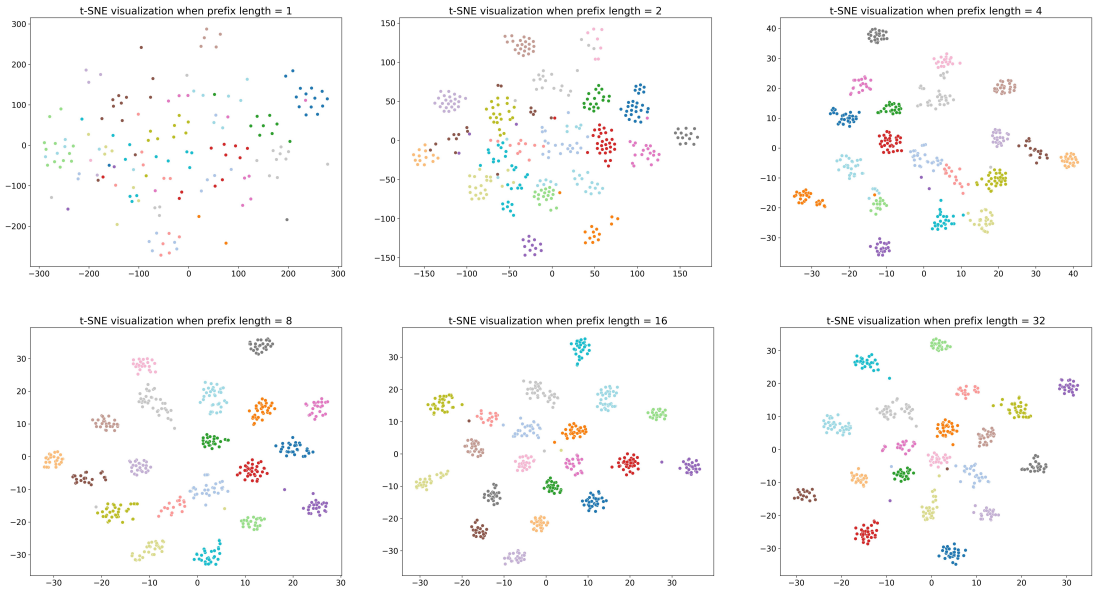


Figure 4: Clusters of the relevant documents to 20 queries sampled from TREC DL. The color indicates the query ID.

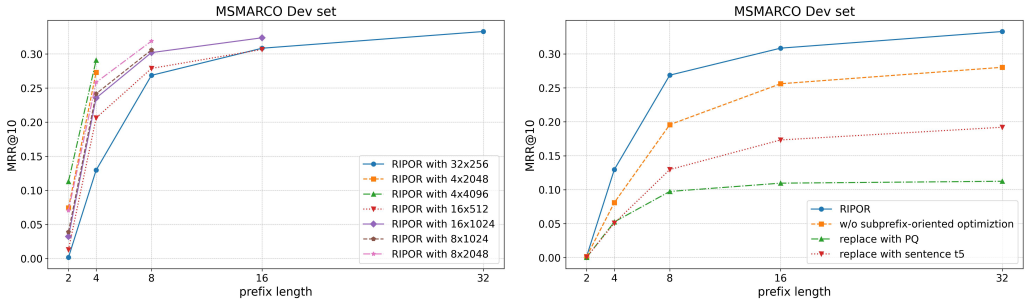


Figure 5: The retrieval performance for different prefix lengths in MSMARO Dev.

impacts information retrieval performance.

In Row 2, we infer the significance of incorporating multi-objective learning within the prefix optimization. This inclusion results in a further improvement of 5% in MRR@10. The enhancement can be credited to the approach's efficacy in mitigating the forgetting issue encountered during the progressive training's latter stages. Notably, this methodology introduces only a minimal addition to the loss computation, ensuring that there is no increase in computational overhead during training.

Row 3 reports the results for RIPOR when self-negative fine-tuning is not used in the final training stage. Incorporating this strategy yields a 2.5% enhancement in MRR@10 and a 3.5% boost in Recall@10. By strategically leveraging these hard negative samples, we bolster the model's capability, ensuring relevant DocIDs are consistently ranked higher than potential high-scoring hard negatives, which subsequently elevates the model's overall effectiveness.

From Row 4, we note that by integrating seq2seq pre-training, RIPOR achieves a 4% increase in MRR@10. This method allows the model to encapsulate document information across the entire corpus, mirroring the indexing phase in dense retrieval models, and driving the observed performance improvement.

From Row 5, when we do not treat the generative retrieval model as a dense encoder and instead use sentence-T5 [Ni et al., 2022] to derive the hidden representation for each document, substantial performance degradation occurs, with a 73% drop in MRR@10.

Finally, in Row 6, substituting RQ with PQ results in a substantial performance decline, with MRR@10 dropping from .333 to .112. While PQ is viewed as an effective quantization algorithm in the dense retrieval domain, our results imply that it is less suitable for generative retrieval. This limitation may stem from PQ's inability to encapsulate the hierarchical structure among documents, an attribute that has been shown to be crucial in generative retrieval, especially when employing beam search.

3.3.3 Analysis and Discussion

The impact of DocID combination The configuration of the document identifier (DocID), specifically its length L and vocabulary size V, influences the effectiveness of model M. We examine this relationship by evaluating various performance metrics on the MSMARCO Dev set, as detailed in Table 3. Firstly, when holding the extra parameters constant (quantified by $L \times V \times D$), we observe that the increase in DocID length L corresponds to enhanced performance in both MRR@10 and Recall@10. Secondly, while maintaining a fixed DocID length and increasing the vocabulary size , there is a noticeable improvement in performance metrics. For instance, when $L = 16$, increasing the vocabulary size from 512 to 1024 leads to the 5.5% improvement in terms of MRR@10.

The quality of document approximated representation In Section 3.2.2, we observed the importance of relevance-based document similarities on the model's performance. To show that our model can capture these signals, we randomly select 20 queries from TREC DL 19 and TREC DL 20, along with their corresponding relevant documents. We utilize the approximated vector representation $\hat{\mathbf{d}} = \sum_{i=1}^{L} \mathbf{E}_i[c_i^d]$ and apply T-SNE [van der Maaten and Hinton, 2008] to project the approximated representations of each document into a 2D space for visualization. We studied clustering quality for different prefix lengths, specifically $L = 1, 2, 4, 8, 16$, and 32, as illustrated in Figure 4. First, when $L \geq 8$, those documents relevant to the same query are located in their corresponding cluster, which indicates that our RIPOR effectively draws relevant documents near each other while distancing the irrelevant ones.

Second, the clustering quality is progressively improved when L increases. This might be because when L increases, the distance between approximated vector $\hat{\mathbf{d}}$ and original vector $\mathbf{d}$ diminishes, enabling the approximation to capture finer-grain information.

The influence of prefix-length The prefix length plays a pivotal role in the RIPOR framework due to its influence on the distortion error between the original and approximated vectors. While Section 3.3.3 provides a qualitative perspective on its effects in terms of document similarities in a low-dimensional space, this section delves into its quantitative impact on retrieval performance, as depicted in Figure 5. Referring to the left figure, which displays different DocID combinations from RIPOR, several trends emerge. First, as the prefix length L grows, there is consistent improvement in performance. Second, the rate of this performance gain is more pronounced for shorter prefix lengths, since we observe that the boost is more substantial when $L \leq 8$ than when $L > 8$. Third, given an equal prefix length, variants with a larger vocabulary size tend to perform better. From the right figure which contrasts RIPOR with three other selected variants from the ablation study in Section 3.3.2, we make the following observations. First, excluding the prefix-oriented optimization invariably results in reduced performance. Second, the performance curve of the "replace with sentence-T5" variant emphasizes the critical role of DocID initialization. Compared to prefix-oriented optimization, DocID initialization might be more critical for the model performance, since removing it leads to a larger performance drop. Third, product quantization (PQ) seems less compatible with generative retrieval, given that its performance barely increases when $L >= 8$. The performance stagnation might be due to PQ's weakness in capturing the hierarchical nuances among documents, subsequently impacting the benefits drawn from longer prefix lengths.

3.4 Summary

In this chapter, we introduced the RIPOR framework for improving the performance of generative retrieval models on large-scale datasets. We employed a novel prefix-oriented

ranking optimization method to harness the sequential nature of DocID generation, and applied residual quantization for DocID construction on relevance-based representations. On the MSMARCO passage dataset with 8.8 million passages, RIPOR surpasses the best performing generative retrieval model by 30.6% in terms of MRR@10, and performs on par with popular dense retrieval models. This work sets an important milestone in generative retrieval research by demonstrating, for the first time, that generative retrieval models can be trained to perform effectively at scale, paving the path towards their implementation in real-world applications.

RIPOR improves what the model learns and how identifiers are constructed, but it does not remove the local-search nature of constrained beam search. Even with improved training objectives and relevance-aware DocID construction, beam search can still prune relevant documents at early decoding steps, and increasing the beam size incurs substantial latency costs. Chapter 4 therefore turns from training-time alignment to inference-time decoding, introducing a hybrid simultaneous-autoregressive framework to guide generation with document-level score estimates.

CHAPTER 4

DECODING IN GENERATIVE INFORMATION RETRIEVAL

Chapter 3 demonstrated that generative information retrieval (GIR) can be substantially improved through better training objectives and retrieval-oriented identifier construction. In particular, RIPOR narrows the performance gap between generative and dense retrieval models on large-scale benchmarks, showing that careful optimization and relevance-aware DocID design can mitigate many of the shortcomings of early GIR systems. However, we find that RIPOR's effectiveness still depends heavily on the beam size used during decoding. On MS MARCO Dev, RIPOR achieves only 0.170 MRR@10 with beam size 10, while increasing the beam size to 1000 raises MRR@10 to 0.333. This improvement comes at a direct efficiency cost: a larger beam requires the autoregressive decoder to keep and expand more partial DocID prefixes, which increases the number of model forward computations and makes inference much slower. For retrieval systems, where latency is a central practical constraint, simply increasing the beam size is therefore not a satisfactory solution. This chapter introduces PAG, which first estimates document-level relevance through simultaneous decoding and then uses those scores to guide prefix expansion during autoregressive DocID generation. This guidance allows the model to use much smaller beams while maintaining retrieval effectiveness.

The underlying vulnerability stems from the nature of beam search. As a local search algorithm that expands and retains prefixes based solely on partial scores accumulated so far, beam search can easily become trapped in locally optimal branches. Unlike open-ended text generation, where diverse surface forms can convey similar meanings, generative retrieval assigns a single document identifier to each document. Thus, once the correct identifier path is pruned from the beam at any decoding step, no alternative trajectory exists to recover the relevant document, regardless of how high its full-sequence score might

have been. This irrecoverability creates a brittle sequential decoding problem where early mistakes are fatal.

To address this limitation, we propose PAG, an optimization and decoding approach that gives beam search a document-level view of where each prefix may lead before committing to local next-token decisions. Each document is represented with two complementary identifiers: a set-based identifier and a sequential identifier. The set-based identifier is unordered, so PAG can use it to approximate document-level scores in one decoding step. PAG then decodes the sequential identifier conditioned on the previous generations. The hypothesis is simple: document-level scores from simultaneous set-based decoding can help relevant prefixes survive early pruning. We therefore revisit both optimization and decoding under this view.

Empirically, PAG establishes a new state-of-the-art for generative retrieval on standard large-scale IR benchmarks. On the MSMARCO passage ranking dataset with 8.8 million passages, PAG improves MRR@10 over RIPOR [Zeng et al., 2024] by 15.6%, while using a $10\times$ smaller beam size and achieving a $22\times$ improvement in query latency on a single A100 GPU. On the TREC Deep Learning Track 2019 and 2020, PAG improves NDCG@10 by 12.3% and 10.9%, respectively, over the previous best generative model. These gains come from a three-stage optimization pipeline. The first stage trains set-based DocID generation, the second trains sequential DocID generation, and the final stage jointly optimizes the two identifiers in one model. Beyond its advantages over existing generative models, PAG also demonstrates competitive or superior performance compared to effective dense retrievers such as tas-b [Hofstätter et al., 2021], RocketQA [Qu et al., 2020], and TCT-ColBERT [Lin et al., 2020], while requiring $7.7\times$ less memory for corpus indexing. By combining document-level lookahead with autoregressive generation, PAG demonstrates that generative retrieval can be both effective and efficient at scale.

4.1 Constrained Beam Search in GIR

Generative retrieval models often decode each DocID autoregressively using constrained beam search [Tay et al., 2022, Mehta et al., 2023, Zeng et al., 2024, Wang et al., 2022, Zhuang et al., 2022b]. At each decoding step i, the beam search algorithm maintains the top k prefixes with the highest probabilities (denoted by P_i^{topk}, where $|P_i^{\text{topk}}| = k$) and expands each prefix by one token. Therefore, at each decoding step, many scored prefixes are pruned due to their low probability. The constrained beam search algorithm additionally uses a prefix tree [Tay et al., 2022] to keep track of valid next tokens for each prefix. The prefix tree is built based on all DocIDs $\{c^d : \forall d \in \mathcal{C}\}$, where $\mathcal{C}$ is the corpus. Therefore, constrained beam search ensures that every newly generated prefix belongs to at least one valid DocID. This can be accomplished using the following masking function g, defined for any sequence length $1 \leq i \leq L$, based on the prefix tree:

$$g([c_1, c_2, \cdots, c_i]) = \begin{cases} 0 & \text{if } [c_1, c_2, \cdots, c_i] \text{ is a valid prefix.} \\ -\infty & \text{if } [c_1, c_2, \cdots, c_i] \text{ is not a valid prefix.} \end{cases}$$

Therefore, at the i^{th} decoding step, the constrained beam search algorithm assigns the following score to expand each prefix $c^p_{<i} = [c_1, c_2, \cdots, c_{i-1}] \in P_{i-1}^{\text{topk}}$ by one token:

$$\begin{aligned} s(c^p_{\leq i}; q) &= s(c^p_{<i}; q) + s(c^p_i; q, c^p_{<i}) + g(c^p_{\leq i}) \\ &= s(c^p_{<i}; q) + \mathbf{E}_i[c^p_i] \cdot \mathbf{h}^p_i + g(c^p_{\leq i}) \end{aligned} \tag{3}$$

Based on the scoring function, the top k expanded prefixes with highest probabilities are maintained for the next step.

Pitfalls of (Constrained) Beam Search:

Beam search is a greedy local search algorithm that tends to get stuck into local optima instead of the global optimum [Zhou and Hansen, 2005, Stahlberg and Byrne, 2019]. Even

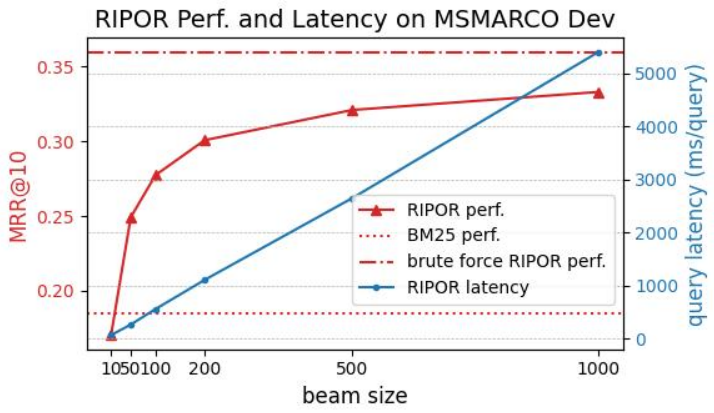


Figure 6: RIPOR effectiveness and latency across beam sizes on MS MARCO Dev. We report retrieval effectiveness (MRR@10) and query latency of RIPOR [Zeng et al., 2024] on MS MARCO Dev, a standard passage retrieval benchmark with 8.8M passages. The experiment is conducted on a single A100 GPU with 80GB memory. Best viewed in color.

though beam search has been successfully used in natural language generation [Sutskever et al., 2014, Graves, 2012], we hypothesize that it is not sufficient for developing effective generative retrieval models. In language generation, there are multiple alternatives that can be equally acceptable outputs, e.g., grammatically correct sentences with same semantics. Hence, if a word token is dropped through beam search, it is likely that other equally good word tokens be kept for the next decoding step. However, in generative retrieval, each relevant document is represented with a unique DocID and if a prefix of this DocID is pruned by the beam search decoding algorithm, there is no way to recover, and that relevant document will not appear in the retrieval result list.

To empirically validate this hypothesis, we study RIPOR [Zeng et al., 2024], the strongest generative retrieval model at the time. RIPOR uses constrained beam search for DocID decoding. Figure 6 reports its effectiveness and latency under different beam sizes on the MS MARCO Dev set [Campos et al., 2016]. Increasing the beam size substantially improves RIPOR's MRR@10, even though evaluation only considers the top 10 retrieved documents. Yet beam size 1000 still does not match brute-force decoding, where every document in the corpus is scored without prefix pruning. Large beams also increase query

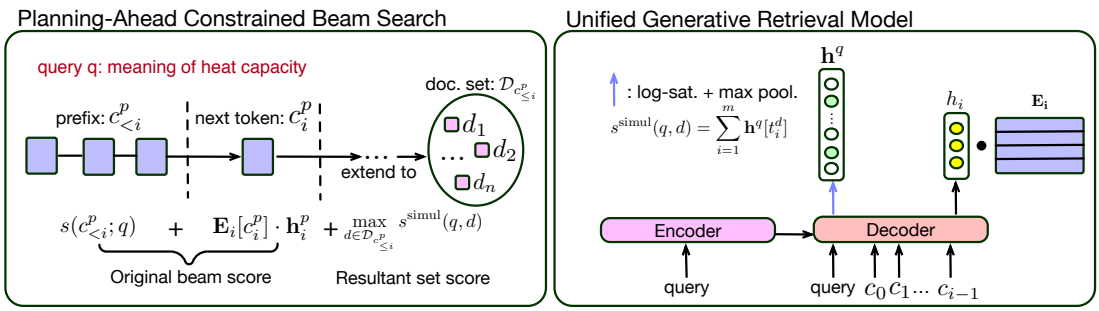


Figure 7: A visualization of the PAG framework. Left: Illustration of simultaneous decoding guiding autoregressive generation with approximate document-level scores. Right: illustration of the model M employing joint decoding of set-based and sequential DocIDs.

latency, limiting the model's practicality. These results suggest that many relevant prefixes are pruned too early by constrained beam search, even when the beam is relatively large.

Motivated by these results, PAG uses the prefix tree to connect each partial DocID with the documents it can still reach. It then guides autoregressive generation through simultaneous decoding, as described in Section 4.2.1. We construct set-based and sequential DocIDs for the two decoding procedures; the construction is introduced in Section 4.2.2. We also use a three-stage training pipeline to adapt the model to joint decoding; details are provided in Section 4.2.3. The high-level overview of the PAG framework is illustrated in Figure 7.

4.2 Methodology

4.2.1 Planning-Ahead Constrained Beam Search

The prefix tree used in constrained beam search ensures that every expanded prefix $c^p_{\leq i}$ at step i is a valid sequence, thus leading to a set of valid DocIDs once decoding finishes. However, according to Equation (3), which is used in state-of-the-art generative retrieval models, document ID prefixes are expanded solely based on the contribution of the next-token score. The motivating experiments in Section 4.1 demonstrate that this is not sufficient: the prefixes of many relevant documents are pruned in constrained beam search,

even with large beam sizes. This section introduces a generative retrieval approach that plans sequential decoding using efficient simultaneous decoding to approximate document-level scores. These scores act as priors for sequential decoding, allowing the decoding phase to keep tokens that are likely to lead to high relevance scores once decoding is complete.

Simultaneous Decoding: We introduce *simultaneous decoding*, which produces a score for each document in one decoding step, using $s^{\text{simul}}(q, d)$. The next subsection incorporates this document-level score into sequential decoding of autoregressive models. To this aim, for each document $d \in \mathcal{C}$, we construct a new type of set-based DocIDs t^d, consisting of a *set* of tokens $\{t_1^d, t_2^d, \cdots, t_m^d\}$, where m is the set size for each document d. Note that m is a constant for all documents and is a hyper-parameter. Unlike c^d, t^d is a set and there is no particular order for tokens in t^d, hence they can be decoded simultaneously.

To compute the simultaneous decoding scores for a given query q, we feed the query text to the generative model M and obtain the contextualized representations:

$$\mathbf{Q} = \text{Decoder}(\text{Encoder}(q), q) \in \mathbb{R}^{|q| \times D}.$$

Here, $|q|$ is the query length and D is the output embedding dimensionality for each token. Let V_T denote the vocabulary size for DocIDs in simultaneous decoding, and $\mathbf{E}_{\text{simul}} \in \mathbb{R}^{V_T \times D}$ denote the associated embedding matrix. Inspired by Formal et al. [2021b,a], we apply log-saturation and max pooling operations to compute a weight per DocID token.

$$\mathbf{h}^q = \text{MaxPool}\big(\log(1 + \text{Relu}(\mathbf{E}_{\text{simul}} \cdot \mathbf{Q}^T))\big) \in \mathbb{R}^{V_T} \tag{4}$$

Then the document-level simultaneous relevance score for every (q, d) pair is then computed as:

$$s^{\text{simul}}(q, d) = \sum_{i=1}^{m} \mathbf{h}^q[t_i^d] \tag{5}$$

$s^{\text{simul}}(q, d)$ can also be written as $s^{\text{simul}}(q, t^d)$.

Guiding Autoregressive Generation through Simultaneous Decoding: PAG uses simultaneous document scoring as priors for computing prefix scores in autoregressive generation. In other words, for decoding any prefix, we consider the maximum approximate document score that can be associated with that prefix as prior. There exist other aggregation functions, such as mean$(\cdot)$ or min$(\cdot)$, to consume here, however, we empirically found max$(\cdot)$ superior. Let $\mathcal{D}$ be a set of top n documents with the highest scores, according to Equation (5). We rewrite Equation (3) as:

$$
\begin{aligned}
s'(c^p_{\leq i}; q) &= \max_{d \in \mathcal{D}_{c^p_{\leq i}}} s^{\text{simul}}(q, d) + s(c^p_{\leq i}; q) \\
&= \max_{d \in \mathcal{D}_{c^p_{\leq i}}} s^{\text{simul}}(q, d) + s(c^p_{<i}; q) + \mathbf{E}_i[c^p_i] \cdot \mathbf{h}^p_i + g(c^p_{\leq i})
\end{aligned}
\tag{6}
$$

where $\mathcal{D}_{c^p_{\leq i}} = \{d \in \mathcal{D} | c^p_{\leq i} = c^d_{\leq i}\}$ is a subset of all documents from $\mathcal{D}$ with the prefix of $c^p_{\leq i}$. This modified scoring function conditions the next token decoding on an approximate of (future) resultant document score through simultaneous scoring. This can minimize the impact of aggressive document ID pruning in the original beam search algorithm. Based on the modified prefix scoring function, we propose a novel decoding method for generative retrieval and term it as *planning-ahead constrained beam search*. This decoding process is illustrated in Algorithm 1.

Computational Cost of Decoding: To assess the computational costs of sequential and simultaneous decoding, we use Floating Point Operations (FLOPs). Sequential decoding mainly incurs costs from multiple forward calls of the generative model M. Assuming a beam size of k, a sequential DocID length L, and P_m FLOPs per forward call of M, its total FLOPs are approximately:

$$
P_{\text{seq}} = L \cdot k \cdot P_m.
$$

In contrast, simultaneous decoding uses a single forward call of M, plus the computations needed to score the corpus using Equation (5). If the corpus size is $|\mathcal{C}|$, its total FLOPs are:

$$P_{\text{simul}} = P_m + |\mathcal{C}| \cdot m.$$

The difference is therefore:

$$\Delta P = P_{\text{seq}} - P_{\text{simul}} = P_m \cdot (L \cdot k - 1) - |\mathcal{C}| \cdot m.$$

Assume M is T5-base, a common backbone for generative retrieval [Zeng et al., 2024, Mehta et al., 2023, Tay et al., 2022, Wang et al., 2022]. This model requires approximately 7.5×10^9 FLOPs per forward call. On MSMARCO, which contains 8.8 million passages, simultaneous decoding is therefore faster than sequential decoding. The latency comparison in Table 6 empirically supports this analysis. This chapter focuses on million-scale retrieval. At billion scale, brute-force simultaneous decoding may become slower. Approximate search techniques may be needed to maintain efficiency [Jégou et al., 2011, Johnson et al., 2019]. We leave this direction for future work.

4.2.2 DocID Construction

We can envision multiple approaches for constructing the sequence- and the set-based document identifiers. Without loss of generality, in the following, we describe the approach used in this chapter.

Sequential DocID Construction: We follow the approach of relevance-based DocID initialization introduced in RIPOR [Zeng et al., 2024] to construct the sequential DocIDs (i.e., c^ds), in which we treat the generative model M as a dense encoder. We utilize the encoder-decoder architecture of M by feeding a document text to the encoder and a start token to the decoder. The document representation is then obtained by the contextualized

Algorithm 1 Planning-Ahead Constrained Beam Search

Require: Generative retrieval model M, query q, beam size k, retrieval corpus $\mathcal{C}$, set-based DocIDs $\{t^d\}_{d\in\mathcal{C}}$, sequential DocIDs $\{c^d\}_{d\in\mathcal{C}}$, vocabulary size for each token embedding V, sequential DocID length L.

Pre-compute document-level scores for every t^d: $s^{\text{simul}}(q, t^d)$ using Eq. (5), and select the top n documents to construct the set $\mathcal{D}$.

Find every possible prefix $c^*_{\leq i}$ and the resultant set $\mathcal{D}_{c^*_{\leq i}}$ from $\mathcal{D}$. Based on that, construct a dictionary T, where the key is $c^*_{\leq i}$ and the value is $\max_{d\in\mathcal{D}_{c^*_{\leq i}}} s^{\text{simul}}(q, t^d)$.

$B_1 \leftarrow \{< 0, 0, \text{BOS} >\}$

for $i = 1$ to L **do**

  $B \leftarrow \emptyset$

  for $(s, s', c^p_{<i}) \in B_i$ **do**

    for $c^p_i \in V$ **do**

      $c^p_{\leq i} \leftarrow [c^p_{<i}, c^p_i]$, $\max_{d\in\mathcal{D}_{c^p_{\leq i}}} s^{\text{simul}}(q, t^d) \leftarrow T[c^p_{\leq i}]$

      Apply Eq. (6) and B.add$\big(< s(c^p_{\leq i}; q)$, $s'(c^p_{\leq i}; q)$, $c^p_{\leq i} >\big)$

    end for

  end for

  $B_{i+1} \leftarrow B$.top(k) based on the $s'(\cdot)$. (the second element)

end for

return B_{L+1} (the third element is DocID, and the second element is corresponding relevant score)

output representations of the decoder:

$$\mathbf{d} = \text{Decoder}(s_0, \text{Encoder}(d)) \in \mathbb{R}^D \tag{7}$$

where s_0 is the start token. We use M as the dense encoder and obtain a representation $\mathbf{d}$ for each document d. We then apply residual quantization (RQ) [Chen et al., 2010] to learn token embedding tables $\{\mathbf{E}_i\}_{i=1}^{L}$. The resulting DocID for each document is $c^d = [c_1^d, \ldots c_L^d]$. This optimization process makes each representation $\mathbf{d}$ approximated by a sequence of token embeddings:

$$\mathbf{d} \approx \sum_{i=1}^{L} \mathbf{E}_i[c_i^d]$$

Set-Based DocID Construction: The sequential DocID c^d captures the document's semantic information by applying the RQ on derived dense representation d. In the realm of IR, it is well-acknowledged that combining the semantic and lexical information of documents would enhance the retrieval system performance [Chen et al., 2022c, Lin et al., 2020, Wang et al., 2021, Lin and Ma, 2021, Zhang et al., 2022]. With this motivation, we set V_T to the vocabulary size of our generative model M, and $\mathbf{E}_{\text{simul}}$ to the embedding table in M. Hence, the tokens $\{t_1^d, t_2^d, \ldots t_m^d\}$ for each set-based DocID t^d will be directly constructed based on the tokenized document content. There exist various methods to score and extract the representative tokens from documents [Sparck Jones, 1972, Lin and Ma, 2021, Formal et al., 2021a,b]. For the scoring part, the traditional methods, such as TF-IDF [Sparck Jones, 1972], weight each token using the corresponding term statistics, e.g., term frequency and inverse document frequency. With the recent advancement of neural lexical models [Formal et al., 2021a], the term importance scores can be directly learned from the supervised training data.

Similar to Equation (4), we take the document content d as the input to the encoder and decoder of the model M to obtain the contextual representation $\mathbf{h}^d \in \mathbb{R}^{V_T}$. Then we

follow the SPLADEv2 training objective [Formal et al., 2021a] by linearly combining the MargiMSE loss (retrieval-oriented objective) with FLOPs regularizer [Formal et al., 2021b, Paria et al., 2020] to sparsify document representations. We describe the training process in Section 4.2.3 in detail. The non-zero weights in $\mathbf{h}^d$ represent the importance score of the corresponding tokens. Hence, we select the top m tokens to form the set-based DocID $t^d = \{t_1^d, \ldots t_m^d\}$ for each document d.

4.2.3 PAG Optimization

The whole optimization process consists of three stages, the first two of which can be trained in parallel. The first two stages are applied to make generative retrieval model capable of predicting set-based DocIDs and sequential DocIDs, respectively. In the final stage, we jointly train the two types of DocIDs together in a unified model, which makes it suitable for the planning-ahead constrained beam search decoding introduced in Section 4.2.1.

Generative Retrieval Model for Set-based DocIDs: This stage contains two training phases: (1) the pre-training phase is used for obtaining the set-based DocIDs and model warmup; (2) the fine-tuning phase is used to train the generative retrieval model for set-based DocIDs prediction.

Pre-training: To obtain the set-based DocIDs t^d, we first treat the generative retrieval model M as a sparse encoder. Given any triple (q, d^+, d^-), where d^+ and d^- represent a relevant and a non-relevant document for q. We follow the MarginMSE method [Hofstätter et al., 2020] to obtain the teacher margin $T(q, d^+, d^-) = S^T(q, d^+) - S^T(q, d^-)$ for each triplet. Usually, the teacher relevance scores $S^T(q, d)$ for each (q, d) pair is derived from the cross-encoder based teacher model [Hofstätter et al., 2020, 2021]. We obtain the sparse representations for q, d^+, d^- using Equation (4). Based on the sparse representations, we apply the training objective of Formal et al. [2021a] with MarginMSE loss and FLOPs regularizer [Paria et al., 2020]. After pre-training, we can apply the "sparse encoder" to

obtain the sparse representation $\mathbf{h}^d$ for every document d. Each non-zero element in the sparse vector represents the important score for the corresponding token. We select top m tokens for each document d as the set-based DocID t^d. We denote the trained model as M^{sp}.

Fine-tuning: We first use M^{sp} as the negative sampler to retrieve the top 100 documents for each query q and denote the set as $\mathcal{D}_q^{sp}$. We construct the training triples: (q, t^{d^+}, t^{d^-}), $d^- \in \mathcal{D}_q^{sp}$ for fine-tuning the generative retrieval model. The model is initialized with M^{sp}. We use Equation (5) to compute the relevance score for each (q, d) pair. The loss function for each triplet is computed as:

$$\mathcal{L}_{\text{set}}(q, t^{d^+}, t^{d^-}) = \left(s^{\text{simul}}(q, t^{d^+}) - s^{\text{simul}}(q, t^{d^-}) - T(q, d^+, d^-)\right)^2$$

Where $T(\cdot)$ is the teacher margin the same as in the pre-training stage. Motivated by the self-negative training strategy [Xiong et al., 2021a, Qu et al., 2020, Zhan et al., 2021], we use the trained model as negative sampler to sample top 100 documents, and merge the negative document set with $\mathcal{D}_q^{sp}$ for each query q, then we apply the same $\mathcal{L}_{\text{set}}$ training loss to fine-tune the model again, and we denote the trained model as M^{set}.

Generative Retrieval Model for Sequential DocIDs: Similarly, this training stage contains two phases. The first phase is to construct the sequential DocIDs and the second phase is to fine-tune the generative retrieval model.

Pre-training: The goal of this stage is to obtain the sequential DocIDs c^d. We follow RIPOR [Zeng et al., 2024], which treats the generative retrieval model as a dense encoder and applies RQ to the resulting dense representations. We train the model with MarginMSE loss and a two-step training strategy. In the first step, negative documents are sampled from the top 100 BM25 results. In the second step, negatives are sampled from the top 100 results of the model trained in the first step. As introduced in Section 4.2.2, we obtain the dense representation for each document d using Equation (7), and then apply RQ to obtain the sequential DocIDs. This stage produces the trained model M^{ds} and the token embeddings

$\{\mathbf{E}_i\}_{i=1}^{L}$.

Similar to RIPOR, this stage also contains a seq2seq pre-training phase. Specifically, we use the doc2query model [Cheriton, 2019] to generate 10 pseudo-queries for each document, then we take each of the pseudo-queries as input to model and predict the corresponding sequential DocID using a seq2seq loss. The model is initialized with M^{ds} and we denoted the trained model as M^{s2s}.

Fine-tuning: Similar to the previous fine-tuning stage, we construct the training triplets by using the M^{ds} to sample top 100 negative documents and denote the negative set as $\mathcal{D}_q^{ds}$ for each query q. We apply the multi-objective training loss in RIPOR [Zeng et al., 2024] for prefix-oriented fine-tuning. The full-length relevance score between (q, c^d) can be computed via Equation (3.1). Similarly, the relevance score by generative retrieval model produced by the first i tokens of c^d: $[c_1^d, \ldots c_i^d]$ can be computed via Equation (3). Given any triplet (q, c^{d^+}, c^{d^-}), we can use the modified MarginMSE loss for each prefix with length i:

$$\mathcal{L}_{\text{seq}}^i = \left(s(c_{\leq i}^{d^+}; q) - s(c_{\leq i}^{d^-}; q) - \alpha_i T(q, d^+, d^-)\right)^2$$

where α_i is the monotonically increasing weight w.r.t i ranging from [0, 1], and $\alpha_L = 1$. Refer to Zeng et al. [2024] for the details. Therefore, the multi-objective loss is:

$$\mathcal{L}_{\text{seq}} = \sum_{i \in S_L} \mathcal{L}_{\text{seq}}^i$$

S_L is a set containing the sampled prefix lengths and L. The optimized model is termed as M^{seq} which starts from M^{s2s}. Once M^{seq} is trained, we use it as the negative sampler for sampling top 100 documents for each query q, denoted as $\mathcal{D}^{sq}$.

Unified Optimization: In this stage, we train a single model to predict both set-based and sequential DocIDs. This makes the model compatible with the proposed *planning-ahead constrained beam search*. We initialize M by averaging the weights of M^{set} and M^{seq}. The training triples contain both identifier types: $\{(q, t^{d^+}, t^{d^-}), (q, c^{d^+}, c^{d^-})\}$. The

negative sample set for each query q is $\mathcal{D}_q^{sp} \cup \mathcal{D}_q^{sq}$. We compute $s^{\text{simul}}(q, d)$ using Equation (5). To support document-level simultaneous relevance scoring, we slightly modify how M generates the hidden representation for the next token c_i^d. Different from Equation (3.1), we additionally feed the query content to the decoder:

$$\mathbf{h}_i^d = \text{Decoder}(\text{Encoder}(q), q, c_{<i}^d) \in \mathbb{R}^D$$

Based on this, $s(c^d; q)$ is computed the same as Equation (3.1). Therefore, we use the following objective to train the unified generative retrieval model for each triplet:

$$\mathcal{L}_{\text{set}}(q, t^{d^+}, t^{d^-}) + \mathcal{L}_{\text{seq}}(q, c^{d^+}, c^{d^-})$$

4.3 Experiments

4.3.1 Experiment Settings

Datasets. We assess our model on the MSMARCO passage retrieval benchmark, which contains 8.8M passages and 532K train queries. Each training query contains 1.1 relevant passages on average. We evaluate our model in three evaluation datasets: (1) MSMARCO Dev with 6980 queries with incomplete relevance annotations; (2, 3) TREC-DL 2019 & 2020: the passage retrieval datasets are used in the first and second iterations of TREC Deep Learning Track [Craswell et al., 2019, 2021], which contains 43 and 54 queries respectively with a relatively complete relevance annotations done by TREC via pooling. For evaluation, we use the official metric per dataset: (1) MRR@10 for MSMARCO Dev; (2) NDCG@10 for TREC-DL 19 and 20. We additionally follow Zeng et al. [2024] and use Recall@10 for all the three datasets.

Implementation Details We use T5-base [Raffel et al., 2019] as the backbone model for our generative retrieval model M. For sequential DocID initialization, we apply the residual

Table 4: Experimental results on MSMARCO and TREC Deep Learning Track Data. Highest generative retrieval performances are boldfaced. Superscript $*$ denotes statistically significant improvement compared to all generative retrieval baselines. Superscripts $^{\triangle}$ and $^{\triangledown}$ denote significantly higher and lower performance compared to PAG for sparse and dense retrieval models. (t-test with Bonferroni correction, p_value < 0.01). We use brute-force search for dense retrieval models.

Model	KD	Mem.	MSMARCO Dev		TREC DL 2019		TREC DL 2020	
			MRR@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10
Generative Retrieval Methods								
DSI	✗	0.03	.045	.138	.163	.076	.150	.070
DSI-QG	✗	0.03	.105	.292	.320	.138	.328	.120
Ultron-PQ	✗	0.84	.117	.248	.331	.135	.342	.134
NCI-QG	✗	0.03	.153	.352	.403	.167	.394	.159
SEAL	✗	-	.127	-	-	-	-	-
NOVO	✗	0.80	.126	.242	.258	.112	.310	.140
MINDER	✗	12.16	.186	.383	.506	.201	.392	.144
LTRGR	✗	12.16	.255	.531	.598	.238	.553	.182
RIPOR	✓	1.06	.333	.562	.628	.205	.631	.191
PAG	✓	3.27	**.385***	**.670***	**.705***	**.267***	**.700***	**.236***
Some Sparse and Dense Retrieval Methods (For Reference)								
BM25	✗	4.00	.185$^{\triangledown}$	.381$^{\triangledown}$	.512$^{\triangledown}$	.178$^{\triangledown}$	.477$^{\triangledown}$	.164$^{\triangledown}$
docT5query	✗	13.00	.272$^{\triangledown}$	.536$^{\triangledown}$	.642$^{\triangledown}$	.247$^{\triangledown}$	.619$^{\triangledown}$	.224$^{\triangledown}$
ANCE	✗	25.30	.330$^{\triangledown}$	.566$^{\triangledown}$	.648$^{\triangledown}$	.239$^{\triangledown}$	.646$^{\triangledown}$	.185$^{\triangledown}$
ADORE	✗	25.30	.347$^{\triangledown}$	.611$^{\triangledown}$	.683$^{\triangledown}$	.264$^{\triangledown}$	.666$^{\triangledown}$	.214$^{\triangledown}$
RocketQA	✓	25.30	.370	-	-	-	-	-
TCT-ColBERT	✓	25.30	.335$^{\triangledown}$	.596$^{\triangledown}$	.670$^{\triangledown}$	.240$^{\triangledown}$	.668$^{\triangledown}$	.218$^{\triangledown}$
MarginMSE	✓	25.30	.325$^{\triangledown}$	.581$^{\triangledown}$	.699$^{\triangledown}$	.250$^{\triangledown}$	.645$^{\triangledown}$	.203$^{\triangledown}$
tas-b	✓	25.30	.344$^{\triangledown}$	.622$^{\triangledown}$	.717$^{\triangle}$	.255$^{\triangledown}$	.685$^{\triangledown}$	.230$^{\triangledown}$
cl-drd	✓	25.30	.382$^{\triangledown}$	.651$^{\triangledown}$	.725$^{\triangle}$	.266	.687$^{\triangledown}$	.216$^{\triangledown}$

quantization (RQ) implementation from Faiss [Johnson et al., 2019]. The sequential DocID length L is set to 8 and the vocabulary size V is set to 2048. As for hyper-parameters used in set-based DocIDs, the size of selected tokens (m) is 64. In the pre-training and fine-tuning stages for set-based DocIDs, we set the learning rate to 0.0005 and use 100 training epochs. The weights for the FLOPs regularization [Paria et al., 2020, Formal et al., 2021a,b] for query and document representations are 0.01 and 0.008, respectively. In the dense encoder pre-training and prefix-oriented fine-tuning stages for sequential DocIDs, the learning rate is also set to 0.0005 and the training epochs are 50 and 150, respectively. The $S_L = \{4, 8\}$, and the corresponding weights α_is are $\{0.5, 1.0\}$. In the seq2seq pre-training stage, the learning rate is 0.001 and the number of training steps is $250,000$. For the final unified training stage, the learning rate is 0.0005 and the number of training epochs is set to 120. We use Adam optimizer [Kingma and Ba, 2015] with linear warmup scheduling, the warmup ratio is set to 4.5% of the total training steps. The beam size is 100, and the top 1000 documents are selected to form $\mathcal{D}$ for inference. The model is trained on 8 A100 GPUs each with 40GB memory. For fair comparison in terms of efficiency, we use an A100 GPU with 80GB memory for all models at inference.

Baselines We compare our model with the following generative retrieval models: DSI [Tay et al., 2022], DSI-QG [Zhuang et al., 2022b], NCI-QG [Wang et al., 2022], Ultron-PQ [Zhou et al., 2022], SEAL [Bevilacqua et al., 2022], NOVO [Wang et al., 2023], MINDER [Li et al., 2023b], LTRGR [Li et al., 2023a] and RIPOR [Zeng et al., 2024]. To the best of our knowledge, RIPOR provides the strongest performance among all generative retrieval baselines. We also select the following competitive sparse and dense retrieval methods as points of reference: BM25 [Robertson and Zaragoza, 2009], docT5query [Cheriton, 2019], ANCE [Xiong et al., 2021a], ADORE [Zhan et al., 2021], RocketQA [Qu et al., 2020], TCT-ColBERT [Lin et al., 2020], MarginMSE [Hofstätter et al., 2020], tas-b [Hofstätter et al., 2021], and cl-drd [Zeng et al., 2022].

4.3.2 Main Results

Comparison with Baselines: Table 4 reports the main comparison. PAG is compared with prior generative retrievers, as well as strong sparse and dense retrievers. Across all three datasets, PAG outperforms the generative baselines. Compared with RIPOR, PAG improves MRR@10 on the MSMARCO Dev set by 15.6%. Both methods use knowledge distillation, so the gain mainly reflects the benefit of adding set-based DocIDs for lexical relevance scoring during decoding. PAG also uses beam size 100, which is 10 times smaller than the beam size of 1000 used in RIPOR. This substantially reduces query latency and supports the effectiveness of planning-ahead constrained beam search. Compared with dense retrieval methods, PAG also achieves competitive or stronger effectiveness while using less index memory. For instance, PAG improves MRR@10 by 11.9% over tas-b, and on TREC-DL 2020, it improves NDCG@10 by 2.2%. PAG also uses $7.7\times$ less index memory than dense retrieval models. **Ablation Studies.** We conduct a thorough ablation study on MSMARCO dataset to investigate the impact of each component in PAG. The results are shown in Table 5.

Beginning with Row 1, eliminating $s^{\text{simul}}(\cdot)$ in Equation (6) causes a 10.3% drop in MRR@10 and a 17.7% drop in Recall@10 on the MSMARCO Dev set. This drop shows the value of simultaneous relevance scoring. Since set-based DocIDs are constructed from lexical signals, they complement the semantic information captured by sequential DocIDs. Row 2 further shows the importance of combining the two signals. Using only $s^{\text{simul}}(\cdot)$ for retrieval decreases MRR@10 and Recall@10 by 27% and 9.1%, respectively.

Based on Rows 3 and 4, we can infer that the more effective M^{seq} can boost the unified generative retrieval model. For instance, applying seq2seq pre-training can lead to a 1.0% improvement in MRR@10, while using the multi-objective loss for fine-tuning can result in a 1.3% increase. Seq2seq pre-training applies the DocID prediction task over the whole corpus mitigating the distribution shift between training and evaluation sets. The multi-objective loss is designed for sequential decoding algorithms, which can reduce the risk of

Table 5: Ablation study results on MSMARCO Dev. Superscript $^{\triangledown}$ denotes significantly lower performance compared to PAG (t-test with Bonferroni correction, p_value < 0.01). itp. stands for interpolation.

	MRR@10	Recall@10	Mem.
PAG	.385	.670	3.27
1. w/o adding $s^{\text{simul}}(\cdot)$	.349$^{\triangledown}$	.614$^{\triangledown}$	0.50
2. Only $s^{\text{simul}}(\cdot)$ for search	.303$^{\triangledown}$	.569$^{\triangledown}$	2.77
3. w/o seq2seq pre-training	.381$^{\triangledown}$	.660$^{\triangledown}$	3.27
4. w/o multi-obj. learning	.380$^{\triangledown}$	.663$^{\triangledown}$	3.27
5. Only M^{set}	.322$^{\triangledown}$	.606$^{\triangledown}$	2.77
6. Only M^{seq}	.339$^{\triangledown}$	.566$^{\triangledown}$	0.50
7. Linear itp. of M^{set} and M^{seq}	.360$^{\triangledown}$	.593$^{\triangledown}$	3.27
8. Only M^{sp}	.378$^{\triangledown}$	.667$^{\triangledown}$	35.28
9. Only M^{ds}	.365$^{\triangledown}$	.641$^{\triangledown}$	25.30

making the relevant prefixes discarded from the beam.

The results in Rows 5 and 6 imply that using M^{set} or M^{seq} alone results in performance degradation. Interestingly, when we linearly interpolate the lexical and semantic scores together for the final relevance score, we observe significant performance gain where the results are shown in Row 7. The simple post-hoc combination leads to 11.8% and 6.2% improvements over M^{set} and M^{seq} on the MSMARCO Dev set. However, it still performs worse than the original model. These findings illustrate the effectiveness of joint modeling of lexical and semantic retrieval models.

Finally, as demonstrated by Rows 8 and 9, PAG achieves superior results over M^{sp} and M^{ds}, improving the MRR@10 by 1.9% and 5.5% respectively. Notably, this performance enhancement is accompanied by a significant reduction in memory usage - PAG requires $\times 10.8$ and $\times 7.7$ less memory compared to M^{sp} and M^{ds}. These imply the set-based and sequential DocIDs can compress retrieval information near-losslessly.

4.3.3 Analysis and Discussion

The impact of beam size and number of selected sub-words: The selection of beam size and number of selected sub-words m affects the effectiveness and efficiency of the

Table 6: Effectiveness and efficiency with different m and k on MSMARCO Dev. The experiment is conducted on an 80GB A100 GPU. Simul. D. and Seq. D. stands for simultaneous and sequential decoding respectively. QL represents query latency (ms / query).

m, k	MRR@10	Recall@10	Mem.	Simul.	Seq.
16, 10	.342	.577	1.30	20	44
32, 10	.367	.626	1.94	22	44
64, 10	.379	.641	3.27	25	44
128, 10	.386	.645	5.96	31	44
16, 100	.355	.620	1.30	20	250
32, 100	.372	.652	1.94	22	250
64, 100	.385	.670	3.27	25	250
128, 100	.390	.664	5.96	31	250

model. A large beam size can reduce the risk of pruning the relevant prefixes out of the beam and a large m might improve the model expressiveness by extracting more lexical information from the document. However, it comes with the trade-off of increasing the query latency and index memory. To quantify this, we report the results of different settings of m and beam size k in Table 6. First, when k is fixed, an increase of m can show a trade-off in retrieval performance and resource utilization. For instance, at $k = 100$, the MRR@10 with $m = 16$ is 0.355, while it increases to 0.390 when $m = 128$, yielding 9.9% enhancement. However, this improvement comes at the cost of $\times 3.58$ and $\times 1.55$ increase in the index memory and query latency, respectively. Second, at a fixed value of m, employing a larger k can enhance retrieval effectiveness. For instance, at $m = 64$, increasing k from 10 to 100 improves MRR@10 by 1.6% albeit at the cost of $\times 5.68$ increase in query latency. It is noteworthy that performance degradation is less pronounced than that observed in RIPOR, as illustrated in Figure 6. This relative robustness can be attributed to the use of planning-ahead constrained beam search in PAG, which re-weights each prefix by a pre-stored document-level relevant score. Finally, we can observe that using simultaneous decoding is much more efficient than autoregressive generation for retrieval and, hence does not introduce too much overhead over the original beam search algorithm.

Comparison between RIPOR and PAG: We train a new RIPOR model with the same

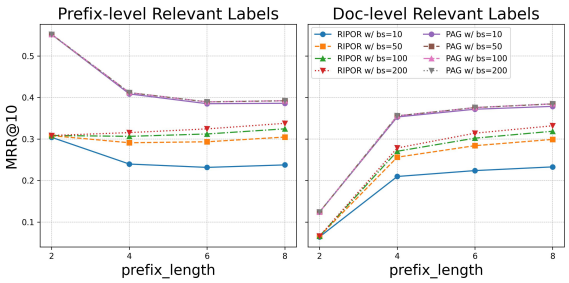


Figure 8: Results on MS MARCO Dev with different beam sizes. The comparison uses prefix-level and document-level labels.

DocID configuration used in PAG ($L = 8, V = 2048$). In addition to the original document-level relevance labels, we construct prefix-level labels where a prefix is treated as relevant if it belongs to a relevant document for the query. We compare RIPOR and PAG under different prefix lengths and beam sizes on the MSMARCO Dev set. The results are shown in Figure 8. PAG consistently outperforms RIPOR under both prefix-level and document-level labeling, and it is less sensitive to beam size. When the beam size is greater than 10, PAG obtains nearly identical MRR@10 across different prefix lengths. In contrast, RIPOR continues to improve as beam size increases. These findings show the effectiveness of incorporating document-level scores into constrained beam search.

Additionally, RIPOR and PAG show different trends under prefix-level relevance labels, as illustrated in the left sub-figure. In PAG, MRR@10 monotonically decreases as the prefix length grows. RIPOR shows the opposite trend, except when the beam size is 10. This difference occurs because simultaneous decoding gives PAG high Recall@10 for relevant prefixes early in decoding. As prefixes become longer, relevant and hard-negative prefixes become more similar, making them harder to distinguish. This characteristic results in the decline of MRR@10 values as prefix lengths increase. Conversely, RIPOR lacks a similar early-stage Recall@10 performance. It relies on a larger beam size (>10) and longer, more expressive prefixes to achieve higher MRR@10 values.

The impact of combining set-based and sequential DocIDs: PAG can also be viewed as a retrieve-then-rerank model. It first retrieves promising documents using set-based

DocIDs, and then refines their scores as the sequential DocID is generated. We qualitatively analyze retrieval effectiveness by examining the quality of set-based DocIDs. We randomly sample 20 queries from TREC 2019 and 2020, and select all relevant documents for these queries. For each document d with set-based DocID $t^d = \{t^d_1, \dots t^d_m\}$, we obtain the T5 embedding learned by PAG for each token, denoted as $\{\mathbf{e}^d_1, \dots \mathbf{e}^d_m\}$. We compute the document embedding as $\mathbf{e}_d = \frac{1}{m}\sum_{i=1}^{m} \mathbf{e}^d_i$. We then apply T-SNE [van der Maaten and Hinton, 2008]. The top panel of Figure 9 visualizes the resulting embeddings. Nearly all documents with the same label, meaning relevance to the same query, are clustered together. This shows that set-based DocIDs capture useful relevance information from document content.

To better understand the reranking effect of combining set-based and sequential DocIDs, we compare PAG with a variant that only uses set-based DocIDs for retrieval. We report per-query ΔMRR@10 on MSMARCO and ΔNDCG@10 on TREC-DL 2019 and 2020. The results are shown in the lower panels of Figure 9. For space, we merge the TREC-DL 2019 and 2020 query sets in this experiment. The plots show that most queries either benefit from joint scoring or are not harmed by it. We acknowledge that not all queries benefit from the joint scoring. Specifically, approximately 1000 out of 6980 queries in MSMARCO Dev and 20 out of 97 queries in TREC-DL 19&20 see a decline in performance. This could be attributed to combined scoring potentially introducing biases that negatively affect queries that are better suited to lexical matching, typically captured by set-based DocIDs.

4.4 Summary

This chapter proposed PAG for generative retrieval decoding. PAG estimates document-level relevance through simultaneous decoding and uses this signal to guide autoregressive generation. Each document is represented by two identifiers. Set-based DocIDs follow the bag-of-words assumption and support simultaneous decoding. Sequential DocIDs come from relevance-based document representations and support autoregressive decoding. We

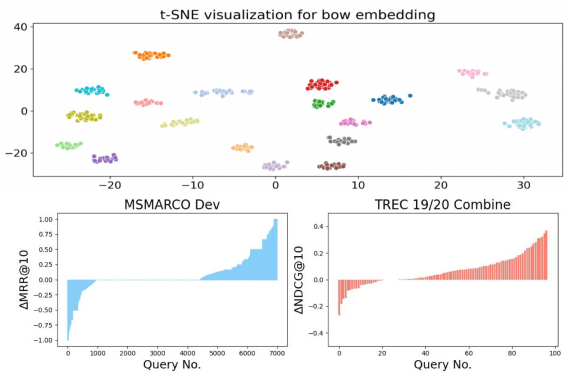


Figure 9: Per-query analysis of PAG on TREC-19/20 and MS MARCO Dev. *Above Figure*: clusters of relevant documents to 20 queries sampled from TREC-19/20, and the color indicates the query ID. *Below Figures*: Δ MRR@10 on MSMARCO Dev and Δ NDCG@10 on TREC-19/20 between simultaneous+autoregressive decoding (PAG) and simultaneous decoding alone for each query.

also introduced a three-stage training pipeline that adapts the model to joint decoding. Experiments show substantial improvements over prior generative retrieval baselines in both effectiveness and efficiency. Compared to RIPOR, PAG achieves a 15.6% relative improvement in MRR@10 on the MSMARCO Dev set while using a 10$\times$ smaller beam size, resulting in a 22$\times$ improvement in query latency on a single A100 GPU. PAG also demonstrates competitive or superior performance compared to effective dense retrievers such as TAS-B, RocketQA, and TCT-ColBERT, while requiring 7.7$\times$ less memory for corpus indexing.

Together with the results from Chapter 3, this chapter establishes that generative retrieval can be both effective and efficient at scale. The next chapter moves to a setting in which retrieval is repeatedly used by an LLM reasoning agent, and asks how the retrieval model should be trained inside that loop.

CHAPTER 5

UNIFYING REASONING AND GENERATIVE INFORMATION RETRIEVAL

Chapters 3 and 4 focus on GIR for standalone ad hoc retrieval: given a query, the system returns a ranked list from a large fixed corpus. This chapter studies agentic search. In this setting, an LLM-based search agent answers complex questions by interleaving reasoning with search: it issues queries, reads retrieved evidence, updates its intermediate reasoning, and decides what to search for next.

Search agents have become increasingly important for knowledge-intensive applications. Large language models (LLMs) have strong reasoning capabilities, but they need external evidence when questions require current, long-tail, or multi-hop knowledge. Retrieval-augmented generation (RAG) [Lewis et al., 2020] addresses this limitation by allowing models to access external knowledge through retrieval. While early approaches relied on single-turn retrieval followed by answer generation, recent work has shown that search agents—which iteratively reason, generate queries, and incorporate retrieved information—yield substantially stronger performance on knowledge-intensive tasks [Trivedi et al., 2023a, Khattab et al., 2021, Shi et al., 2024b]. Training such agents through reinforcement learning (RL), where the reward is based on final answer correctness, has proven effective at teaching them *when* and *what* to search [Jin et al., 2025, Li et al., 2025b, Zeng et al., 2026].

Despite this progress, a common assumption across existing agentic search systems is that the retrieval model or search engine remains unchanged. The agent learns to use retrieval, but the retrieval component itself does not adapt. Through analysis of RL-trained search agents, we observe that many errors arise at the retrieval stage: at a given reasoning step, the agent may issue a sub-query that targets the information needed next, but the retrieval system fails to return evidence that would support that step, forcing the agent to retry with reformulated queries or draw incorrect conclusions from noisy information.

To quantify this gap, we conduct an oracle retrieval experiment in which we promote all documents matching the gold answer to the top positions at each retrieval step. This simple intervention consistently improves average F1 by +14.7% for a 7B-parameter agent and by +26.8% for a 3B agent across seven QA benchmarks. These findings motivate the central question of this chapter: *can we jointly optimize the retrieval system alongside the multi-turn reasoning agent, so that they learn to complement each other?*

To address this question, we propose CoSearch, a reinforcement learning framework that jointly trains a main reasoning agent and a generative document ranker for agentic search. The main agent operates in a standard ReAct loop [Yao et al., 2023b], generating thoughts and search queries. The generative ranker receives the main agent's sub-queries along with candidate documents from a fixed first-stage dense retriever, and selects and reranks a subset for the main agent to observe; together, the retriever and ranker form the retrieval system. The main agent and the ranker are trained simultaneously using Group Relative Policy Optimization (GRPO) [DeepSeek-AI et al., 2025], with rewards derived from the correctness of the final answer.

Applying GRPO to the ranker introduces two key technical challenges. First, GRPO computes advantages over groups of responses generated from the *same input prompt*. While this is naturally satisfied for the main agent—where multiple trajectories are sampled from the same user query—the ranker receives different sub-queries across trajectories, preventing direct grouping. We address this through a *semantic grouping* strategy that first partitions ranker calls by answer availability and then clusters sub-queries by token-level F1 similarity, forming valid GRPO groups without additional rollouts. Second, the ranker's effect on the final answer is indirect: good retrieval may still yield an incorrect answer if reasoning fails, and vice versa. We design a *composite reward* combining a relevance reward based on a ranking quality metric (Hit@k) with a trajectory-level reward reflecting the final answer correctness, providing both immediate feedback on retrieval precision and long-term signal on downstream utility. Both reward components are computed using rule-

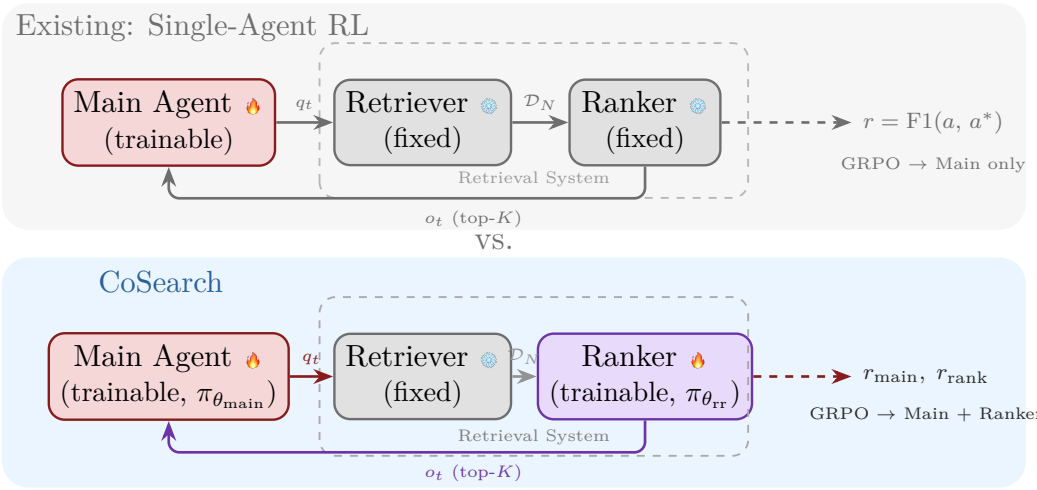


Figure 10: Both panels share the same architecture: Main Agent → Retriever → Ranker. Top: existing approaches treat the entire retrieval system as fixed (❄) and train only the main agent. Bottom: CoSearch decomposes the retrieval system into a fixed dense retriever and a trainable generative ranker (🔥), jointly optimizing both the main agent and the ranker via GRPO.

based metrics, requiring no expensive LLM annotations.

We evaluate CoSearch on seven QA benchmarks spanning single-hop and multi-hop settings. Using Qwen2.5-7B-Instruct as the backbone for the ranker, and evaluating both 7B and 3B main agents, our method achieves the best average F1 among the evaluated systems: +6.6% relative over Search-R1 with a 7B agent and +10.0% with a 3B agent. These results demonstrate that jointly training the retrieval system and reasoning agent is feasible and effective, and suggest that co-optimization of retrieval and reasoning is a promising direction for building more capable agentic search systems.

5.1 Methodology

We present CoSearch, a framework that jointly optimizes a multi-step reasoning (main) agent and a generative document ranker through reinforcement learning (RL) (see Figure 10). We first formalize the framework (§5.1.1), describe the main agent optimization (§5.1.2), and then detail our two core technical contributions: a semantic grouping strategy for applying GRPO to the ranker (§5.1.3) and a composite reward design (§5.1.4).

5.1.1 Problem Formulation

We formalize agentic search as a system comprising a reasoning (main) agent and a retrieval system. Given an initial user query q_0, the *main agent* $\pi_{\theta_{\text{main}}}$ interacts with the retrieval system through a sequence of reasoning and retrieval steps to produce a trajectory:

$$y = \big(q_0,\ \tau_1, q_1, o_1,\ \tau_2, q_2, o_2,\ \ldots,\ \tau_m, q_m, o_m,\ \tau_{m+1}, a\big), \tag{8}$$

where τ_t is the reasoning thought at step t, q_t is the sub-query generated by the main agent, o_t is the observation (retrieved documents) returned by the retrieval system, and a is the final answer.

In existing agentic search frameworks [Jin et al., 2025, Gao et al., 2025, Li et al., 2025a, Zheng et al., 2025], the observation o_t is provided by a fixed retrieval. In our framework, we decompose the retrieval system into two stages and make the second stage trainable: a first-stage dense retriever produces a candidate set $\mathcal{D}_N$ of N documents, and a *generative ranker* $\pi_{\theta_{\text{gr}}}$ then selects and ranks a subset $\mathcal{D}_K \subset \mathcal{D}_N$ of K documents to form the observation o_t (see Appendix B.2 for the full algorithm and concrete examples). The dense retriever remains fixed; the ranker is the trainable component. The joint optimization objective is:

$$\max_{\theta_{\text{main}},\, \theta_{\text{gr}}} \quad \mathbb{E}_{T \sim \pi_{\theta_{\text{main}}},\, \pi_{\theta_{\text{gr}}}} \big[r(T) \big], \tag{9}$$

where $r(T)$ measures the quality of the final answer. Unlike existing approaches that freeze the retrieval system and only optimize θ_{main}, our framework makes o_t a function of the trainable ranker, enabling the ranker to be optimized end-to-end by downstream answer correctness.

5.1.2 Main Agent Optimization

The main agent is optimized over reasoning trajectories using GRPO [DeepSeek-AI et al., 2025]. For each query q_0, we sample G trajectories $\{y_i\}_{i=1}^{G}$ from the current policy. The GRPO objective is:

$$\mathcal{J}_{\text{GRPO}}(\theta_{\text{main}}) = \mathbb{E}_{q_0,\,\{y_i\}_{i=1}^{G}} \left[\frac{1}{G} \sum_{i=1}^{G} \sum_{t=1}^{|y_i|} m_{i,t} \min\Big(\rho_{i,t}\, \hat{A}_i,\ \text{clip}(\rho_{i,t},\, 1{-}\epsilon,\, 1{+}\epsilon)\, \hat{A}_i \Big) \right], \quad (10)$$

where $\rho_{i,t} = \pi_\theta(y_{i,t} \mid y_{i,<t}) \,/\, \pi_{\theta_{\text{old}}}(y_{i,t} \mid y_{i,<t})$ is the importance sampling ratio and $\hat{A}_i$ is the group-normalized advantage. Following Jin et al. [2025], we apply a loss mask $m_{i,t}$ that equals 1 for tokens generated by the main agent and 0 for retrieved document tokens.

Reward. The main agent reward evaluates both answer correctness and format compliance:

$$r_{\text{main}}(q_0, y_i) = \begin{cases} s_{\text{ans}}, & \text{if } f = 1, \\ -\alpha, & \text{if } f = 0, \end{cases} \quad (11)$$

where s_{ans} is the token-level F1 score between the predicted and gold answers, $f \in \{0, 1\}$ indicates whether the trajectory follows the required ReAct format [Yao et al., 2023b, Jin et al., 2025], and $\alpha = 0.2$ is the format penalty.

5.1.3 Ranker Optimization

At each search step t of the i-th rollout, the ranker receives a tuple $(q_0,\, q_t^{(i)},\, \mathcal{D}_N)$, where q_0 is the original user query, $q_t^{(i)}$ is the sub-query issued by the main agent, and $\mathcal{D}_N$ is the top-N documents retrieved by the dense retriever. The ranker first reasons about the relevance of each candidate, then selects and ranks a subset of K documents to form the observation $o_t^{(i)}$. We provide the detailed prompt and input–output examples in Appendix B.1.2. The ranker is trained with GRPO using the same clipped surrogate objective (Eq. 10).

However, directly applying GRPO to the ranker introduces a fundamental challenge.

GRPO requires multiple responses sampled from *the same input prompt* to compute group-normalized advantages. For the main agent, this grouping is natural: all G trajectories originate from the same query q_0. For the ranker, however, the input includes the sub-query $q_t^{(i)}$, which varies across rollouts—different reasoning paths produce different sub-queries. Consequently, we cannot naively treat all ranker calls under the same q_0 as a single GRPO group.

A naive alternative would be to sample dedicated ranker rollouts for GRPO: for each of the $G \cdot m$ sub-queries produced by the main agent, generate H independent ranker responses and continue each resulting trajectory to completion to obtain a reward. However, each ranker response alters the observation o_t and consequently all downstream reasoning, producing new sub-queries that themselves require H rollouts—cascading across m steps into $G \cdot H^m$ total trajectories, an exponential blowup that is clearly intractable.

Semantic grouping and filtering. We address the challenge by a key observation: though sub-queries are not identical across rollouts, many are *semantically equivalent*—different reasoning paths tend to ask similar sub-questions, differing only in surface-level phrasing. For a fixed q_0, the G rollouts produce a pool of sub-queries $\mathcal{Q}(q_0) = \{q_t^{(i)} \mid i \in [1, G],\ t \in [1, T_i]\}$. Before clustering, we partition $\mathcal{Q}(q_0)$ by the answer-availability indicator I_{ans} defined in Section 5.1.4, which records whether the candidate set contains a pseudo-relevant document. This keeps ranker calls from different reward regimes in separate pools. Within each pool, we construct semantic groups $\mathcal{G}(q_0) = \{g_1, g_2, \ldots, g_M\}$ using the following greedy procedure. We iterate through the sub-queries and assign each query q to an existing group g_m if its token-level F1 similarity with the group's representative query q_m^{rep} exceeds a threshold δ:

$$q \in g_m \iff \text{F1}_{\text{token}}(q,\ q_m^{\text{rep}}) \geq \delta, \tag{12}$$

where δ is a similarity threshold. Token-level F1 treats each query as a bag of tokens and computes the harmonic mean of precision and recall. If no existing group matches, a new group is created with q as its representative. Groups with fewer than k_{min} members are

discarded to maintain stable advantage estimation.

Each remaining group g_m contains ranker calls with semantically equivalent inputs under the same q_0, forming a valid GRPO group. Advantages are computed within each group using standard group normalization. This strategy incurs *zero additional sampling cost*: we reuse the trajectories already generated for the main agent, rather than performing separate rollouts for the ranker. We provide a detailed algorithm and an illustration in Appendix B.3.

5.1.4 Ranker Reward Design

The ranker aims to help the main agent solve a multi-step reasoning task by providing useful documents at each search step. However, reward design is non-trivial. A trajectory-level reward based on the correctness of the final answer cannot accurately assign credit to individual search steps, because each step's contribution is mixed with subsequent retrieval decisions and with the main agent's reasoning, making the signal noisy and unstable. A relevance-based reward also has an important limitation in our setting. Since pseudo-relevance labels are constructed by matching documents to the gold answer, they fail to supervise intermediate steps whose retrieved documents are useful for decomposition or reasoning but do not directly mention the final answer. To address these two issues, we use a composite reward that combines trajectory-level supervision with relevance-based supervision.

Relevance reward. The reward r_{rel} measures ranking quality with pseudo-relevance labels: a document is labeled pseudo-relevant if it contains the gold answer. We compute r_{rel} as the average of Hit@k over a set of cutoff values $\mathcal{K}$:

$$r_{\text{rel}} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \text{Hit@}k,$$

where Hit@k indicates whether any pseudo-relevant document appears in the top-k positions

of the ranker's output.

Main agent reward. The main agent reward r_{main} follows the same definition as Eq. 11, i.e., it is given by the token-level F1 score between the predicted and gold answers. Notice that Eq. 11 assigns a penalty $-\alpha$ when the main agent violates the required format ($f = 0$). In this case, we do not propagate this penalty to the ranker. Instead, we filter out such trajectories and exclude them from reward computation and subsequent optimization.

Composite reward. Let $I_{\text{ans}} \in \{0, 1\}$ indicate whether the candidate set $\mathcal{D}_N$ contains a pseudo-relevant document, and let γ be a threshold controlling the reward composition. The final ranker reward for format-valid trajectories is:

$$r_{\text{rank}} = \begin{cases} r_{\text{rel}}, & f = 1,\ I_{\text{ans}} = 1,\ r_{\text{rel}} \leq \gamma, \\ r_{\text{rel}} + r_{\text{main}}, & f = 1,\ I_{\text{ans}} = 1,\ r_{\text{rel}} > \gamma, \\ r_{\text{main}}, & f = 1,\ I_{\text{ans}} = 0,\ r_{\text{rel}} = 0. \end{cases} \tag{13}$$

Format-invalid trajectories are excluded from ranker optimization, as described above. The design follows three cases. When $I_{\text{ans}} = 1$, a pseudo-relevant document exists in the candidate set. If r_{rel} is low ($r_{\text{rel}} \leq \gamma$), it indicates that the ranker fails to rank relevant documents effectively. In this case, we rely solely on r_{rel} for supervision, since incorporating r_{main} would introduce noise from the main agent's reasoning process. When $r_{\text{rel}} > \gamma$, the ranking is sufficiently strong, and we additionally incorporate r_{main} to capture whether the retrieved documents contribute to the final answer. When $I_{\text{ans}} = 0$, no pseudo-relevant document exists in the candidate set, and we rely on r_{main} as the only available signal.

In sum, the r_{rank} prioritizes ranking precision when a relevant document is available but poorly ranked, adds trajectory-level credit when the ranking is already strong, and falls back to answer correctness alone when no relevant document exists in the candidate set.

5.2 Experiments

5.2.1 Experimental Setup

Datasets. We evaluate on seven QA benchmarks spanning single-hop and multi-hop settings: PopQA [Mallen et al., 2023], Natural Questions (NQ) [Kwiatkowski et al., 2019], TriviaQA (TQA) [Joshi et al., 2017], HotpotQA [Yang et al., 2018], 2WikiMultiHopQA (2Wiki) [Ho et al., 2020], Musique [Trivedi et al., 2022], and Bamboogle [Press et al., 2023]. We use the test splits from Search-R1 [Jin et al., 2025] for consistent comparison. Performance is measured by token-level F1 score.

Baselines. We compare against Direct Inference, CoT [Wei et al., 2022], RAG [Lewis et al., 2020], Search-o1 [Li et al., 2025a], Search-R1 [Jin et al., 2025], ZeroSearch [Sun et al., 2025a], and two internal variants: (1) **Retrieval Only**, which removes the ranker and directly uses the top-K documents from the dense retriever; and (2) **Fixed Ranker**, which uses a ranker fine-tuned on a relevance-based IR dataset but kept frozen during RL training.

Implementation details. For the main agent, we evaluate both Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct [Yang et al., 2024] to assess performance across different model scales. The ranker in CoSearch is based on Qwen2.5-7B-Instruct. We use the 2018 Wikipedia dump [Karpukhin et al., 2020a] as the knowledge source and E5 base model [Wang et al., 2024] as the first-stage retriever. During rollout, each query generates $G = 8$ trajectories with up to 6 search calls per trajectory. The first-stage retriever returns $N = 50$ candidate passages, from which the ranker selects the top $K = 5$. Training uses a learning rate of 1×10^{-6}, rollout batch size of 512, and effective batch size of 128 for training, and sampling temperature of 1.0. For semantic grouping, we set the similarity threshold to $\delta = 0.8$ and minimum group size $k_{\text{min}} = 3$. The composite reward threshold is $\gamma = 0.5$ and the Hit@k cutoff set is $\mathcal{K} = \{1, 3, 5\}$. Further details on training data and Fixed Ranker training are provided in Appendix B.4.

Method	General QA			Multi-Hop QA				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	
Qwen2.5-7B-Instruct								
Direct Inference	0.279	0.504	0.215	0.292	0.304	0.118	0.372	0.298
CoT	0.198	0.456	0.150	0.244	0.264	0.095	0.310	0.245
RAG	0.420	0.689	0.387	0.371	0.244	0.148	0.296	0.365
Search-o1	0.345	0.526	0.248	0.316	0.286	0.126	0.422	0.324
Search-R1	0.556	0.681	**0.539**	0.576	0.547	0.272	0.558	0.533
ZeroSearch	0.562	0.693	0.521	0.575	0.559	0.291	0.572	0.539
Retrieval Only	0.564	0.728	0.508	0.572	0.556	0.320	0.574	0.546
Fixed Ranker	0.578	0.735	0.514	0.578	0.560	0.315	0.588	0.553
CoSearch	**0.582**	**0.747**	0.527	**0.595**	**0.587**	**0.334**	**0.601**	**0.568**
Oracle	0.680	0.764	0.684	0.626	0.584	0.393	0.652	0.626
Qwen2.5-3B-Instruct								
Direct Inference	0.198	0.343	0.168	0.202	0.249	0.055	0.068	0.180
CoT	0.191	0.363	0.159	0.197	0.229	0.070	0.100	0.187
RAG	0.317	0.478	0.291	0.371	0.193	0.110	0.058	0.223
Search-o1	0.329	0.501	0.351	0.225	0.128	0.096	0.216	0.236
Search-R1	0.458	0.637	0.460	0.450	0.397	0.179	0.414	0.428
ZeroSearch	0.451	0.642	0.471	0.455	0.387	0.191	0.431	0.433
Retrieval Only	0.465	0.651	0.478	0.454	0.406	0.189	0.436	0.440
Fixed Ranker	0.515	0.697	0.483	0.474	0.411	**0.195**	0.448	0.460
CoSearch	**0.538**	**0.714**	**0.489**	**0.478**	**0.428**	0.193	**0.457**	**0.471**
Oracle	0.548	0.749	0.612	0.589	0.510	0.308	0.589	0.558

Table 7: Main results (token-level F1) on seven QA benchmarks. Results are grouped by main agent backbone. The CoSearch ranker uses Qwen2.5-7B-Instruct in both settings. Best non-oracle results are in **bold**; second-best non-oracle results are underlined. Oracle (gray) is a diagnostic retrieval intervention that promotes answer-containing documents to rank 1.

5.2.2 Main Results

Table 7 presents the full results across all seven benchmarks. We observe that:

Joint training achieves the best average performance. CoSearch achieves an average F1 of 0.568 with the 7B agent, outperforming Search-R1 by a relative 6.6%, Retrieval Only by 4.0% and Fixed Ranker by 2.7%. The gains are broad, although some individual datasets are better served by Search-R1 or Fixed Ranker, indicating that jointly training the ranker provides a robust average benefit. Among non-RL baselines, RAG improves over direct inference by 22.5% (0.365 vs. 0.298), and Search-R1 further surpasses RAG by 46.0%

Configuration	Single-Hop	Multi-Hop	Avg
CoSearch (full)	**0.619**	**0.529**	**0.568**
(a) Reward signal			
w/o r_{main}	0.612	0.508	0.560
w/o r_{rel}	0.591	0.513	0.552
Replace Hit@k with nDCG@k	0.605	0.519	0.556
Replace Composite with LLM-as-judge	0.608	0.520	0.558
(b) Semantic grouping			
w/o semantic grouping	0.595	0.503	0.549
(c) Architecture			
Shared backbone	0.613	0.518	0.559
Cross-Encoder Ranker	0.601	0.501	0.551
3B ranker	*diverged (training crash)*		

Table 8: Ablation study. The main agent uses Qwen2.5-7B-Instruct. Scores are token-level F1 scores. Each row changes one component of the full model. Averages are computed over single-hop and multi-hop groups separately.

(0.533 vs. 0.365), confirming the value of iterative reasoning, yet our method improves further by optimizing retrieval itself.

Weaker agents benefit more from retrieval improvements. With a 3B main agent paired with a 7B ranker, CoSearch achieves a relative 10.0% gain over Search-R1, 7.0% over Retrieval Only, and 2.4% over Fixed Ranker. Compared to the diagnostic oracle intervention, CoSearch closes approximately 27% of the retrieval gap for the 7B agent and 26% for the 3B agent, confirming that end-to-end RL optimization meaningfully narrows the remaining retrieval gap.

5.2.3 Ablation Study

Table 8 reports ablation results. On reward design, removing the trajectory-level reward r_{main} reduces average F1 by a relative 1.4%, while removing the ranking-quality reward r_{rel} causes a larger drop of 2.8%, confirming that both components provide complementary supervision: r_{rel} supplies a direct signal on retrieval precision, while r_{main} captures whether retrieved documents actually contribute to final answer correctness. Replacing Hit@k with

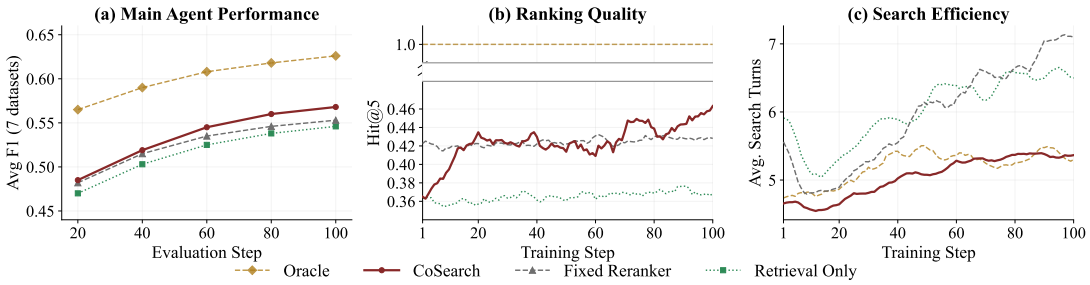


Figure 11: Training and evaluation dynamics across 100 steps. (a) Validation F1 (avg. over 7 datasets) every 20 steps. (b) Ranking quality (Hit@5) at each training step; y-axis broken to show Oracle $= 1.0$ above. (c) Average search turns per trajectory.

nDCG@k reduces performance by a relative 2.1%, and substituting the composite reward with an LLM-as-judge yields a 1.8% drop—both confirming that our composite design is effective and computationally lightweight.

Removing the semantic grouping and filtering strategy—treating all ranker calls sharing the same initial query q_0 as one group regardless of sub-query similarity—results in the largest single-component degradation of 3.3% relative (0.568 $\rightarrow$ 0.549). This confirms that mixing semantically heterogeneous sub-queries into a single GRPO group produces noisy advantage estimates that impede ranker learning, and that the filtering step described in §5.1.3 is critical for stable training.

Among architectural choices, a shared backbone degrades F1 by 1.6% due to conflicting ranking and reasoning gradients. A cross-encoder ranker drops 2.9%: its pointwise scoring is ill-suited for direct RL optimization. Scaling the ranker to 3B causes training to diverge entirely—the smaller model cannot reliably produce valid permutation outputs over 50 candidates (see Appendix B.1.2).

5.2.4 Analysis

Figure 11 compares all four systems across three dimensions over 100 training steps. Panels (a) and (b) reveal a clear positive correlation between ranking quality (Hit@5) and downstream F1, with our ranker starting below the Fixed Ranker but steadily surpassing

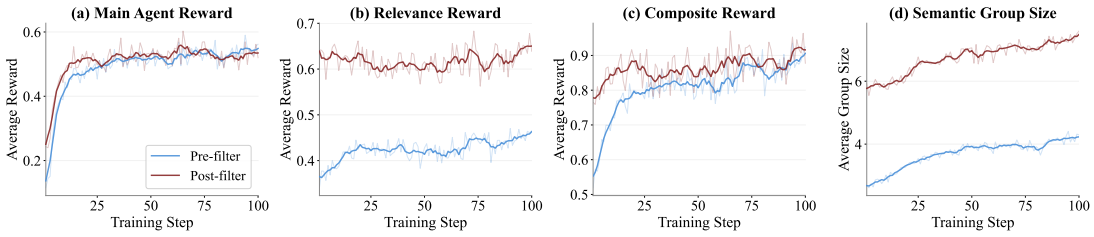


Figure 12: Ranker training dynamics across 100 steps. **Pre-filter**: statistics over all semantic groups; **post-filter**: statistics over groups that pass the minimum-size filter described in §5.1.3. (a) Main agent reward. (b) Relevance reward (Hit@5). (c) Composite reward. (d) Average semantic group size.

it during training. Panel (c) shows that CoSearch and Oracle require fewer search turns, suggesting higher-quality documents reduce the need for additional retrieval rounds.

Figure 12 tracks the ranker's training dynamics. In panels (a) and (c), pre-filter rewards eventually converge toward post-filter, suggesting the distributional gap is progressively mitigated. Panel (b) shows pre-filter relevance reward rising steadily (from $\sim$0.35 to $\sim$0.50), indicating genuine improvement in ranking ability. Panel (d) shows group size increasing monotonically, with post-filter roughly twice pre-filter, reflecting increasingly consistent sub-queries as the policy converges.

K	Retrieval Only	Fixed Ranker	CoSearch	Oracle
1	0.498 (−8.8%)	0.513 (−7.2%)	0.538 (−5.3%)	0.612 (−2.2%)
3	0.531 (−2.7%)	0.539 (−2.5%)	0.561 (−1.2%)	0.631 (+0.8%)
5	0.546	0.553	**0.568**	0.626
10	0.555 (+1.6%)	0.559 (+1.1%)	0.571 (+0.5%)	0.639 (+2.1%)

Table 9: Effect of the number of ranked documents K on Avg F1. The ranker selects the top K from N=50 first-stage candidates. K=5 (bold) is the default used in Table 7. Percentages in parentheses indicate relative change from K=5 for each method.

Table 9 examines how K affects performance. When K is reduced from 5 to 1, CoSearch degrades by only 5.3%, comparable to Oracle (−2.2%) and far less than Retrieval Only (−8.8%) and Fixed Ranker (−7.2%). Notably, CoSearch at K=3 (0.561) already surpasses Fixed Ranker at K=10 (0.559), and increasing K from 5 to 10 yields diminishing returns across all methods (+0.5% to +1.6%).

5.3 Summary

In this chapter, we identified retrieval quality as a key bottleneck in agentic search, with an oracle experiment revealing up to +26.8% relative F1 improvement, and presented CoSearch, which jointly trains a reasoning agent and a generative ranker via reinforcement learning with a semantic grouping strategy and composite reward. Experiments on seven QA benchmarks show +6.6% relative F1 over Search-R1 with a 7B agent and +10.0% with a 3B agent, demonstrating that joint optimization of reasoning and retrieval is both feasible and effective. These results show that retrieval for search agents should be optimized not only for standalone relevance, but also for how retrieved evidence supports multi-step reasoning.

CHAPTER 6

CONCLUSION

6.1 Summary

The first contribution of the dissertation is to make Generative Information Retrieval (GIR) more practical for large-scale corpora. GIR represents documents with identifiers and retrieves documents by generating those identifiers. Early GIR systems showed the promise of this formulation, but they struggled on large retrieval benchmarks and relied on expensive autoregressive decoding. Chapters 3 and 4 address these two limitations.

Chapter 3 introduced RIPOR to improve the scalability of GIR. RIPOR creates relevance-aware DocIDs through residual quantization and trains the generative model with prefix-oriented ranking optimization. On MSMARCO (8.8M passages), RIPOR substantially improves over prior GIR methods and narrows the gap to strong dense retrievers.

Chapter 4 introduced PAG to improve decoding efficiency. PAG uses set-based lexical identifiers to estimate document-level relevance before sequential decoding, providing a planning-ahead signal for constrained beam search. This reduces the dependence on large beam sizes and improves query latency while maintaining strong retrieval effectiveness.

The second contribution studies retrieval optimization for search agents. In this setting, an LLM-based search agent answers complex questions by interleaving reasoning with search: it issues queries, reads retrieved evidence, and decides what to search for next over multiple turns. Chapter 5 presented CoSearch, a framework that jointly trains the reasoning agent and a generative document ranker with reinforcement learning. Across seven QA benchmarks, CoSearch improves over strong agentic-search baselines, showing that retrieval can be optimized as part of the reasoning system rather than treated only as a fixed external tool.

Thus, the dissertation first improves GIR as a standalone large-scale retrieval model,

and then studies how retrieval can be jointly optimized with reasoning in agentic search.

6.2 Limitations and Future Directions

This dissertation also leaves several open problems. One important direction is to rethink what DocIDs should represent. In Chapters 3 and 4, DocIDs are constructed mainly for document retrieval: they should help the model generate relevant documents for a query and avoid pruning errors during decoding. If GIR is eventually trained together with language modeling in a shared backbone, DocIDs may need to play a richer role. They may need to serve not only as addresses of documents, but also as discrete symbols that preserve enough semantic meaning for reasoning. For example, a model that reasons before retrieval may need to use DocID tokens or prefixes as references to entities, topics, attributes, or evidence types. A challenge for studying this direction is to build suitable benchmarks and datasets that can evaluate whether joint modeling of retrieval and language tasks actually improves both retrieval quality and downstream reasoning.

This also raises a representation challenge. A DocID optimized only for retrieval may not be the best DocID for reasoning, generation, or recommendation. Retrieval-oriented DocIDs should separate relevant from irrelevant documents; reasoning-oriented identifiers should preserve interpretable semantic structure; recommendation-oriented identifiers may need to encode behavioral or collaborative signals. It is not yet clear whether these objectives are compatible, conflicting, or should be assigned to different parts of the identifier. Future work should study constructing DocIDs that balance retrieval effectiveness with semantic structure, and how to train models so that the meaning of these identifiers can be used by the generative backbone rather than only decoded as final outputs.

Another direction is to move beyond text-only document retrieval. Many real search and recommendation systems involve products, media, or entities whose relevance depends on multiple sources of information. In product search, for example, textual descriptions are only one part of the item representation; images, brands, prices, availability, reviews, and

other metadata may all affect what a user considers relevant. Future GIR systems should study how to build identifiers that integrate such heterogeneous evidence. This may require multi-part DocIDs, modality-specific codebooks, or training objectives that preserve both semantic content and task-specific signals such as personalization, visual similarity, or structured product constraints.

Decoding efficiency remains central as retrieval collections grow. PAG reduces the dependence on large beam sizes, but GIR still searches over a large discrete identifier space at query time. Future work may combine autoregressive generation with more efficient search procedures, such as learned pruning, approximate prefix search, cached document-level scores, speculative decoding, or hybrid retrieval structures. The goal is not only to reduce average latency, but also to make GIR decoding predictable and reliable enough for practical search systems.

A broader long-term question is whether GIR can develop its own pre-training and post-training paradigm. Current GIR systems are usually trained for specific retrieval benchmarks and often use relatively small backbones, such as T5-base. A more ambitious direction is to train generative retrieval models across retrieval, recommendation, and language tasks, and to study how performance changes as both model size and training data scale. Such work would clarify whether GIR has scaling behavior comparable to modern language models, and whether unified generative modeling can support end-to-end training across search, recommendation, multimodal retrieval, and natural-language generation.

For agentic search, Chapter 5 optimizes the generative ranker used by the agent, but the first-stage retriever remains fixed. This leaves candidate generation outside the learning loop. A natural next step is to optimize the retriever itself from agent interactions, so that it can adapt to the sub-queries produced during multi-turn search before reranking begins. This would make the full retrieval system, rather than only the final ranking stage, responsive to the evolving information needs of the search agent.

Another open problem is the training signal for retrieval in agentic search. Although

Chapter 5 combines relevance feedback with final answer correctness, the optimization is still strongly shaped by outcome-level rewards. A correct or incorrect final answer may depend on many factors beyond one retrieval decision, making credit assignment difficult. With more computational resources, future work could introduce richer step-level rewards that evaluate intermediate search actions, evidence usefulness, query reformulation, and document ranking quality. More systematic ablations over these reward components would clarify which signals most reliably improve retrieval inside multi-step reasoning.

The other direction is to reduce the separation between the search agent and the retrieval model. Chapter 5 uses separate model backbones for the main agent and the generative ranker, which gives strong performance but increases deployment cost because each search step may require running both models. The shared-backbone ablation in Chapter 5 shows that naive sharing hurts performance, suggesting interference between reasoning and ranking objectives. Future work should study how to share computation without losing accuracy, for example through better training schedules, modular adapters, or architectures that separate reasoning and retrieval behavior while still using a common backbone.

Current agentic-search benchmarks used in the dissertation also remain relatively short-horizon. The datasets studied in Chapter 5 require multi-hop reasoning, but the number of search calls is small compared with more open-ended deep research tasks. In those settings, an agent may need tens or even hundreds of tool calls, accumulating far more retrieved evidence than can be handled by simply appending everything to the context. Long-horizon agentic search therefore raises a context-management problem: the system must decide what to keep, summarize, discard, revisit, and cite as the investigation progresses. Future work on search agents should study retrieval together with memory and context management, so that the agent can sustain long research trajectories without being overwhelmed by its own intermediate evidence.

Overall, Chapters 3 and 4 show that GIR can be made more scalable and efficient on large-scale retrieval benchmarks. Chapter 5 broadens the dissertation to agentic search,

where retrieval quality is measured by its effect on reasoning. These results leave open both system-level questions for GIR and learning questions for search agents.

CHAPTER A

SUPPLEMENTARY MATERIAL FOR CHAPTER 3

A.1 The Comparison Between RQ and PQ

Table 2 demonstrates a significant decline in retrieval effectiveness when substituting residual quantization (RQ) with product quantization (PQ). This decline can be attributed to PQ's limited capability in capturing the hierarchical structure crucial for the encoding document semantics in generative retrieval models. Hierarchical structures range from coarse-grained to fine-grained, significantly influencing the performance of (constrained) beam search.

To assess the abilities of RQ and PQ in encoding coarse-grained document semantics, we analyze labeled queries from the TREC-DL 2019 dataset, where each query is associated with an average of 58.16 relevant documents. We compute the count of unique prefixes of lengths $\{1, 2, 3, 4\}$ for relevant document sets corresponding to each query. Table 10 presents these counts, showing that RQ uses fewer unique prefixes than PQ at early positions, indicating a more efficient representation of documents with similar relevance.

Table 10: Average unique number of prefixes for relevant documents.

Prefix Length	RQ	PQ
1	8.20	18.70
2	21.70	38.63
3	35.50	46.81
4	45.30	50.77

For finer-grained semantic distinctions, we analyze the diversity of codes at each position for the same set of documents. Figure 13 plots the number of unique codes per position, showing that RQ employs fewer codes in initial stages but diversifies more in later stages, enhancing fine-grained semantic differentiation.

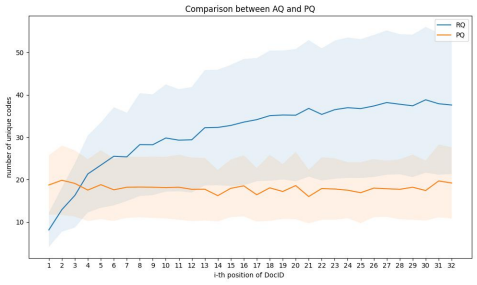


Figure 13: Unique number of codes per position for documents relevant to the same query.

A.2 DocID Repetition

We only apply the residual quantization for constructing DocIDs. When L is set to 32 and V is set to 256, the encoding space is large enough to assign almost every document a unique DocID. Based on the statistics, we construct 8,826,152 unique DocIDs for 8,841,823 documents in the MS MARCO dataset. The percentage of DocIDs that are mapped to more than 1 document only accounts for 0.2% of the total number of DocIDs. Notably, the MS MARCO dataset is not fully clean, i.e., it might contain duplicate documents. For example, we have a DocID: 155-180-0-122-96-51-148-26-167-114-166-6-58-253-246-16-248-166-240-172-161-176-200-226-1-222-223-151-60-250-105-203 that maps to two documents with id '13184', '13188'. The contents of these two documents are duplicates, which is:

Ross Cameron is 52 years old. To be more precise (and nerdy), the current age as of right now is 18981 days or (even more geeky) 455544 hours. That's a lot of hours!

For those DocIDs that have more than 1 document, we quantitatively compute the term overlap rate between these documents. For each duplicate (non-unique) DocID c^{dup}, we assume that all documents belonging to it are defined as $\mathcal{D}(c^{dup})$, the unigram set for each document d is set to $\text{Uni}(d)$, then the term overlap rate r is:

$$r = \frac{1}{|\mathcal{D}(c^{dup})|(|\mathcal{D}(c^{dup})| - 1)} \sum_{d_1, d_2 \in \mathcal{D}(c^{dup})} \frac{|\text{Uni}(d_1) \cap \text{Uni}(d_2)|}{|\text{Uni}(d_1) \cup \text{Uni}(d_2)|}$$

Table 11: The training time of each round in the rank-oriented fine-tuning phase.

Rounds	MRR@10	Recall@10	Time
Initial FT	.280	.475	12 h
prefix-oriented FT	.324	.543	56 h
self-neg FT	.333	.562	14 h

The average of term overlap ratios r across all duplicated DocIDs is 92.5%. Considering that the average number of unigrams per document is 42.2, the average number of overlapped terms between any two documents belonging to the same DocID is 39. When the duplicated DocID is retrieved, all the documents mapped to it will be retrieved and assigned the same scores.

A.3 Training Time of RIPOR

We use 8 A100 GPUs with 40GB memory. Our training contains 3 phases: 1) DocID initialization; 2) Seq2seq pre-training; 3) rank-oriented fine-tuning. DocID initialization takes 34 hours, Seq2seq pre-training takes 20 hours, and rank-oriented fine-tuning contains three rounds of training. We report performance and training time for the three rounds of fine-tuning on MSMARCO Dev in Table 11.

Based on the table, each fine-tuning round can lead to a further performance boost. We realize that the most time-consuming round is *prefix-oriented fine-tuning*, since it requires progressive training with 4 steps $i = 4, 8, 16, 32$, and each step takes 14 hours. If we want to reduce the total training time, we can remove the progressive training in the prefix-oriented fine-tuning round, meaning that we only use the multi-objective loss function for $i = 32$. Table 12 shows the performance and training time for the new variant.

A.4 Performance on a Smaller-Scale Sampled Dataset

Following the methodology of prior work [Pradeep et al., 2023a], we also create a scaled-down version encompassing 1M passages. Initially, we include all passages relevant

Table 12: The training time of each round when removing the progressive training in the prefix-oriented fine-tuning round.

Rounds	MRR@10	Recall@10	Time
Initial FT	.280	.475	12 h
prefix-oriented FT - prog. train	.319	.537	14 h
self-neg FT	.333	.562	14 h

to the 532K training queries and the 7K Dev set queries, summing to 522K passages. The rest are selected at random from the main collection, totaling 1M passages. The results are shown in Table 13. We report these results for the sake of comparison with prior work [Pradeep et al., 2023a], but do not recommend future work to use MSMARCO-1M, since it is artificially constructed, in which 51.6% of documents are covered by the train query set. In comparison, only 5.8% of documents are covered by the train query set in the real MSMARCO-8.8M dataset.

Table 13: Experimental results on the MSMARCO-1M subset. Results are reported on MSMARCO Dev and TREC Deep Learning Track Data. Highest generative retrieval performances are boldfaced. Superscript $*$ denotes statistically significant improvement compared to all generative retrieval baselines. Superscripts $^{\triangle}$ and $^{\triangledown}$ denote significantly higher and lower performance compared to RIPOR (t-test with Bonferroni correction, p-value < 0.01). For dense retrieval models, the HNSW [Malkov and Yashunin, 2016] index is used for ANN search.

Model	**MSMARCO Dev**		**TREC DL 2019**		**TREC DL 2020**	
	MRR@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10
Generative Retrieval						
DSI	.132	.201	.093	.041	.098	.035
DSI-QG	.508	.726	.438	.351	.420	.327
NCI-QG	.511	.720	.446	.354	.436	.326
SEAL	-	-	-	-	-	-
MINDER	-	-	-	-	-	-
LTRGR	-	-	-	-	-	-
RIPOR (ours)	**.580***	**.793***	**.531***	**.438***	**.529***	**.412***
Sparse and Dense Retrieval Models (For Reference)						
BM25	.418$^{\triangledown}$	.625$^{\triangledown}$	.270$^{\triangledown}$	.050$^{\triangledown}$	.279$^{\triangledown}$	.058$^{\triangledown}$
DPR	.542$^{\triangledown}$	.775$^{\triangledown}$	.520$^{\triangledown}$	.344$^{\triangledown}$	.490$^{\triangledown}$	.335$^{\triangledown}$
ANCE	.547$^{\triangledown}$	.779$^{\triangledown}$	.522$^{\triangledown}$	.336$^{\triangledown}$	.506$^{\triangledown}$	.340$^{\triangledown}$
MarginMSE	.556$^{\triangledown}$	.781$^{\triangledown}$	.535$^{\triangle}$	.406$^{\triangledown}$	.539$^{\triangle}$	.392$^{\triangledown}$
TAS-B	.573$^{\triangledown}$	.789$^{\triangledown}$	.545$^{\triangle}$	.423$^{\triangledown}$	.538$^{\triangle}$	.413

CHAPTER B

SUPPLEMENTARY MATERIAL FOR CHAPTER 5

B.1 Agent Prompts

B.1.1 Main Agent Prompt

The main reasoning agent operates in a ReAct-style loop with a single search tool. At each turn, the agent outputs a <reason> block followed by either a <tool_call> block (to issue a search query) or an <answer> block (to finalize its response). The full prompts used across all experiments are presented below.

```
You are a tool-augmented search agent for wiki-based factoid question answering.

Your task is to answer questions drawn from Wikipedia-style datasets.
The final answer is evaluated using exact match (EM) or token-level F1, so it must be short and precise.

You have ONE tool available:
- search(query: string) -> returns a list of Wikipedia passages

============================================================
CRITICAL OUTPUT FORMAT (MUST FOLLOW EXACTLY)
============================================================

For EVERY assistant turn, you MUST output EXACTLY TWO TAG BLOCKS in this order:

1) <reason> ... </reason>
2) EITHER:
   (A) <tool_call> ... </tool_call>
   OR
   (B) <answer> ... </answer>

No other text is allowed outside these tags.
Do NOT output <tool_response>. The environment will provide tool results separately.

Allowed patterns:
- <reason> ... </reason>
```

<tool_call> ... </tool_call>

- <reason> ... </reason>
<answer> ... </answer>

If you violate the format, your output is invalid.

==========
TOOL CALL JSON SCHEMA (STRICT)
==========

When calling the tool, the <tool_call> block MUST contain ONLY a valid JSON object:

<tool_call>
{
"name": "search",
"arguments": {
"query": "<string>"
}
}
</tool_call>

Rules:
- "name" MUST be exactly "search"
- "arguments" MUST be an object
- "query" MUST be a single string
- Do NOT add extra keys
- Do NOT wrap JSON in Markdown
- Do NOT include comments, trailing commas, or natural language

==========
GENERAL TOOL USAGE
==========

Use the search tool whenever additional evidence would help you determine the correct answer.
If you believe you already have sufficient information to answer correctly, answer directly.

You may use multiple search calls across turns.

==========
SEARCH GUIDELINES

==========

- Write search queries that are clear and specific to what you want to confirm or find.
- After receiving evidence, reassess whether you can answer; if not, search again with a refined query.

==========
REASONING CONTENT REQUIREMENTS
==========

- Do NOT include tool JSON inside <reason>.
- Do NOT include <tool_call> or <answer> tags inside <reason>.

==========
ANSWER REQUIREMENTS (STRICT: SHORT ANSWER)
==========

Inside <answer>, you MUST:
- Output ONLY the final answer string
- Do NOT include explanations, reasoning, or extra text
- Do NOT include citations, sources, or formatting
- Use a concise canonical form (Wikipedia-style when possible)

Examples of valid answers:
- Paris
- 1997
- George Washington
- The Lord of the Rings

If the expected answer type is a person/place/organization/title/date, output only that span.
If multiple surface forms are possible, output the most standard form.

==========
INTEGRITY
==========

- Do not fabricate facts.
- If you are uncertain, use search to verify.
- If evidence is conflicting, search again with a query that resolves the conflict.

==========
BEGIN

```
============================================================

Question: {question}
```

B.1.2 Ranker Prompt

The ranker receives the original user query q_0, the sub-query q_t issued by the main agent at step t, and the top-N candidate documents $\mathcal{D}_N$ returned by the dense retriever. It outputs a <reason> block containing its relevance analysis, followed by a <rerank> block listing the selected K document indices in descending order of relevance (e.g., $[3] > [1] > [5] > [2]$). The full prompt template is shown below.

```
You are a document ranking agent assisting a search-augmented reasoning system.

You will be given:
- An original user question
- A sub-query issued by the reasoning agent at the current search step
- A list of N candidate documents retrieved by a dense retriever,
  each labeled [1], [2], ..., [N]

Your task is to select the K most relevant documents and rank them in descending
order of relevance to help the reasoning agent answer the original question.

============================================================
CRITICAL OUTPUT FORMAT (MUST FOLLOW EXACTLY)
============================================================

You MUST output EXACTLY TWO TAG BLOCKS in this order:

1) <reason> ... </reason>
2) <rerank> ... </rerank>

No other text is allowed outside these tags.

============================================================
REASON BLOCK
============================================================
```

Inside <reason>, analyze the relevance of each candidate document.
Consider whether the document provides factual evidence that can help answer the
original question, taking the sub-query into account as additional context.

==========
RERANK BLOCK
==========

Inside <rerank>, list exactly K document indices in descending order of relevance:

<rerank>[id1] > [id2] > ... > [idK]</rerank>

Rules:
- Use only the provided document indices (e.g., [1], [3], [7])
- Select exactly K indices; do NOT select more or fewer
- Do NOT duplicate indices
- Do NOT include any explanation inside <rerank>

==========
BEGIN
==========

Original Question: {original_question}
Sub-Query: {sub_query}

Candidate Documents:
{documents}

B.1.3 Ranker Input/Output Examples

We present two concrete examples of ranker inputs and outputs at training step 100. Each example shows the original question, the sub-query issued by the main agent at a single retrieval step, the first five of the N=50 candidate passages (truncated), and the ranker's full output comprising a relevance analysis (<reason>) followed by a ranked list of K=5 document indices (<rerank>).

B.1.3.0.1 Example 1: answer-containing document present ($I_{\text{ans}} = 1$), high composite reward.

Golden answer: *Harpswell*. answer_in_docs = true, $r_{\text{rel}} = 1.00$, $r_{\text{main}} = 1.00$, $r_{\text{total}} = 2.00$.

Input (first 5 of 50 candidates shown):

```
Original Question: What town has an island with a community building built in 1911-12?
Sub-Query: island with community building built 1911-12

Passages (50 total):
[1] "Bailey Island Library Hall"
Bailey Island Library Hall (locally just Library Hall) is a historic community building
at 2167 Harpswell Island Road...
[2] "Bumpkin Island"
Bumpkin Island, also known as Round Island, Bomkin Island, Bumkin Island, or Ward's Island,
is an island in the Hingham Bay area of the Bosto...
[3] "Chocorua Island Chapel"
Although the camp had fallen into disrepair by that date, the chapel continued to provide
worship services for visitors and area residents. Ev...
[4] "Gooden Grant House"
octagonal tower with turreted roof projects at the southwest corner, and a
partially-enclosed single-story porch wraps around the west and south sides...
[5] "Cherry Grove Community House and Theatre"
almost 400 other such sites have been identified as candidates. Cherry Grove Community
House and Theatre The Cherry Grove Communit...
... [45 more candidates not shown]
```

Output:

```
<reason>
The Initial Query asks for a town with an island that has a community building built in
1911-12. Passage [1] directly mentions a community building built in 1911-12 and locates
it on Harpswell Island Road, making it the top choice. Passage [48] also fits the criteria
but contains less detailed information. Passages [2], [32] discuss islands but do not
mention a community building from that period. Passage [39] describes a small island with
no matching structure.
</reason>
<rerank>[1] > [48] > [2] > [32] > [39]</rerank>
```

B.1.3.0.2 Example 2: answer-containing document absent ($I_{\text{ans}} = 0$), ranker relies on r_{main}.

Golden answer: *yes*. answer_in_docs = false, $r_{\text{rel}} = 0.00$, $r_{\text{main}} = 1.00$, $r_{\text{total}} = 1.00$.

Input (first 5 of 50 candidates shown):

```
Original Question: Are Naked Obsession and New York (1916 Film) from the same country?
Sub-Query: New York (1916 Film)

Passages (50 total):
[1] "New York (1916 film)"
New York (1916 film) New York is a lost 1916 American silent comedy drama film directed by
George Fitzmaurice and starring Florence Reed...
[2] "New York (1916 film)"
thus Oliver King becomes a benedict, and Reel 3 contains two views of a nude model.
New York is a lost 1916 American silent comedy drama film...
[3] "Lights of New York (1916 film)"
Lights of New York is a 1916 American silent drama film directed by Van Dyke Brooke.
Produced by the Vitagrap...
[4] "Lights of New York (1916 film)"
with him anyway. Lights of New York (1916 film) Lights of New York is a 1916 American
silent drama film directed by Van Dyke Brooke...
[5] "The Pride of New York"
German officer and nurse in room where bed is shown including taking nurse to room and
excluding other young woman...
... [45 more candidates not shown]
```

Output:

```
<reason>
The Initial Query asks whether Naked Obsession and New York (1916 Film) are from the same
country. The Sub-Query targets New York (1916 Film). Passages [1] and [2] directly
describe this film and confirm its American origin, making them the top choices. Passages
[3] and [4] describe a different 1916 film (Lights of New York) that is also American but
less directly relevant. Passage [5] discusses yet another unrelated film and ranks lowest.
</reason>
<rerank>[1] > [2] > [3] > [4] > [5]</rerank>
```

B.2 Two-Stage Retrieval System

At each retrieval step t, the two-stage retrieval system takes the original query q_0, the sub-query q_t issued by the main agent, and the corpus $\mathcal{C}$, and returns an observation o_t consisting of K ranked documents. The first stage is a fixed dense retriever that efficiently recalls N candidate documents; the second stage is the generative ranker $\pi_{\theta_{\text{gr}}}$, which reasons about candidate relevance and outputs an explicit ranked list. Algorithm 2 gives the full procedure. For the prompt format and concrete input–output examples of the generative ranker, see Appendix B.1.2.

Algorithm 2 Two-Stage Retrieval at Step t

Require: original query q_0, sub-query q_t, corpus $\mathcal{C}$, retrieval size N, ranking size K
Ensure: observation $o_t = \mathcal{D}_K$ (top-K ranked documents)

$\mathcal{D}_N \leftarrow \text{DenseRetrieve}(q_t, \mathcal{C}, N)$ {fixed retriever}
$I_{\text{ans}} \leftarrow \mathbf{1}[\text{gold answer} \in \mathcal{D}_N]$ {used for composite reward, §5.1.4}
$\textit{prompt} \leftarrow \text{BuildPrompt}(q_0, q_t, \mathcal{D}_N, K)$
$\textit{output} \leftarrow \pi_{\theta_{\text{gr}}}(\textit{prompt})$ {generative ranker produces ranked list}
$\mathcal{D}_K \leftarrow \text{ParseRanking}(\textit{output}, \mathcal{D}_N, K)$ {extract top-K documents}
return $\mathcal{D}_K$

B.3 Semantic Grouping Algorithm

Applying GRPO to the ranker requires grouping ranker calls with the same input prompt. Since sub-queries vary across rollouts, we perform a two-level grouping: first split by whether the candidate set contains the gold answer (I_{ans}), then cluster within each split by token-level F1 similarity. Figure 14 illustrates the procedure for a simplified example with $G{=}5$ rollouts and up to 5 search steps per trajectory; in our experiments we use $G{=}8$ rollouts with up to 6 search steps. Algorithm 3 gives the full procedure.

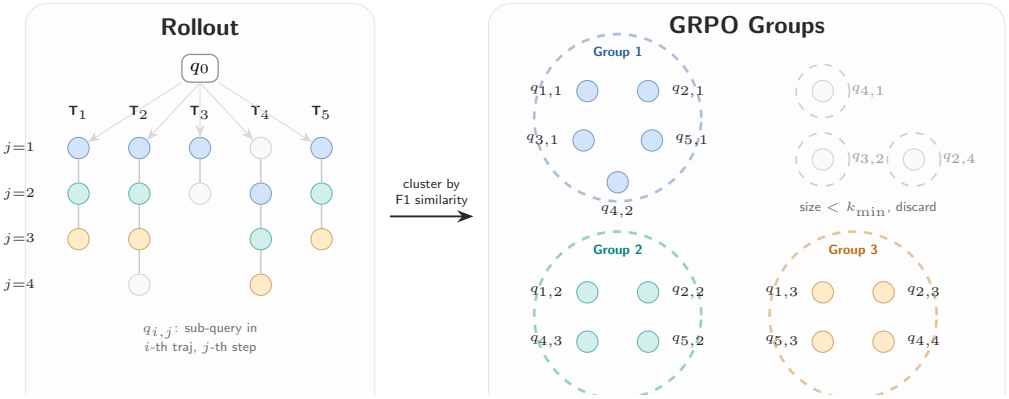


Figure 14: Semantic grouping for GRPO training. The illustration shows $G{=}5$ rollout trajectories with up to 5 search steps each. Left: each dot is a sub-query $q_{i,j}$ (trajectory i, step j) colored by group membership. Right: all sub-queries are first split by answer availability and then clustered by token-level F1 similarity into GRPO groups; gray singletons (size $< k_{\text{min}}$) are discarded before optimization.

B.4 Additional Implementation Details

B.4.0.0.1 Training data.

The RL training set consists of 51,200 questions drawn from four datasets: Natural Questions (20,480 questions, 40%), HotpotQA (14,220, 28%), Musique (9,000, 18%), and 2WikiMultiHopQA (7,500, 15%). All questions are sampled from each dataset's official training split. For evaluation, we use the official test split of each benchmark, or the evaluation split when no test split is available.

B.4.0.0.2 Fixed Ranker training.

The Fixed Ranker baseline is trained as follows. We randomly sample 50K queries from MS MARCO and 50K from Natural Questions, then use e5-base to retrieve the top-50 documents for each query. Queries for which no relevant document appears in the top-50 are discarded, yielding 92,160 training queries. The ranker is trained with RL using the same Hit@$\{1, 3, 5\}$ composite reward and the same generative listwise output format as the jointly trained ranker in CoSearch.

Algorithm 3 Semantic Grouping for Ranker GRPO Training

Require: ranker calls $\mathcal{O}$ (from G rollouts of q_0), similarity threshold δ, minimum group size k_{min}

Ensure: UID-labeled ranker calls ready for GRPO

// Step 1: Split by answer availability
Partition $\mathcal{O}$ into $\mathcal{O}_{\text{easy}}$ ($I_{\text{ans}} = 1$) and $\mathcal{O}_{\text{hard}}$ ($I_{\text{ans}} = 0$)
for each split $\mathcal{B} \in \{\mathcal{O}_{\text{easy}}, \mathcal{O}_{\text{hard}}\}$ **do**
  // Step 2: Greedy cluster by token-level F1
  clusters $\leftarrow \{\}$ {cluster_id $\rightarrow$ representative sub-query}
  for each call $o \in \mathcal{B}$ **do**
    $q \leftarrow$ Normalize(o.sub_query)
    matched $\leftarrow$ False
    for each $(c,\, q^{\text{rep}}) \in$ *clusters* **do**
      if $\text{F1}_{\text{token}}(q,\, q^{\text{rep}}) \geq \delta$ **then**
        assign $o \rightarrow$ cluster c; *matched* $\leftarrow$ True; **break**
      end if
    end for
    if not *matched* **then**
      create new cluster with q as representative; assign $o \rightarrow$ new cluster
    end if
  end for
  // Step 3: Discard small groups
  Remove all clusters with |cluster| $< k_{\text{min}}$
  // Step 4: Assign UIDs
  for each surviving cluster c **do**
    uid $\leftarrow \{\text{main_uid}\}_\{\text{easy}|\text{hard}\}_\text{cluster}_\{c\}$
    assign uid to all $o \in c$
  end for
end for

B.5 Semantic Grouping Examples

Tables 14 and 15 present 20 sampled initial queries from the step-100 rollout along with their sub-queries after semantic grouping and filtering. For each initial query, the G=8 rollout trajectories produce a pool of sub-queries; these are first split by answer availability and then clustered by token-level F1 similarity with threshold δ=0.8. Column **C** indicates the cluster index assigned to each sub-query. The table illustrates that multi-hop questions naturally produce multiple distinct semantic clusters (e.g., one cluster for each sub-question in a two-hop chain), while simpler questions tend to produce a single large cluster of near-

paraphrases.

Table 14: Sub-queries after semantic grouping and filtering (Part 1 of 2). The examples are from training step 100. For each initial query, sub-queries from $G{=}8$ rollouts are first partitioned by answer availability and then clustered by token-level F1 similarity ($\delta{=}0.8$). Column **C** denotes the cluster index.

Initial Query	**C**	**Sub-query**
who has played the most state of origins	0	who has played the most state of origins
	1	current record holder for most State of Origin appearances
	2	most State of Origin appearances
	2	most appearances in State of Origin
what is the host nation of the 2004 summer olympic g...	0	host nation of 2004 Summer Olympic Games
	1	host nation of 2004 Summer Olympics
	1	host nation of the 2004 Summer Olympics
What is the birth city of Blessed John the Fool-For-...	0	capital of former Soviet Union and country with AGT show
	0	capital of the former Soviet Union and country with AGT

Continued on next page

Table 14: Sub-queries after semantic grouping and filtering (Part 1 of 2, continued).

Initial Query	C	Sub-query
	1	Blessed John the Fool-For-Christ birth city
	1	Blessed John the Fool-For-Christ birth city Azerbaijan
	1	Blessed John the Fool-For-Christ birth city Baku Azerbaijan
What race is the majority of the population of the c...	0	The Unbeatables I production country
	1	The Unbeatables I
	1	The Unbeatables I produced
Who plays the writer who mentioned The Angel Pub in ...	0	Charles Dickens The Angel Pub
	0	Charles Dickens The Angel Pub writings
	0	Charles Dickens mentioned The Angel Pub
	1	The Man Who Invented Christmas
Are Naked Obsession and New York (1916 Film) from th...	0	Naked Obsession
	1	New York (1916 Film)

Continued on next page

Table 14: Sub-queries after semantic grouping and filtering (Part 1 of 2, continued).

Initial Query	C	Sub-query
Standard Chartered Bank, who sponsors the annual Hon...	0	Standard Chartered Bank headquarters
	1	Standard Chartered Bank sponsors Hong Kong Marathon
on what day of the year is the sun near the star reg...	0	when is the sun near Regulus (Alpha Leo) on the calendar
	0	when is the sun near Regulus (Alpha Leo) on the exact date
	0	when is the sun near Regulus (Alpha Leo) on the year
	1	day of the year when the sun is near Regulus (Alpha Leo)
	1	day of the year when the sun is near Regulus (alpha leo)
	1	exact day of the year when the Sun is near Regulus (Alpha Leo)
who did the congress send to london as a minister in...	0	who did the congress send to london as a minister in 1784

Continued on next page

Table 14: Sub-queries after semantic grouping and filtering (Part 1 of 2, continued).

Initial Query	C	Sub-query
	1	who was sent as a minister to london by the congress in 1784
	1	who was sent as a minister to london in 1784
	1	who was sent as minister to london by congress in 1784
In what season of the 2017 year is the Netflix Germa...	0	Dark (TV series) 2017 release season
	0	Dark (TV series) release schedule 2017
	1	Dark (Netflix German series) season release
	1	Dark Netflix German series 2017 release season
	1	Dark Netflix German series release season

Table 15: Sub-queries after semantic grouping and filtering (Part 2 of 2). The examples are from training step 100. For each initial query, sub-queries from $G{=}8$ rollouts are first partitioned by answer availability and then clustered by token-level F1 similarity ($\delta{=}0.8$). Column **C** denotes the cluster index.

Initial Query	**C**	**Sub-query**
What part did The King of Hollywood play in China Seas?	0	The King of Hollywood China Seas
	0	The King of Hollywood China Seas part
	1	Clark Gable role in China Seas
	1	Clark Gable's role in China Seas
What is the Izzo (H.O.V.A.) performer's record label?	0	Izzo (H.O.V.A.) performer record label
	1	Jay-Z current record label
	1	Jay-Z record label
Where was the place of death of the director of film...	0	director of film When A Man Sees Red 1934
	0	director of film When A Man Sees Red 1934 Pursued

Continued on next page

Table 15: Sub-queries after semantic grouping and filtering (Part 2 of 2, continued).

Initial Query	C	Sub-query
	0	director of film When A Man Sees Red 1934 version
	1	Frank Ellis death place
	1	Frank Ellis place of death
	1	place of death Frank Ellis
Who was born later, Dani Pacheco or Agnė Čepelytė?	0	Dani Pacheco birth year
	1	Agnė Čepelytė birth year
Who is the operator of Embassy of Northern Cyprus in...	0	Embassy of Northern Cyprus in Istanbul operator
	0	Embassy of Northern Cyprus in Kemerhisar operator
	0	Embassy of Northern Cyprus in Samsun operator
What town has an island with a community building bu...	0	island with community building built 1911-12

Continued on next page

Table 15: Sub-queries after semantic grouping and filtering (Part 2 of 2, continued).

Initial Query	C	Sub-query
	0	town with an island and a community building built in 1911-12
	0	town with island and community building built in 1911-12
The M66 is a motorway in Lancashire and Greater Manc...	0	M66 motorway in Lancashire and Greater Manchester
	0	M66 motorway in Lancashire and Greater Manchester, England
who plays the queen of hearts in alice and wonderland	0	who plays the Queen of Hearts in Alice and Wonderland
	0	who plays the Queen of Hearts in Alice in Wonderland
	0	who plays the Queen of Hearts in recent Alice in Wonderland films
What nationality is the director of film Porky'S Rev...	1	James Komack nationality

Continued on next page

Table 15: Sub-queries after semantic grouping and filtering (Part 2 of 2, continued).

Initial Query	C	Sub-query
What presenter of Market Kitchen has won a Guild of ...	0	Market Kitchen presenter Guild of Food Writers award
	0	Market Kitchen presenter who won Guild of Food Writers award
	0	Market Kitchen presenter won Guild of Food Writers award

B.6 Oracle Retrieval Construction

The oracle retrieval experiment (Table 16) measures the performance gap caused by imperfect retrieval. Unlike post-hoc analysis, the oracle trajectory is constructed *online* during rollout: the agent's reasoning at each step is conditioned on the (potentially modified) observation, so improved retrieval at step t influences all subsequent reasoning and sub-queries.

	7B Main Agent			3B Main Agent		
	Single	Multi	**Avg**	Single	Multi	**Avg**
Standard	0.600	0.506	0.546	0.531	0.371	0.440
Oracle Retrieval	0.709	0.564	0.626	0.636	0.499	0.558
Rel. Gain	**+18.2%**	**+11.5%**	**+14.7%**	**+19.8%**	**+34.5%**	**+26.8%**

Table 16: Oracle retrieval gap (F1 score). Performance averaged over single-hop (PopQA, NQ, TQA) and multi-hop (HotpotQA, 2Wiki, Musique, Bamboogle) benchmarks.

Concretely, given an initial query q_0 with gold answer a^*, the agent generates a trajectory step by step following Eq. 8. At each step t, the agent produces a reasoning thought τ_t and sub-query q_t, and the dense retriever returns a candidate set $\mathcal{D}_N$. Let $\mathcal{D}^+ = \{d \in \mathcal{D}_N \mid a^* \subseteq d\}$ denote the documents in $\mathcal{D}_N$ that contain the gold answer. The observation presented to the agent is:

$$o_t = \begin{cases} \mathcal{D}^+ \,\|\, \text{top-}(K - |\mathcal{D}^+|) \text{ from } \mathcal{D}_N \setminus \mathcal{D}^+, & \text{if } \mathcal{D}^+ \neq \emptyset, \\ \text{top-}K \text{ from } \mathcal{D}_N, & \text{if } \mathcal{D}^+ = \emptyset, \end{cases} \tag{14}$$

where $\|$ denotes concatenation. When documents matching the gold answer exist in the candidate set, we promote them to the top positions and fill the remaining slots with the highest-ranked non-matching documents. For intermediate sub-queries where no retrieved document matches the gold answer, we use the default retriever ranking unchanged. The agent then observes o_t, continues reasoning, and repeats until it produces a final answer a. Algorithm 4 provides the full procedure.

Algorithm 4 Oracle Retrieval Rollout

Require: Initial query q_0; gold answer a^*; main agent $\pi_{\theta_{\text{main}}}$; candidate set size N; output size K

Ensure: Oracle trajectory y^{oracle}

$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  $\tau_t, q_t \leftarrow \pi_{\theta_{\text{main}}}(q_0, \tau_1, q_1, o_1, \ldots, \tau_{t-1}, q_{t-1}, o_{t-1})$ {agent reasons and generates sub-query}
  $\mathcal{D}_N \leftarrow \text{DenseRetrieve}(q_t, N)$
  $\mathcal{D}^+ \leftarrow \{d \in \mathcal{D}_N \mid a^* \subseteq d\}$ {gold-answer-matching documents}
  if $\mathcal{D}^+ \neq \emptyset$ **then**
    $o_t \leftarrow \mathcal{D}^+ \,\|\, \text{top-}(K - |\mathcal{D}^+|)$ from $\mathcal{D}_N \setminus \mathcal{D}^+$ {promote to top}
  else
    $o_t \leftarrow$ top-K from $\mathcal{D}_N$ {keep default ranking}
  end if
until agent outputs final answer a
return $y^{\text{oracle}} = (q_0, \tau_1, q_1, o_1, \ldots, \tau_m, q_m, o_m, \tau_{m+1}, a)$

Model	0	1	2	3	4	5	6	Avg
Oracle	0.0	49.3	38.2	9.8	2.3	0.3	0.1	1.66
CoSearch	**0.0**	**42.3**	**48.9**	**8.1**	**0.7**	**0.0**	**0.0**	**1.67**
Fixed Reranker	0.0	4.2	35.7	54.5	5.6	0.0	0.0	2.62
Retrieval Only	0.0	33.2	40.4	16.3	7.9	1.8	0.3	2.05

Table 17: Search turn distribution (% of questions) at validation step 100. **Avg** is the mean number of search turns per question.

B.7 Search Turn Distribution

Table 17 shows the full search turn distribution. CoSearch and Oracle concentrate 91% and 87% of questions at 1–2 turns respectively, while Fixed Ranker pushes 55% of questions to 3 turns, requiring significantly more retrieval rounds on average (2.62 vs. 1.67). This confirms that higher-quality ranking reduces the number of retrieval steps needed to resolve a question.

BIBLIOGRAPHY

Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. Umass at trec 2004: Novelty and hard. In *Proceedings of the Thirteenth Text REtrieval Conference*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology, 2004.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024.

Artem Babenko and Victor S. Lempitsky. Additive quantization for extreme vector compression. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 931–938, 2014.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*, 2024.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. Autoregressive search engines: generating substrings as document identifiers. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 17–23 Jul 2022.

Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. Ms marco: A human generated machine reading comprehension dataset. In *ArXiv*, volume abs/1611.09268, 2016.

Jiangui Chen, Ruqing Zhang, J. Guo, Yixing Fan, and Xueqi Cheng. Gere: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022a.

Jiangui Chen, Ruqing Zhang, J. Guo, Y. Liu, Yixing Fan, and Xueqi Cheng. Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022b.

Jiangui Chen, Ruqing Zhang, J. Guo, M. de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. Continual learning for generative retrieval over dynamic corpora. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023a.

Jiangui Chen, Ruqing Zhang, J. Guo, M. de Rijke, Y. Liu, Yixing Fan, and Xueqi Cheng. A unified generative retriever for knowledge-intensive language tasks via prompt learning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023b.

Xilun Chen, Kushal Lakhotia, Barlas Oguz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 250–262, Abu Dhabi, United Arab Emirates, December 2022c. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.19.

Yongjian Chen, Tao Guan, and Cheng Wang. Approximate nearest neighbor search by residual vector quantization. In *Sensors (Basel, Switzerland)*, volume 10, pages 11259 – 11273, 2010.

David R. Cheriton. From doc2query to doctttttquery. 2019.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the trec 2019 deep learning track. In *TREC*, 2019.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. Overview of the trec 2020 deep learning track. volume abs/2102.07662, 2021.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, and Aston Zhang et al. The llama 3 herd of models. 2024.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. Splade v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086*, 2021a.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292, 2021b. doi: 10.1145/3404835.3463098.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025.

Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2843–2853, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.203.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 36, pages 744–755, 2014.

Alex Graves. Sequence transduction with recurrent neural networks. In *ArXiv*, volume abs/1211.3711, 2012.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation. In *ArXiv*, volume abs/2010.02666, 2020.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and Allan Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 861–864, 2011.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, 2023.

Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, Suhang Wang, Jiawei Han, and Xianfeng Tang. Language models as semantic indexers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22244–22259. PMLR, 21–27 Jul 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. volume 7, pages 535–547. IEEE, 2019.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2020c.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. SIGIR '20, page 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075.

Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: robust multi-hop reasoning at scale via condensed retrieval. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. Natural questions: A benchmark for question answering research. In *Transactions of the Association for Computational Linguistics*, volume 7, pages 453–466, 2019.

John D. Lafferty and ChengXiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 120–127, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133316. doi: 10.1145/383952.383972.

Hyunji Lee, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vladimir Karpukhin, Yi Lu, and Minjoon Seo. Nonparametric decoding for generative retrieval. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, NeurIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.276.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. In *Advances in Neural Information Processing Systems*, 2025b.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl, 2025c.

Yongqing Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8716–8723, 2023a. doi: 10.1609/aaai.v38i8.28717.

Yongqing Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Multiview identifiers enhanced generative retrieval. In *Annual Meeting of the Association for Computational Linguistics*, 2023b.

Jimmy J. Lin and Xueguang Ma. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. In *ArXiv*, volume abs/2106.14807, 2021.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy J. Lin. Distilling dense representations for ranking using tightly-coupled teachers. In *ArXiv*, volume abs/2010.11386, 2020.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. Agentic-r: Learning to retrieve for agentic search. *ArXiv*, abs/2601.11888, 2026.

Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1895–1898, New York, NY, USA, 2009. Association for Computing Machinery. doi: 10.1145/1645953.1646259.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657951.

Sean MacAvaney, Franco Maria Nardini, R. Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. Training curricula for open domain answer re-ranking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

Yury Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 42, pages 824–836, 2016.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. In *IEEE Transactions on Neural Networks and Learning Systems*, volume 31, pages 3732–3740, 2017.

Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. DSI++: Updating transformer memory with new documents. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8198–8213, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.510.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.146.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. In *ArXiv*, volume abs/1901.04085, 2019.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63.

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. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Biswajit Paria, Chih-Kuan Yeh, Ian E.H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. Minimizing flops to learn efficient sparse representations. In *International Conference on Learning Representations*, 2020.

Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

M. F. Porter. *An algorithm for suffix stripping*, page 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1558604545.

Ronak Pradeep, Kai Hui, Jai Gupta, Adam Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Tran. How does generative retrieval scale to millions of passages? In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1305–1321, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.83.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy J. Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *ArXiv*, abs/2312.02724, 2023b.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *North American Chapter of the Association for Computational Linguistics*, 2020.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *J. Mach. Learn. Res.*, volume 21, pages 140:1–140:67, 2019.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.224.

Ruiyang Ren, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. TOME: A two-stage approach for model-based retrieval. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6102–6114, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.336.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. In *Found. Trends Inf. Retr.*, volume 3, pages 333–389, 2009.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In *Text Retrieval Conference*, 1994.

Alireza Salemi and Hamed Zamani. Towards a search engine for machines: Unified ranking for multiple retrieval-augmented large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 741–751, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657733.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.272.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Xiaolong Huang, Binxing Jiao, Linjun Yang, and Daxin Jiang. LexMAE: Lexicon-bottlenecked pretraining for large-scale retrieval. In *The Eleventh International Conference on Learning Representations*, 2023.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384, 2024a.

Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7339–7353, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.397.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. volume 28, pages 11–21, 1972.

Karen Sparck Jones. *A statistical interpretation of term specificity and its application in retrieval*, page 132–142. Taylor Graham Publishing, GBR, 1988. ISBN 0947568212.

Felix Stahlberg and Bill Byrne. On NMT search errors and model errors: Cat got your tongue? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1331.

Hongjin SU, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Liu Haisu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O Arik, Danqi Chen, and Tao Yu. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching, 2025a.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, and Ji-Rong Wen. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025b.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. Learning to tokenize for generative retrieval. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.923.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. Transformer memory as a differentiable search index. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. In *Journal of Machine Learning Research*, volume 9, pages 2579–2605, 2008.

Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian k-means. In *IEEE Transactions on Knowledge and Data Engineering*, volume 27, pages 180–192, 2014.

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

Shuai Wang, Shengyao Zhuang, and G. Zuccon. Bert-based dense retrievers require interpolation with bm25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, 2021.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. A neural corpus indexer for document retrieval. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. Novo: Learnable and interpretable document identifiers for model-based ir. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021a.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

An Yang et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 497–506, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3271800.

Hansi Zeng, Hamed Zamani, and Vishwa Vinay. Curriculum learning for dense retrieval distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.

Hansi Zeng, Surya Kallumadi, Zaid Alibadi, Rodrigo Nogueira, and Hamed Zamani. A personalized dense retrieval framework for unified information access. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023. ISBN 9781450394086. doi: 10.1145/3539618.3591626.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. Scalable and effective generative information retrieval. In *Proceedings of the Web Conference 2024*, WWW '24, 2024.

Hansi Zeng, Julian Killingback, and Hamed Zamani. Scaling sparse and dense retrieval in decoder-only llms. *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.

Hansi Zeng, Zoey Li, Yifan Gao, Chenwei Zhang, Xiaoman Pan, Tao Yang, Fengran Mo, Jiacheng Lin, Xian Li, and Jingbo Shang. Synplanresearch-r1: Encouraging tool exploration for deep research with synthetic plans, 2026.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. 22(2):179–214, April 2004. ISSN 1046-8188. doi: 10.1145/984321.984322.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, M. Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

Kai Zhang, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, Binxing Jiao, and Daxin Jiang. Led: Lexicon-enlightened dense retriever for large-scale retrieval. In *Proceedings of the ACM Web Conference 2023*, 2022.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. Generative retrieval via term set generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 458–468, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657797.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 414–431, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.22.

R. Zhou and Eric A. Hansen. Beam-stack search: Integrating backtracking with beam search. In *International Conference on Automated Planning and Scheduling*, 2005.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Yu Wu, Peitian Zhang, and Ji rong Wen. Ultron: An ultimate retriever on corpus with a model-based indexer. In *ArXiv*, volume abs/2208.09257, 2022.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022a.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, G. Zuccon, and Daxin Jiang. Bridging the gap between indexing and retrieval for differentiable search index with query generation. In *ArXiv*, volume abs/2206.10128, 2022b.