

Democratizing and Modernizing Information Access: From Open Rerankers to Scalable RAG Evaluation

by

Ronak Pradeep

A thesis
presented to the University of Waterloo
in fulfillment of the
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Ronak Pradeep 2025

Examining Committee Membership

The following serve on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Member:

Ophir Frieder
Professor
Department of Computer Science
Georgetown University

Supervisor:

Jimmy Lin
Professor
David R. Cheriton School of Computer Science
University of Waterloo

Internal Member:

Charles L.A. Clarke
Professor
David R. Cheriton School of Computer Science
University of Waterloo

Internal Member:

Victor Zhong
Assistant Professor
David R. Cheriton School of Computer Science
University of Waterloo

Internal-External Member: **Mark D. Smucker**

Professor
Department of Management Science and Engineering
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The thesis includes contributions from six papers. Ronak Pradeep is the first or co-first author of the papers described in Chapters 3–8 and led or co-led the ideation, methodology, implementation, experimentation, analysis, and writing.

List of publications and co-authors (* indicates equal contribution):

- [Chapter 3](#): The EXPANDO-MONO-DUO Design Pattern for Text Ranking with Pre-trained Sequence-to-Sequence Models. Ronak Pradeep, Rodrigo Nogueira, Jimmy Lin. *arXiv:2101.05667*, 2021.
- [Chapter 4](#): Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking. Ronak Pradeep*, Yuqi Liu*, Xinyu Zhang, Yilin Li, Andrew Yates, Jimmy Lin. *In Proceedings of the European Conference on Information Retrieval*, 2022.
- [Chapter 5](#): How Does Generative Retrieval Scale to Millions of Passages? Ronak Pradeep*, Kai Hui*, Jai Gupta, Adam Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, Vinh Tran*. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.
- [Chapter 6](#): RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! Ronak Pradeep, Sahel Sharifymoghaddam, Jimmy Lin. *arXiv:2312.02724*, 2023.
- [Chapter 7](#): RAGNARÖK: A Reusable RAG Framework and Baselines for TREC 2024 Retrieval-Augmented Generation Track. Ronak Pradeep*, Nandan Thakur*, Sahel Sharifymoghaddam, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, Jimmy Lin. *In Proceedings of the European Conference on Information Retrieval*, 2025.
- [Chapter 8](#): The Great Nugget Recall: Automating Fact Extraction and RAG Evaluation with Large Language Models. Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, Jimmy Lin. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.

Abstract

Modern information access increasingly relies on complex pipelines involving large language models (LLMs), fundamentally changing how users interact with information, from sophisticated multi-stage retrieval pipelines to end-to-end retrieval-augmented generation (RAG) systems. While these advancements enhance user experience, they also introduce significant challenges. The research community’s growing reliance on proprietary, black-box models for key tasks like document reranking creates barriers to innovation and reproducibility (the Component Challenge). Furthermore, progress is hampered by the lack of a shared, standardized ecosystem for executing and measuring information access systems (the Benchmarking Challenge). Finally, the generative nature of RAG systems makes them fundamentally harder to evaluate than traditional systems that return document lists; new methodologies are required to assess factual accuracy and completeness in a reliable, scalable manner (the Evaluation Challenge). We argue that progress depends on the synergistic development of open, high-effectiveness system components and the reliable, scalable evaluation frameworks necessary to assess them.

This thesis addresses these challenges through a narrative arc that begins with pushing existing paradigms to their limits. But, given that frontier models dominate today’s landscape, we find a pressing need for new open-source solutions. We begin by analyzing the dominant supervised ranking paradigm, developing multi-stage pipelines that demonstrated high effectiveness but also highlighted inherent complexity and cost. Subsequently, we conducted a systematic exploration of model backbones, loss functions, and negative mining strategies to squeeze effectiveness gains from supervised pointwise cross-encoders. Next, we continue with a large-scale empirical study on the newly evolving generative retrieval paradigm, which revealed its scalability limitations on large, real-world collections. This portion culminates in the final contribution to the Component Challenge: RankZephyr, an open-source 7B-parameter listwise reranker. By leveraging a carefully designed instruction distillation curriculum, RankZephyr matches and often surpasses the effectiveness of much larger proprietary models like GPT₄. It provides the community with a powerful, transparent, and accessible zero-shot reranking module, breaking the dependence on black-box systems for this critical task. All methods described have broad community adoption, and our models and evaluation frameworks continue to support ongoing research efforts across open-source IR and RAG development.

With powerful open components in hand, the focus shifts to benchmarking. To address the Benchmarking Challenge, this work introduces Ragnarök, a reusable, end-to-end RAG framework designed to standardize how retrieval-augmented generation systems are constructed and assessed. Serving as the backbone for the TREC 2024 Retrieval-Augmented

Generation Track, Ragnarök provides the research community with a shared experimental platform, critical data resources, and reproducible and effective baselines. By encapsulating the full RAG pipeline — from retrieval and grounding to generation and scoring — within a single, transparent framework, TREC 2024 Retrieval-Augmented Generation Track and Ragnarök enable reproducible experimentation at scale. This not only ensures fair comparisons across diverse approaches but also establishes a foundation for cumulative progress in open-domain information access research, where previously ad hoc and non-replicable setups have often impeded reliable evaluation.

Building on this infrastructure, the thesis then directly tackles the Evaluation Challenge by introducing the AutoNuggetizer framework. This framework refactors the classic and well-studied nugget-based evaluation methodology for the modern era of LLMs. By automating the evaluation of the recall of the information nugget in RAG responses and validating the approach at scale in TREC 2024 Retrieval-Augmented Generation Track, this work provides a reliable and scalable methodology to measure the quality of generative information access systems.

In summary, this thesis contributes to the field of information access by exploring the limits of existing retrieval and ranking paradigms, developing powerful open-source components for modern information access systems, and creating the frameworks and methodologies required to benchmark and evaluate them. The contributions include a comprehensive analysis of supervised ranking and generative retrieval paradigms, an open-source state-of-the-art listwise reranker (RankZephyr), a standardized framework for RAG benchmarking (Ragnarök), and a scalable methodology for evaluating generative systems (AutoNuggetizer). Together, this thesis addresses the three core challenges identified at the outset, providing the community with both the tools to build effective systems and the methodologies to assess them rigorously. The widespread adoption of these artifacts by researchers and practitioners already underscores their tangible impact and utility in driving the field forward.

In the future, on the reranking front, we would like to build faster, more efficient rerankers that can *reason* over the texts and generalize to several domains. On the benchmarking front, we will expand tasks to capture “deep research” information needs that demand multi-hop reasoning and query decomposition. On the evaluation front, we hope to extend the AutoNuggetizer methodology to several tasks that go beyond web retrieval, into other domains like biomedical texts and conversational question answering.

Acknowledgements

I would first like to thank my advisor, Jimmy Lin, who has supported me since the day I first met him in the DSG lab as a wee little undergrad. He pushed me to think about long-term impact, not to worry about conference acceptances, to build things that matter, and to ship ideas into the world. Jimmy gave me the freedom to explore ambitious directions while somehow always steering me just enough not to fall into the trap of taking on more projects than I can handle with his perfectly timed “Duuuude, you gotta write up your thesis!” I could not have asked for a better mentor to learn from, debate with, and ultimately cross the finish line alongside.

I want to thank my committee members — Charles, Mark, Ophir, and Victor — for all their time spent reading my thesis and attending my defence. I hope I didn’t make it too painful.

I am grateful for the support of the Waterloo Apple PhD Fellowship in Data Science and Machine Learning, which afforded me the rare privilege of living like a non-broke PhD student — including the freedom to travel, live life, and eat *somewhat* freely.

I was fortunate to have wonderful mentors through my internships. Jeff, Pascal, and Rodrigo at Waterloo; Jie at MILA; Don, Kai, and Vinh at Google; Yunyao, Ihab, Mostafa, and Saloni at Apple; Gilad and Pankaj at Yupp — each of you shaped how I think about doing impactful research. All of you taught me how to judge and pick the correct research problems, and that great teams make even the *occasional* 5 AM “WHAT IS GOING ON” sessions feel fun. I have had the pleasure of collaborating with some truly exceptional researchers through my PhD, as well as leading the TREC 2024 RAG Track. Special shoutouts to Minghan, Nandan, Shivani, Crystina, Jack, Matt, Yuqi, Carlos, Mofe, Ehsan, Tunde, Eugene, Ralph, Nick, Daniel, Ian, and Hoa.

At Apple, I got to find some wonderful friends in Daniel, Revanth, and Simone. From weekly work-related rants (research speak) to grabbing the daily salad to weekend ice cream and breakfast club runs, all of you made my time in Seattle so much fun! To my friends, Anurag, Benson, Fadhil, Gaurav, Kanav, Kerem, Sairaj, Tejasvi, and Vijay, we have had so much fun over the years at the DSG with long, long conversations about all sorts of things, and I will forever cherish the time we spent together. Outside of all this, I would especially like to thank Pratyush for being a wonderful friend to me from my early Waterloo undergrad days and being someone I can always lean on!

One of the most unexpected pleasures of my PhD journey was working with several great undergrad researchers, many of whom work with their own undergrad researchers

now. Faraz, Jasper, Manveer, Steven, and Xueguang, you all are incredible people and researchers, and I am always excited to see what you build next!

I am grateful for *my Achamma* (Malayalam for dad’s mother), who always had my back for the first 26 years of my life. You were a strong and incredible force, and through life I hope to have some of the strength you did. I also want to thank *my Ammoma* (Malayalam for mother’s mother), for always being such a fun, kind, and loving person to all the versions of me you have known, including the ones that kept playing dumb pranks on you. I remember being a five-year-old, coming down to your home, excited to solve all the different puzzles *with you*. Even as life has changed, I will continue to do that while I can.

Roshan, I could never have asked for a better brother. From all the times we learned, played, and annoyed the living hell out of one another, I enjoyed it all thoroughly, right from the moment I first held you in my arms as a newborn baby, to every argument and adventure since.

I would never be where I am today without the love and unwavering support of my parents. My papa, Pradeep, you worked hard and lived frugally to make sure I could have the best life. I hope that seeing some “Pradeep et al.” brings you some joy (in choosing to add a patronym). My mummy, Beena, you have always been there for me, ever since I was a little kid, and taught me to enjoy reading, music, and math, and to always believe in myself. More than anything, you encouraged me to always stay curious. You trusted that I would always find my way in this world, and while I still haven’t yet, that hope gives me confidence that someday I will. I can only aspire to pay forward a fraction of what you both have given me, and to live a life that reflects the values you raised me with.

Most importantly, I thank Antea for all her love and support through this long journey. I could never have asked for a better partner — from the start of my PhD, we would spend hours talking about everything under the sun, to our weekly Pink Floyd’s “The Wall”, to you making me your wonderful yogurt mixes every day, to all the times you burst into singing Ella Fitzgerald, Tool, and everything in between. Even now, you sit with me over Zoom, keeping me company with your beautiful smile, while I struggle to write this paragraph, knowing that no words will ever feel enough. I guess, as the fox in *The Little Prince* says, “It is only with the heart that one can see rightly; what is essential is invisible to the eye.” You believed in me, gave me comfort, and powered me forward when I most needed it. For all the journeys we have shared, and all that will come, I look forward to the life we will build together, hand in hand.

Dedication

To Antea, Mummy, and Papa.

Table of Contents

Examining Committee	ii
Author’s Declaration	iii
Abstract	v
Acknowledgements	vii
Dedication	ix
List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Challenges in Modern Information Access	2
1.2 A Path Through the Challenges	3
1.3 Contributions	5
2 Background and Related Work	7
2.1 Foundations of Multi-Stage Information Retrieval	7
2.1.1 Traditional First-Stage Retrieval	8
2.1.2 The Reranking Cascade	9

2.2	Large Language Models	10
2.3	Neural Models for Text Ranking: A Deep Dive	12
2.3.1	Bi-Encoders for First-Stage Retrieval	12
2.3.2	Cross-Encoders for Reranking	13
2.4	Generative Information Access	15
2.4.1	Generative Retrieval	15
2.4.2	Retrieval-Augmented Generation (RAG)	15
2.4.3	Synthetic Data Generation	16
2.5	Zero-Shot Reranking with LLMs	16
2.6	Benchmarks and Evaluation Methodologies	17
2.6.1	Corpora and Topics	18
2.6.2	The Nugget Evaluation Methodology	19
2.6.3	Automated Evaluation with LLMs	20
3	Expando-Mono-Duo: A Design Pattern for Supervised Reranking	22
3.1	Expando-Mono-Duo with T5	23
3.1.1	H_{-1} : Document Expansion with T5	25
3.1.2	H_0 : Keyword Retrieval	26
3.1.3	H_1 : Pointwise Reranking with monoT5	26
3.1.4	H_2 : Pairwise Reranking with duoT5	27
3.2	Experimental Settings	28
3.2.1	MS MARCO Passage Ranking	29
3.2.2	MS MARCO Document Ranking	31
3.2.3	TREC-COVID	35
3.3	Results	39
3.3.1	MS MARCO Passage Ranking	39
3.3.2	TREC 2020 Deep Learning Track Passage Ranking	42
3.3.3	MS MARCO Document Ranking	44

3.3.4	TREC 2020 Deep Learning Track Document Ranking	46
3.3.5	TREC-COVID	47
3.4	Concluding Thoughts	50
4	Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking	52
4.1	Methodology	53
4.1.1	Cross-Entropy and Hinge Loss	53
4.1.2	Localized Contrastive Estimation	54
4.1.3	Experimental Setup	55
4.1.4	Data	55
4.1.5	Training and Inference	56
4.2	Results and Discussion	57
4.2.1	Loss Functions	58
4.2.2	Hard Negatives and Retrieval Alignment	59
4.2.3	LCE Group Size	60
4.3	Concluding Thoughts	62
5	On the Scalability of Generative Retrieval	64
5.1	Methods	65
5.1.1	Background	65
5.1.2	Inputs and Targets	65
5.1.3	Model Variants	68
5.2	Experimental Setting	69
5.2.1	Corpus and Training Data	69
5.2.2	Synthetic Query Generation	70
5.2.3	Evaluation Dataset and Metrics	70
5.2.4	Model Variants	71

5.3	Experimental Results	71
5.3.1	Ablations over Small Corpora	71
5.3.2	Scaling Corpus Size	73
5.3.3	Scaling Model Size	75
5.4	Concluding Thoughts	77
6	RankZephyr: An Open-Source Listwise Reranker	78
6.1	Methodology	79
6.1.1	Prompt Design	79
6.1.2	Training Stage 1: Distilling from GPT _{3.5}	79
6.1.3	Training Stage 2: Distilling from GPT ₄	80
6.1.4	Inference	81
6.2	Experimental Setup	81
6.2.1	Datasets	81
6.2.2	Baseline Systems	82
6.2.3	Evaluation Metrics and Procedures	82
6.3	Results and Analysis	82
6.3.1	Main Effectiveness Results	82
6.3.2	Training Strategy Ablation	85
6.3.3	Robustness Analysis	86
6.3.4	First-Stage Retrieval Impact	87
6.3.5	Generalization to Novel Data	88
6.4	Concluding Thoughts	89
7	Ragnarök: A Framework for Benchmarking Generative Information Access	91
7.1	Our Framework	92
7.1.1	RAG Input/Output Definitions	93

7.2	Document Collection	94
7.3	Topic Collection	95
7.4	TREC RAG Baselines	97
7.4.1	Retrieval	97
7.4.2	Augmented Generation	99
7.4.3	RAG-Bench Evaluation	101
7.5	Ragnarök System Arena	101
7.6	Concluding Thoughts	103
8	AutoNuggetizer: A Framework for Evaluating Generative Information Access	105
8.1	The TREC 2024 RAG Track	106
8.2	The AutoNuggetizer Framework	107
8.2.1	Nugget Creation	107
8.2.2	Nugget Assignment	111
8.2.3	Evaluation Strategies	113
8.2.4	Scoring	113
8.3	Results	114
8.3.1	Descriptive Statistics	114
8.3.2	Fully Automatic Nugget Evaluation	115
8.3.3	Automatic Nugget Assignment	118
8.3.4	Human vs. LLM Assignment	120
8.4	Concluding Thoughts	121
9	Conclusions and Future Work	123
	References	126

List of Figures

1.1	Illustration of a modern information access pipeline.	2
3.1	The Expando-Mono-Duo multi-stage ranking architecture, from document expansion to pairwise reranking.	24
3.2	Distribution of passage lengths in the MS MARCO passage corpus.	29
3.3	Distribution of document lengths in the MS MARCO document corpus.	32
3.4	Effectiveness of duoT5-3B on the MS MARCO development set versus the number of pairwise inferences performed.	42
6.1	The reranking prompt template and sample generation for RankZephyr.	80
7.1	Schematic diagram of the Ragnarök RAG framework, showing the retrieval and augmented-generation steps.	92
7.2	The ChatQA prompt template used for RAG generation with in-text citations in the Ragnarök framework.	100
7.3	WebUI for the Ragnarök System Arena, showing a side-by-side comparison of two RAG pipeline outputs	102
8.1	The AutoNuggetizer prompt for the iterative nuggetization at turn i	108
8.2	The AutoNuggetizer prompt for determining the importance of nuggets.	110
8.3	The AutoNuggetizer prompt for nugget assignment.	111
8.4	Scatter plots comparing the scores from the fully-automatic AutoNuggetizer vs. the fully manual or semi-manual evaluation conditions	116

8.5	Scatter plots comparing automatic vs. manual nugget assignment to isolate its effect on evaluation scores.	119
8.6	Confusion matrices showing agreement between automatic and manual nugget assignment strategies.	120

List of Tables

3.1	Results of Expando-Mono-Duo Variants on the MS MARCO passage ranking task.	40
3.2	Results of Expando-Mono-Duo Variants on the TREC 2020 Deep Learning Track passage ranking task.	43
3.3	Results of Expando-Mono-Duo Variants on the MS MARCO document ranking task.	45
3.4	Results of Expando-Mono-Duo Variants on the TREC 2020 Deep Learning Track document ranking task.	46
3.5	Results of selected Expando-Mono-Duo Variants on the TREC-COVID task.	48
4.1	Results of monoELECTRA with different loss functions, and hard-negative and first-stage retrievers on the MS MARCO passage ranking task.	57
4.2	Results of monoELECTRA across different hard negatives and first-stage retriever combinations.	59
4.3	Results of the top monoELECTRA variant on the official MS MARCO passage leaderboard	61
4.4	Results of monoELECTRA across different hard negative sources and number of negatives on the MS MARCO passage ranking task.	61
5.1	Coverage statistics for the generative retrieval benchmark datasets and their training query sets.	69
5.2	Results of the generative retrieval variants on the small-scale NQ and TriviaQA datasets.	72
5.3	Results of the generative retrieval variants on different scales of the MS MARCO passage ranking task.	74

5.4	Effect of scaling model size for generative retrieval on the MS MARCO passage ranking task.	75
5.5	Effect of scaling model size for generative retrieval on the TREC 2020 Deep Learning Track passage ranking task.	76
6.1	Results of RankZephyr compared to state-of-the-art proprietary models on the primary retrieval benchmarks.	83
6.2	Results comparing different RankZephyr training ablations on the passage ranking task for TREC 2019 and 2020 Deep Learning Tracks.	85
6.3	Results comparing RankZephyr vs. RankGPT _{3.5} reranking different first-stage models (shuffled or not) on the passage ranking task for TREC 2019 and 2020 Deep Learning Tracks.	86
6.4	Results comparing RankZephyr reranking different first-stage models on the passage ranking task for TREC 2019 and 2020 Deep Learning Tracks. . . .	88
6.5	Results of RankZephyr on the NovelEval-2306 task.	89
7.1	Document and segment counts for MS MARCO V2 vs. the deduplicated V2.1	94
7.2	Topic distribution for the TREC-RAGgy 2024 and TREC-Researchy 2024 collections.	95
7.3	Results of baselines on the document ranking task of the TREC-RAGgy 2024.	99
8.1	Sample answer from the TREC 2024 RAG Track for topic 2024-79081. . .	106
8.2	Comparison of five sample nuggets across different nugget creation strategies.	109
8.3	Comparison of nugget assignment across different nugget creation approaches.	112
8.4	Descriptive statistics for nugget creation and assignment under different conditions.	114

Chapter 1

Introduction

The field of information access has transformed with the advent of large language models (LLMs), altering how users interact with information systems and, as a result, how these systems are designed and evaluated. Modern information retrieval (IR) systems are increasingly architected as multi-stage pipelines. Systems leverage the generative capabilities of these LLMs to provide more nuanced and comprehensive responses to user queries. At the forefront of this is the retrieval-augmented generation (RAG) paradigm [110], which has reshaped the interaction between users and information systems by bridging traditional IR and modern generative AI.

The RAG paradigm represents a departure from traditional search systems that return ranked lists of documents. In RAG, LLMs synthesize information from multiple relevant sources (documents) to generate a direct answer to a user’s query. This approach combines the strengths of IR systems — quick access to vast, up-to-date knowledge — with the natural language generation capabilities of LLMs.

Modern information access systems, therefore, operate not as a single retrieval step but as layered pipelines that integrate ranking and generation. As illustrated in [Figure 1.1](#), a user query first passes through a retriever to fetch a candidate set of documents, which may then be reranked by increasingly powerful models before finally being fed into an LLM that synthesizes an answer — often with citations. This response is ultimately what the user consumes, directly delivering the final information need in a synthesized and actionable form.

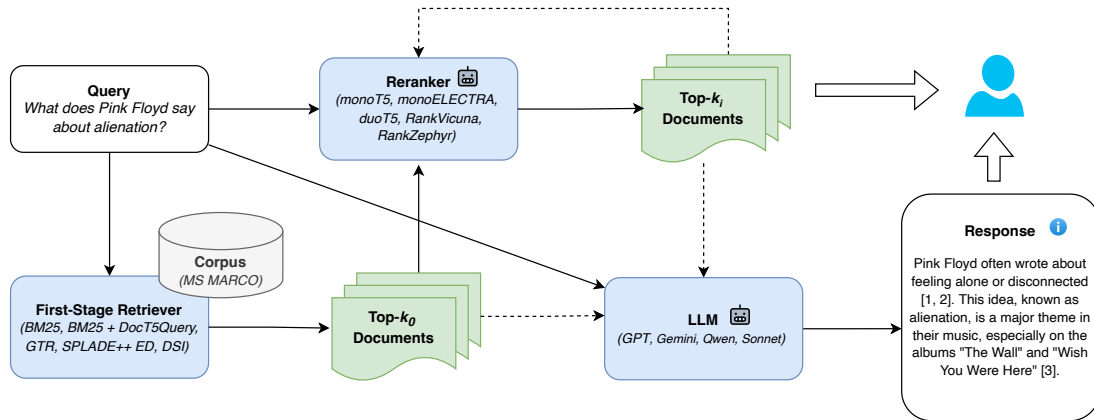


Figure 1.1: Illustration of a modern information access pipeline based on LLMs. The pipeline includes a first-stage retrieval followed by several reranking stages. In standard information retrieval, these results are directly presented to the user. In the RAG paradigm, the retrieved results (with or without reranking) are passed to an LLM to generate a response with citations. This response is what the user ultimately sees. Dotted lines indicate optional steps; the LLM may also answer from parametric memory without using the retrieved context or by skipping just the reranking stage.

1.1 Challenges in Modern Information Access

While such advancements enhance user experience, they also introduce new challenges that must be addressed to advance on solid ground. Developing complex information access pipelines reveals three core intertwined needs that represent bottlenecks in the current research landscape:

1. **The Component Challenge:** There is a pressing need for open-source modules that ensure reproducibility and effectiveness to serve as the building blocks for these information access pipelines. The research community’s growing reliance on proprietary, black-box models for key tasks like document reranking creates barriers to innovation and limits reproducibility. The result is a concentration of power in the hands of a few large tech companies.
2. **The Benchmarking Challenge:** Progress requires shared, standardized ecosystems for running and evaluating long-form RAG systems transparently. The lack

of a unified open-source framework that integrates state-of-the-art components and provides resources for rigorous RAG evaluation makes comparing different systems an ad-hoc and unreliable process, slowing the pace of innovation.

3. **The Evaluation Challenge:** The generative, open-ended nature of RAG systems makes them fundamentally more difficult to evaluate than traditional retrieval systems. Metrics focused on document relevance or factoid question answering are insufficient for assessing the quality of long-form, system-generated answers that may contain a mixture of correct and incorrect information, and for which there is often no single correct answer. New methodologies are required to measure critical aspects like factual accuracy, completeness, and coherence in a manner that is both reliable and scalable.

These challenges involve research equity, reproducibility, and democratization of information access, striking at the heart of how we measure progress in generative information systems. The central thesis underlying this report is that progress in information access depends on the synergistic development of open system components that are effective and the reliable evaluation frameworks necessary to assess them at scale.

1.2 A Path Through the Challenges

This thesis addresses the above challenges by presenting a body of work that pushes open-source rerankers to their limits over the years and creates the necessary infrastructure and methodologies to evaluate long-form RAG systems.

[Chapter 2](#) lays the groundwork by providing a comprehensive background on the evolution of information access, from traditional multi-stage architectures to the latest generative and LLM-driven approaches, as well as the evaluation methodologies that have accompanied them.

The subsequent chapters tackle the Component Challenge. Here, we begin by exploring the limits of the dominant supervised ranking paradigm. [Chapter 3](#) introduces the Expando-Mono-Duo design pattern, a simple yet highly effective pipeline for retrieval and reranking using the T5 model [186], demonstrating state-of-the-art effectiveness, circa 2020. Here, we also highlight the inherent costs and trade-offs of such systems, encouraging a more holistic evaluation that extends beyond effectiveness measures. This work represents the birth of a highly influential pipeline that has remained a popular baseline for five years since its inception. Continuing this thread, [Chapter 4](#) shows that even well-established

BERT-based [50] reranker models [154] can be significantly improved through a systematic combination of better backbones, loss functions, and negative mining strategies, further “squeezing” effectiveness from supervised techniques.

We also explore the challenges for a radically different approach: generative retrieval, introduced by Tay et al. [211]. In this paradigm, a Transformer [223] model serves as a monolithic differentiable index that maps string queries directly to relevant DocIDs; in other words, the model answers queries by returning ranked lists of identifiers directly using only its parameters, dramatically simplifying the whole retrieval process. Chapter 5 presents the first large-scale empirical study on its scalability. The findings reveal that while promising on small corpora, the paradigm struggles to compete with established baselines on large, real-world collections, suggesting that the strengths of large sequence-to-sequence models may be better applied elsewhere in the retrieval pipeline.

The final contribution to the Component Challenge in this thesis comes after the large wave of *frontier* and *proprietary* LLMs, around 2022-2023, took the reranking paradigm by storm. In a zero-shot fashion, models like GPT₄ could be prompted to reorder a list of passages (a paradigm we dub “listwise”) and achieve state-of-the-art effectiveness in several IR benchmarks [207, 135, 106]. But these proprietary models come with several reproducibility challenges, a lack of transparency, and costs of over half a dollar per query [207]. Chapter 6 introduces RankZephyr, an *open-source* 7B-parameter listwise reranker that matches and often surpasses the effectiveness of much larger proprietary models like GPT₄. By leveraging a carefully designed instruction distillation curriculum, RankZephyr provides the community with a powerful, transparent, and accessible zero-shot reranking module, breaking the dependence on black-box systems for this crucial task.

With a powerful open component in hand, the focus of the thesis shifts to evaluation. Chapter 7 addresses the Benchmarking Challenge by introducing Ragnarök, a reusable, end-to-end RAG framework designed to standardize evaluation. Serving as the backbone for the TREC 2024 Retrieval-Augmented Generation Track, Ragnarök provides the community with a common platform, critical data resources, and strong, reproducible baselines, creating the essential infrastructure for rigorous RAG assessment.

Finally, building on this infrastructure, Chapter 8 tackles the Evaluation Challenge. It introduces the AutoNuggetizer framework, which refactors the classic, well-studied nugget-based evaluation methodology for the modern era of LLMs. By automating the assessment of information nugget recall in RAG answers and validating the approach at scale in the TREC 2024 RAG Track, this work provides a reliable and scalable methodology for measuring the quality of generative information access systems.

1.3 Contributions

In summary, this thesis contributes to the field by exploring the limits of existing retrieval and ranking paradigms, developing powerful open-source components for modern information access systems, and creating the frameworks and methodologies required to benchmark and evaluate them. The specific contributions are as follows:

1. **Pushing the Boundaries of Supervised Reranking (Chapter 3 and Chapter 4):** We developed and analyzed simple yet effective state-of-the-art supervised reranking systems. These systems include the Expando-Mono-Duo T5-based design pattern, a multi-stage architecture, and a systematic study that significantly improved cross-encoder effectiveness by optimizing model backbones, loss functions, and hard-negative mining strategies. This work established the then state-of-the-art of the supervised paradigm pre-LLM era.
2. **Large-Scale Analysis of Generative Retrieval (Chapter 5):** We conducted the first large-scale empirical study on the scalability of generative retrieval, evaluating its effectiveness on corpora up to 8.8 million passages. The findings identified the practical limits of this paradigm, showing that its effectiveness does not scale gracefully and motivating the search for more viable applications of large sequence-to-sequence models in IR.
3. **An Open-Source State-of-the-Art Listwise Reranker (Chapter 6):** To address the Component Challenge in the era of LLMs, we developed RankZephyr, a 7B-parameter, open-source listwise reranker. Through a novel two-stage distillation process, RankZephyr achieves effectiveness competitive with, and often superior to, much larger proprietary models like GPT₄, providing the research community with a transparent and robust building block for modern retrieval systems.
4. **A Standardized Framework for RAG Benchmarking (Chapter 7):** To address the Benchmarking Challenge, we created the Ragnarök framework, an open-source, end-to-end RAG ecosystem. Ragnarök provides the software, curated data collections (MS MARCO V2.1), and diverse topic sets that served as the official platform for the TREC 2024 Retrieval-Augmented Generation Track, enabling standardized and reproducible community-wide evaluation.
5. **A Scalable Methodology for Evaluating Generative Systems (Chapter 8):** To address the Evaluation Challenge, we developed the AutoNuggetizer framework,

which automates the classic nugget-based evaluation methodology using LLMs. Validated at scale against human judgments in the TREC 2024 Retrieval-Augmented Generation Track, AutoNuggetizer provides a reliable, scalable, and flexible approach to measuring information recall in RAG systems, completing the cycle of building and evaluating modern information access systems.

Collectively, these contributions provide a synergistic set of open components and evaluation frameworks that form a solid foundation for future research and the development of more capable and transparent information access systems.

Chapter 2

Background and Related Work

This chapter provides a comprehensive background on information access, setting the stage for the contributions presented in this thesis. First, we revisit *ad hoc* retrieval and the multi-stage ranking architecture, the long-standing backbone of production search systems. We then trace the paradigm shift brought about by neural networks, beginning with an overview of large language models (LLMs) and the Transformer architecture that underlies modern systems. This background leads to a detailed examination of neural ranking models, including bi-encoders for retrieval and cross-encoders for reranking, and to an exploration of the nuances of pointwise, pairwise, and listwise training objectives. Following this, we dive into the latest generative approaches to information access, covering generative retrieval, RAG, and LLMs for synthetic data generation. We highlight a major inflection point: the rise of zero-shot LLM reranking, where ranking is performed through prompting rather than task-specific training. Next, we cover several question answering and RAG benchmarks.

2.1 Foundations of Multi-Stage Information Retrieval

We assume the standard definition of *ad hoc* retrieval, where given a corpus of texts $\mathcal{C}$, the goal of a ranking model is to return a top- k ranked list of texts from the corpus in response to an information need q that maximizes some metric of ranking quality such as nDCG or MRR. The basic idea behind multi-stage ranking architectures is to break *ad hoc* retrieval down into a series of pipeline stages. Following an initial retrieval stage (also called candidate generation or first-stage retrieval), each subsequent stage reranks the list of candidates passed along from the previous stage until the final top- k results are

generated for consumption. Recognizing that the unit of retrieval might differ depending on the task, per standard parlance in IR, we use *document* to refer to the text being retrieved in a generic sense, even though it may be a passage or some hybrid construction.

Modern search systems are almost universally designed as multi-stage ranking pipelines [140, 26, 230] to achieve an optimal balance between effectiveness and computational efficiency. This architectural choice reflects the fundamental trade-off in IR between the need to process large document collections quickly and the desire to apply sophisticated ranking systems. In a typical cascade architecture, an initial retrieval stage uses a fast but relatively less precise model to fetch a large set of candidate documents from the corpus. This first stage might employ traditional BM25 [191], more recent sparse [58, 59], or dense [95, 89] retrievers to identify potentially relevant documents from collections containing millions or billions of documents. The goal at this stage is high recall — ensuring that relevant documents are included in the candidate set — rather than precise ranking.

Subsequent stages in the pipeline then apply increasingly powerful but computationally more expensive models to rerank progressively smaller subsets of these candidates. These models capture rich query–document interactions and subtle semantics, leveraging large-scale pre-training and fine-tuning. The final reranking stage determines the ordering of results that users or generation models see, as we have illustrated in Figure 1.1, making it essential that these models achieve the highest possible effectiveness.

2.1.1 Traditional First-Stage Retrieval

The initial retrieval stage has traditionally been dominated by bag-of-words (BoW) models that rely on lexical overlap. BoW models estimate the relevance of a document to a query primarily based on word overlap. It disregards word order (and thus most of syntax) and represents text documents as vectors of word counts or term frequencies. Its simplicity makes it reasonably effective for various tasks, including document classification, information retrieval, and topic modeling.

One of the most effective BoW models is BM25 [191], which refines the traditional term frequency-inverse document frequency (TF-IDF) approach by incorporating document-specific adjustments. BM25 modifies term frequency to account for diminishing returns of term repetition and includes a document length normalization parameter to handle the bias towards longer documents. All of this is governed by the following formula:

$$\text{score}(q, d) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgl}})}, \quad (2.1)$$

where q_i is the i -th term in the query, $f(q_i, d)$ is the term frequency of q_i in the document d , $|d|$ represents the length of the document, and avgdl is the average document length across the text collection. The parameters k_1 and b are adjustable free parameters.

Unless stated otherwise, we leverage the BM25 implementation found in Anserini [245], a toolkit for reproducible information retrieval research built on Lucene, or Pyserini [119], a Python wrapper of Anserini. Both of these use default values of $k_1 = 0.9$ and $b = 0.4$.

In application, the document BoW representations are usually organized in an inverted index for efficient search. An inverted index consists of two main components: a dictionary and posting lists. The dictionary contains all the unique terms that appear in the dataset, while the postings lists record each term’s occurrences across the documents. Each entry in a postings list typically includes the document identifier and can also store additional information like term frequency, position, or context. Methods such as tiered indexing, posting compression, and partitioning are applied to optimize the efficiency of inverted indexes further, but they are not relevant to the scope of this thesis.

2.1.2 The Reranking Cascade

Multi-stage ranking architectures have received much interest in academia [140, 230, 9, 29, 138] as well as industry. Documented production deployments of this architecture include the Bing web search engine [168] as well as Alibaba’s e-commerce search engine [127]. These represent instances of a mature and well-studied design, which originally evolved to strike a balance between model complexity and search latency by controlling the size of the candidate set at each stage. Increasingly richer (and typically slower) models can be made practical by considering successively smaller candidate sets. For certain easier queries, stages of the pipeline can be skipped entirely, known as “early exits” [26]. Viewed in this manner, multi-stage ranking captures the same intuition as progressive refinement in classifier cascades [224], which has a long history. Given this setup, several researchers have proposed techniques based, for example, on boosting for composing these stages in an end-to-end manner [230, 243]. While these techniques were previously explored mostly in the context of feature-based learning to rank [128, 112], many of the same ideas remain applicable in a transformer-based setting [156, 204, 239]. We explore more details of what exactly constitutes the models in each of these stages in [Section 2.3](#).

2.2 Large Language Models

At their core, language models are neural networks, often with millions of parameters (models in the order of several billion parameters are dubbed “LLMs”, albeit these distinctions have varied with improvements in serving systems). Vaswani et al. [223] explore replacing previous recurrent architectures [195, 81, 35] that were plagued with issues like vanishing gradients [14], with the self-attention mechanism that enabled efficient parallel processing and the modeling of long-range dependencies in text. They called this the Transformer, with an encoder that converts the input sequence into contextual representations and a decoder that attends to those representations to generate an output sequence. Pre-trained language models based on the Transformer have demonstrated impressive capabilities when fine-tuned for various downstream tasks. Depending on their architecture, pre-trained Transformers can be classified into three categories: encoder-only models, encoder-decoder models, and decoder-only models.

Encoder-only Models. These models, exemplified by BERT [50], utilize only the encoder stack of the Transformer. They were designed for language understanding tasks by creating rich, contextualized representations of the input text. Encoder-only models process the entire input sequence at once, allowing for bidirectional attention where each token can attend to all other tokens in the sequence. They are typically pre-trained using a masked language modeling (MLM) objective, where the model learns to predict randomly masked tokens based on their surrounding context. The result is a model effective for tasks such as text classification, sentiment analysis, and named entity recognition, as evident from the results of BERT [50] and those that quickly followed [250, 129, 105]. Four years later, encoder-only models still remain an active domain of interest to the retrieval community, as the vector encodings directly capture contextual information. However, given that the other categories of models we will discuss express multiple tasks at once without the need for task-specific classification heads like encoder-only models do, they have gradually lost attention from the wider NLP community. Efforts like ModernBERT [236] and NeoBERT [20] hope to bring much-needed attention and updates to this useful paradigm of language modeling, especially to the information access community.

Encoder-Decoder Models. The encoder-decoder architecture uses both Transformer stacks: an encoder that converts the input sequence into contextual representations and a decoder that attends to those representations to generate an output sequence. This design is well-suited for tasks that require mapping an input sequence to a different out-

put sequence, such as machine translation, text summarization, and question answering. Foundational models in this category include T5 [186] and BART [109].

The training paradigm for these models typically involves a self-supervised pre-training phase on massive unlabeled text corpora, which allows the model to build a general representation of language [186]. Following this, the model can either be fine-tuned on smaller, task-specific labeled datasets or, for well-trained cases, used directly for new tasks through in-context learning and prompt engineering, without requiring any updates to the model’s parameters [8, 237].

The encoder–decoder separation decouples parameters for different functionalities and thus enjoys higher freedom in handling contextual representation and challenging tasks at smaller scales [210, 232]. It also offers higher flexibility in changing the encoder and decoder size to control effectiveness–efficiency trade-off [96, 254], an increasingly important aspect for LLM deployment. Despite these benefits, however, such models have received little attention nowadays, potentially due to the scale and ease of decoder-only models and the common sentiment, albeit anecdotal, that at larger scales, the complexity added by both encoders and decoders is not worth the minor improvements it *might* bring.

Decoder-only Models. This architecture, popularized by the GPT series [184, 185, 22], uses only the decoder stack of the Transformer. Decoder-only models are inherently generative and operate in an autoregressive fashion, meaning they generate text one token at a time, with each new token prediction conditioned on the sequence of previously generated tokens. They are pre-trained on a causal language modeling (CLM) objective, which is simply the next-token prediction. This architectural simplicity and training efficiency have made the decoder-only approach the dominant paradigm for today’s largest and most capable LLMs, excelling at a wide range of text generation tasks.

The trend of scaling up LLMs by pre-training larger decoder-only models using larger and higher-quality corpora significantly enhances model capabilities for general-purpose NLP tasks such as question answering and code generation. Scaling is achieved by fine-tuning the pre-trained LLMs with instruction-following data using reinforcement learning with human feedback (RLHF). InstructGPT [162] is an early example. Other popular examples from the recent past include Gemini 2.5 [6, 39], GPT₅ [161], and Sonnet 4.5 [7]. Note that these models are *proprietary*, i.e., their model weights are not shared with the public, often dubbed “closed-weights models”. The LLaMA [218, 219, 52], Vicuna [33], Zephyr [220], and Qwen [11, 12, 212] series are among the most popular and effective open-weight variants on several benchmarks, albeit this list is ever-growing and we most definitely only capture some of the most notable.

2.3 Neural Models for Text Ranking: A Deep Dive

Over the past decade, the deep learning revolution has brought tremendous excitement to the information access community. Although ranking models based on machine learning have been well studied since the mid-2000s in the context of learning to rank [128, 112], the paradigm was heavily driven by manual feature engineering. Continuous vector space representations coupled with advancements from neural models promised to obviate the need for handcrafted features. Well-known early neural ranking models include DRMM [74], DUET [146], KNRM [240], and Co-PACRR [87]; the literature is too vast for an exhaustive review here, and thus we refer readers to recent overviews [159, 145].

The introduction of BERT [50], however, marked the beginning of a new era in neural ranking models, beginning with Nogueira and Cho [154]. Soon after, we witnessed an increasing dominance of reranking models based on pre-trained encoder-only transformers [47, 136, 2, 111]. In parallel, retrieval using approximate nearest-neighbor search on learned dense representations also emerged as a promising direction [107, 95, 65, 241, 98, 122], with models often competing with hard-to-beat traditional methods like BM25, and some improving over it in target tasks. For more details, the survey by Lin et al. [120] provides an overview of these developments in encoder-only transformers, both as retrievers and rerankers.

2.3.1 Bi-Encoders for First-Stage Retrieval

A popular solution to the limitations of BoW models is the dual encoder, or bi-encoder [72, 95, 31]. The success of DPR [95] and ANCE [241] revitalized bi-encoders in the new era of BERT. The goal of a bi-encoder is to learn a transformer-based mapping from queries and documents into dense fixed-width vectors wherein the inner product between the query vector and the relevant document vector is maximized. A lot of work has gone into understanding and better learning such a mapping [62, 63, 83, 122, 123]. For instance, Lin et al. [122] train a bi-encoder by using on-the-fly knowledge distillation from a ColBERT [98] reranker as the teacher model that computes soft labels for in-batch negatives.

Building on T5’s success in NLU tasks (including text ranking; see Chapter 3), Ni et al. [152] scaled up dual encoders by pre-training T5-style models for retrieval. The Generalizable T5 Retriever (GTR) [153] extends this idea to ad hoc retrieval tasks, with models being pre-trained on large-scale question-answering datasets followed by fine-tuning on text ranking tasks like MS MARCO [13].

The Importance of Hard Negatives. Prior work shows that the selection of negative examples is critical in training bi-encoders. Karpukhin et al. [95] compare the model effectiveness, training with random negatives, BM25 negatives, and in-batch negatives, and find that a mixture of BM25 and in-batch negatives yields better models. Xiong et al. [241] prove theoretically that local negatives are suboptimal for dense retrieval learning and proposed preparing global negatives using the current dense retrieval model in parallel with training. Qu et al. [182] also prepare denoised hard negatives to improve effectiveness further. Other work confirms the importance of hard negatives and proposes various strategies for mining them effectively [253, 63, 62, 123, 83]. TCT-ColBERTv2 [123] use a “HN+” hard-negative mining strategy to improve further their base dual encoder, a strong baseline we will use in Chapter 4.

2.3.2 Cross-Encoders for Reranking

After an initial retrieval stage, candidate passages are passed through “cross-encoders”, which jointly encode the query and a candidate passage with full *cross-attention* to produce a single relevance score, hence the name. The first effective cross-encoder for reranking, monoBERT [154], quickly emerged after the introduction of BERT [50] itself. It followed the approach recommended by the BERT authors to handle query–passage input pairs and demonstrated a huge leap in effectiveness on the MS MARCO passage ranking [13] and TREC CAR [51] datasets. This model represents the first instance of what has later become known as cross-encoders for retrieval, a class of reranking models that includes MaxP [47], CEDR [136], Birch [2], PARADE [111], and many others. While monoBERT showed great improvements on passage retrieval, it still left a lot to be desired in terms of handling longer documents (limited by context length, lack of high-quality training data). Much of the immediate follow-up work attempted to address these limitations [47, 136, 2, 111].

In addition to cross-encoders relying on BERT-based pre-trained Language Models (pLMs), another genre of cross-encoders takes advantage of the sequence-to-sequence pLM paradigm. Examples of these are monoT5 [158] and duoT5 [172], which use T5 [186], the extensively pre-trained encoder–decoder language model. Cross-encoders demonstrate state-of-the-art effectiveness in various domains, even in a zero-shot setting [255, 169, 170, 213], and form a vital backbone in tasks like fact verification [171, 92] and question answering [248]. Other work has focused on improving pLMs with IR-specific pre-training objectives, such as the representative words prediction (ROP) task in PROP [132] and B-PROP [131].

Cross-encoders can be further categorized based on how they approach the ranking task, leading to three distinct training and inference paradigms:

Pointwise Rerankers. Initially, transformer-based rerankers focused on a pointwise approach, assessing document relevance in isolation. Models such as monoBERT [154] treat ranking as a classification or regression problem, scoring each query–document pair independently. These models take a query and a single document as input and produce a relevance score for that pair. While simple and straightforward to implement, pointwise approaches have the limitation that they cannot directly compare documents or capture relationships between different candidate documents. The final ranking is produced by sorting documents according to their individual scores, which may not always produce the optimal overall ranking. We extended this to sequence-to-sequence models with monoT5 [158], which remains an effective baseline. Some pointwise models are trained with listwise loss functions. These methods often incorporate hard negatives (several documents for a query) to improve system effectiveness. Gao et al. [67] successfully incorporated hard negatives in cross-encoder training by proposing the Localized Contrastive Estimation (LCE) loss, which significantly improves reranking effectiveness, especially when training instances follow the same distribution as the results returned by the first-stage retrievers. Recent advancements have adopted pairwise and listwise losses in cross-encoder models [67, 261], enhancing reranking accuracy, particularly in high-precision contexts.

Pairwise Rerankers. In the pairwise paradigm, reranking models estimate whether one document is *more relevant* than another document for a particular query, addressing some of the limitations of pointwise approaches by directly comparing pairs of documents. Models like duoBERT [156] introduced pairwise reranking with encoder-only models, taking a query and two documents as input and predicting which document is more relevant to the query. This approach can capture relative relevance relationships more effectively than pointwise methods and can potentially produce better rankings. However, pairwise approaches require a quadratic number of comparisons to rank a list of documents, making them computationally expensive for large candidate sets. Additionally, aggregating pairwise comparisons into a final ranking is complex and may not always produce consistent results.

Listwise Rerankers. The listwise paradigm represents the most sophisticated approach, generalizing the pairwise paradigm to directly reorder an entire list of input documents in terms of relevance. These models can capture inter-document relationships, consider the global context of the ranking problem, and produce rankings that are optimized for the entire list rather than individual pairs. Two contemporary works that were starting points for this approach are LRL [135] and RankGPT [207], where an LLM is prompted to reorder a list of documents in order of relevance to the query. More in Section 2.5.

2.4 Generative Information Access

While the retrieve-and-rerank paradigm has proven highly effective, recent research has explored more direct, generative approaches to information access, leveraging the power of large frontier sequence-to-sequence models.

2.4.1 Generative Retrieval

Generative retrieval seeks to replace the entire information retrieval process with a single sequence-to-sequence model capable of mapping queries directly to relevant document identifiers [143]. Differentiable Search Index (DSI) [211] first demonstrated the potential of this paradigm, where T5 is used to parameterize an end-to-end search system, with the model parameters encoding all information about the corpus. Zhuang et al. [263] explored the effectiveness of DSI on small subsets of MS MARCO and XOR QA [10]. Neural Corpus Indexer [234] builds on the success of DSI and introduces a combination of more input variants and architectural additions. Many have explored various document identifier designs, including document substrings [15], metadata-based approaches [260, 265], and learned quantization [187, 206]. Chen et al. [30] proposed a distillation approach on top of DSI. Note that this generative retrieval paradigm differs from the “generative retrieval models” of Lesota et al. [108], which instead refer to probabilistic models of relevance estimation rather than neural sequence-to-sequence approaches.

However, none of these works explored training or evaluating generative retrieval systems on corpora larger than $O(100K)$ documents. Given that the generative retrieval paradigm has extended beyond traditional retrieval into areas such as recommender systems [187] and vision [258], this remains a crucial area for study.

2.4.2 Retrieval-Augmented Generation (RAG)

RAG in the context of LLMs has been used to generate entirely new, non-extractive answers to user queries [76, 110, 88, 18]. In this paradigm, integrating external knowledge improves factual accuracy and mitigates issues such as hallucination [97, 70]. Open-source frameworks such as LangChain [27], LlamaIndex [125] were quickly developed to support the construction and evaluation of RAG systems. However, they lack the academic rigor to build on solid ground, often just releasing models with little to no benchmarking. RAG benchmarks have been developed to evaluate their effectiveness on different tasks, ranging from factoid QA [249, 28] to long-form answer generation [242, 77], and across

various sources like knowledge graphs [177, 57], multilingual documents [214, 215], and more [235, 216, 91, 85]. We touch more on benchmarking and evaluation in [Section 2.6](#).

2.4.3 Synthetic Data Generation

Another related area of research involves using LLMs to create synthetic data for training other models. Existing approaches often apply synthetic query generation to improve retrieval effectiveness. Nogueira et al. [157] first leveraged a vanilla (not pre-trained) sequence-to-sequence Transformer to train a model that can map passages to queries. The quality of synthetic data was further improved with T5 [155], and Ma et al. [133] showed its effectiveness across a wide range of corpora. Other methods like InPars [17, 19], Promptagator [48], and HyDE [68] also use LLMs to generate relevant training pairs for retrievers or rerankers. These methods utilize LLMs indirectly as a data generation tool rather than as a direct component of the retrieval or ranking process at inference time.

2.5 Zero-Shot Reranking with LLMs

A more recent approach focuses on using LLMs as rerankers in zero-shot scenarios, where no task-specific training data is provided. This approach builds on recent advances using LLMs as rerankers in multi-stage pipelines, mainly leveraging prompt engineering [207, 135, 181, 264, 262]. These models represent a paradigm shift in how reranking is approached, leveraging the instruction-following capabilities of LLMs to generate rankings as text output directly. The models take a query and a list of passages as input, formatted as a natural language prompt, and directly generate the reranked list as output text. This approach has several advantages: it can leverage knowledge encoded during pre-training and fine-tuning; handle complex ranking instructions and criteria; and potentially reason about ranking in ways traditional neural models cannot.

However, these prompt-decoders have almost exclusively relied on large, proprietary, and closed-source LLMs such as GPT₄ or PaLM [36]. This reliance on proprietary models creates several problems for the research community. First, it limits reproducibility, as researchers cannot fully control or understand the models they are using. Second, it creates dependencies on commercial services that may change, become unavailable, or become prohibitively expensive. Third, it concentrates power in the hands of large tech companies, potentially limiting innovation and creating barriers to entry for researchers and organizations with limited resources.

We introduced RankVicuna [175], hoping to rectify this by leveraging instruction fine-tuning to distill the listwise reranking effectiveness of RankGPT_{3.5} [207] down to a smaller 7B-parameter model. Here, we introduced the term “prompt-decoders”, contrasting with BERT-style cross-encoders, to describe LLM-based rerankers operating in a zero-shot fashion. In this paradigm, instruction data can be generated from a model like RankGPT and then used to *directly* instruction fine-tune an LLM on the reranking task, potentially addressing the limitations of relying on large proprietary models. Chapter 6 introduces RankZephyr and a lot of the training principles followed in RankVicuna; we would like to think that RankVicuna walked so that RankZephyr can fly, and hence shift focus to it instead.

2.6 Benchmarks and Evaluation Methodologies

The shift from document retrieval to answer generation necessitates a fundamental shift in evaluation methodologies, representing one of the most significant challenges facing the information access community today. Traditional IR evaluation, which has been refined over decades of research and is heavily reliant on pooling and human relevance judgments at the document level [79, 227, 23, 24], is ill-suited for assessing the fine-grained quality of generative outputs. The traditional evaluation paradigm assumes that the system output consists of a ranked list of documents, and evaluation focuses on whether relevant documents appear high in the ranking. This approach works well when the goal is to help users find relevant documents, but it breaks down when the system is expected to generate coherent, factually accurate, and comprehensive answers.

The challenges of evaluating generative systems are multifaceted and complex. A generated paragraph might be fluent, well-written, and engaging, but factually incorrect in subtle ways that are difficult to detect. Conversely, a response might contain accurate information but miss key details that are essential for fully answering the user’s question. Traditional metrics like precision and recall, which work well for document retrieval, do not easily translate to the evaluation of generated text. Furthermore, the evaluation of generated text often requires understanding not just whether information is present, but whether it is presented clearly, whether it is properly contextualized, and whether it addresses the specific information need expressed in the query. The increasing complexity of retrieval and generation systems necessitates more sophisticated evaluation methodologies beyond simple relevance judgments.

The research community has explored various approaches to address these challenges, with varying degrees of success. Some approaches have attempted to adapt traditional

IR metrics to the generative setting, while others have developed entirely new evaluation frameworks. Automatic metrics borrowed from other fields, such as BLEU [165] and ROUGE [73] from machine translation and summarization, have been applied to RAG evaluation, but often fail to capture the nuances of factual accuracy and completeness that are critical for information access applications.

More recently, the research community has increasingly turned to human evaluation, recognizing that the quality of generated answers often requires human judgment to assess properly. However, human evaluation is expensive, time-consuming, and difficult to scale to the large numbers of queries and systems that are typical in information retrieval research. Furthermore, human evaluation can be inconsistent, with different annotators sometimes reaching different conclusions about the quality of the same generated answer. The subjectivity inherent in evaluating generated text makes it challenging to establish reliable benchmarks and to compare systems.

2.6.1 Corpora and Topics

Corpus Selection. The quality of evaluation is fundamentally tied to the quality of the underlying data. Current RAG datasets are constructed using the English Wikipedia as the document collection. However, their scale is limited to provide rich and comprehensive information to support RAG systems. A prominent option, ClueWeb22 [163] offers an extensive collection of 22 billion curated web pages, previously utilized in parts by TREC tracks such as the TREC Conversational Assistance Track (CAST) [164] and the TREC Interactive Knowledge Assistance Track (iKAT) [4]. However, ClueWeb22 remains gated and serves as a significant hurdle for researchers with limited computational resources or those unable to obtain access credentials. Another alternative is the MS MARCO V2 document collection, which was used primarily in the TREC Deep Learning (DL) track. The open availability of MS MARCO makes it an attractive option for researchers seeking to develop and evaluate RAG systems at scale.

Topic selection. Recently, there has been a surge in datasets providing topics with long-form answers for evaluating RAG systems. ASQA [205], ELI5 [55], and QAMPARI [5] were utilized for evaluation in the Automatic LLMs’ Citation Evaluation (ALCE) framework [70]. Similarly, related long-form QA datasets include AquaMuse [102], ExpertQA [139], and TruthfulQA [124]. Another recently introduced dataset is ClapNQ [193], created from the subset of Natural Questions (NQ) [103] and HAGRID [94] built on a subset of MS MARCO Dev [13]. Almost all previous datasets are built on the English Wikipedia. In

contrast, our work deliberately avoids English Wikipedia to prevent the overfitting seen in existing retrieval benchmarks [213, 147].

2.6.2 The Nugget Evaluation Methodology

To address the challenges of evaluating complex question answering systems, the research community has turned to a classic, robust methodology from the TREC Question Answering (QA) track: nugget-based evaluation [228, 115, 116]. Introduced in the TREC QA Track in 2003 [228], the nugget evaluation methodology was originally developed to address similar challenges in evaluating complex question answering systems, where the goal was to assess whether systems could provide complete and accurate answers to factual questions. The core idea behind nugget evaluation is to decompose the ideal answer to a complex information need into a set of atomic facts, or *nuggets*, each representing a distinct piece of information that would be valuable to include in a complete answer. Voorhees [229] demonstrated the importance of constructing nuggets to evaluate definition questions, thereby aligning complex question answering with retrieval-augmented generation. Further studies have refined and extended the basic nugget evaluation methodology, for example, as Summarization Content Units (SCUs) [149], with further refinements for QA evaluation [117, 49, 118]. Lin and Demner-Fushman [115, 116] explored automated approaches, and Pavlu et al. [167] introduced a nugget-based methodology for evaluating retrieval itself.

The nugget methodology provides several key advantages for evaluation. First, it provides a more granular measure of information recall than traditional document-level metrics, allowing evaluators to assess not just whether a system found relevant information, but whether it included specific facts that users would find valuable. Second, it provides a framework for assessing completeness by measuring what proportion of important nuggets are covered by a system’s response. Third, it provides a way to handle the fact that there may be multiple valid ways to express the same information by focusing on the underlying facts rather than the specific wording used to express them.

The nugget methodology has been extensively validated and refined through years of use in TREC evaluations. Research has shown that nugget-based evaluation is stable and reliable [118, 49], producing consistent results across different annotators and evaluation settings. The methodology has been shown to correlate well with user satisfaction and to provide meaningful distinctions between systems of different quality levels. Furthermore, the methodology has been adapted and extended for various types of information needs and evaluation scenarios, demonstrating its flexibility and robustness.

However, the traditional nugget evaluation methodology has significant limitations when applied to modern RAG systems. The original methodology relied heavily on manual annotation, with human experts identifying nuggets and manually checking system responses for the presence of these nuggets. This manual process is extremely time-consuming and expensive, making it challenging to apply to large-scale evaluations that are common in modern information access settings. Furthermore, the process can be inconsistent, with different annotators sometimes identifying different nuggets or reaching different conclusions about whether a nugget is present in a system response.

2.6.3 Automated Evaluation with LLMs

In recent years, the research community has explored using LLMs themselves as evaluators, a trend often termed LLM-as-a-Judge [259, 71, 61, 69]. This approach leverages the natural language understanding capabilities of LLMs to assess the quality of generated text, potentially providing a more scalable alternative to human evaluation. Using LLM judges [259] in place of human judges has gained popularity for evaluating different aspects of RAG, such as factuality [101], retrieval quality [197], robustness [203], and contextual faithfulness [151]. Researchers have also used LLMs to perform pairwise comparisons [183, 77, 179] and incorporate grading rubrics [233]. LLM-based evaluation has shown promise in various domains, with studies demonstrating that LLM judgments can correlate well with human assessments in certain settings.

However, LLM-as-a-Judge approaches also have significant limitations. First, many of these approaches can be black box, making it difficult to understand why a particular judgment was made or to identify potential biases in the evaluation. Furthermore, these approaches may lack grounding in established evaluation theory, making it unclear whether they are measuring the aspects of quality that are most important for users. There is also the concern that LLM-based evaluation may be biased toward outputs that are similar to the training data of the LLM being used for evaluation, potentially favoring certain types of responses over others.

LLM automation extends to the nugget methodology; we first cover this in a preliminary draft [178] of work covered in Chapter 8. Since we have noted several works building on automated nugget evaluation [141, 3, 216]. Frameworks like FactScore [144] evaluate biographies in terms of supported atomic facts. The RUBRIC framework [56] evaluates answers based on an LLM-generated rubric. More generally, one promising direction is to explicitly ground the power of LLMs in the well-understood and validated principles of nugget-based evaluation. Rather than asking LLMs to make holistic quality judgments,

such approaches could use LLMs to perform specific, well-defined tasks within the nugget evaluation framework: identifying important facts that should be included in answers and checking whether these facts are present in system responses. This approach could combine the scalability advantages of LLM-based evaluation with the theoretical grounding and validation of the nugget methodology. These automated processes are often validated and calibrated against high-quality human annotations, and studies have established correlations between nugget-based metrics and human preferences even in arena-style chats [201].

Nugget evaluation is not everything when it comes to RAG/LLM response evaluation. Existing RAG evaluation frameworks assess responses along multiple dimensions. For example, RAGAs [53] introduced metrics to evaluate faithfulness, answer relevance, and context relevance. In Thakur et al. [217], we explore RAG evaluation along the dimension of support. ARES [196] demonstrated that fine-tuning an open-source LLM with a small amount of training data suffices to produce an effective automatic judge.

The change from document-evaluation metrics to automated nugget-based evaluation represents a necessary response to the shift from retrieval to generation in information access. Yet establishing reliable, scalable, and grounded evaluation remains an open challenge and will perhaps be the focus of information access evaluation in the coming years. Having surveyed the landscape of existing approaches and identified key open problems in model development, benchmarking, and evaluation, we now turn to the contributions of this thesis, which address these challenges through new architectures, training methodologies, benchmarks, and evaluation frameworks.

Chapter 3

Expando-Mono-Duo: A Design Pattern for Supervised Reranking

Modern information retrieval systems have long relied on multi-stage ranking architectures to balance effectiveness and efficiency. In this paradigm, a query first retrieves a large set of candidates using a first-stage retriever, then progressively refines this set through increasingly effective and computationally expensive rerankers. We present a refinement of this general approach that has been empirically demonstrated to work well for multiple *ad hoc* retrieval tasks in different domains. In contrast to most current approaches that build on encoder-only pretrained transformers such as BERT [50], our approach instead relies on pretrained sequence-to-sequence transformers within a multi-stage ranking architecture. In our case, we use T5 [186], but our approach can be extended to other sequence-to-sequence models such as BART [109] and Pegasus [256] as well. The key features of our approach are as follows:

- Document expansion using a sequence-to-sequence model to enrich keyword representations of texts from the corpus before indexing (“Expando”); we’ve also previously called this approach “doc2query”.
- Initial keyword-based retrieval (also called first-stage retrieval or candidate generation) using standard inverted indexes.
- A two-stage reranking pipeline comprising a pointwise reranker (“Mono”) followed by a pairwise reranker (“Duo”), both built on pretrained sequence-to-sequence models.

Developed before the widespread adoption of LLMs for ranking, the complete pipeline achieved the then state-of-the-art effectiveness across multiple benchmarks, and its influ-

ence persists — the system continues to serve as a strong baseline in the community five years later. Yet, this success came with crucial lessons about the inherent costs and limitations of such approaches, lessons that would motivate the subsequent chapters of the Component Challenge. Our investigation addresses four key questions:

- (RQ1) How can a single sequence-to-sequence model architecture be adapted to perform three distinct ranking tasks: document expansion (“Expando”), pointwise reranking (“Mono”), and pairwise reranking (“Duo”)?
- (RQ2) What is the individual contribution of each component to the overall effectiveness of the pipeline?
- (RQ3) How does the complete Expando-Mono-Duo pattern perform against the then state-of-the-art methods across diverse ad-hoc retrieval benchmarks, including MS MARCO and the TREC Deep Learning tracks?
- (RQ4) Can this design pattern achieve high effectiveness in a zero-shot setting, without being fine-tuned on the target collection?

Through extensive experiments on five benchmark datasets, we demonstrate that the Expando-Mono-Duo pattern achieves effectiveness at or near the then state of the art. This chapter details the implementation of each component, quantifies its contribution through ablation studies, and validates the robustness of the overall design, providing a cohesive framework for building highly effective supervised ranking systems to date.

3.1 Expando-Mono-Duo with T5

In our formulation, a multi-stage ranking architecture comprises a number of stages, which we denote H_0 to H_N . Except for H_0 , which retrieves k_0 candidates based on keyword search (i.e., from an inverted index), each stage H_n receives a ranked list R_{n-1} comprising k_{n-1} candidates from the previous stage. Each stage, in turn, provides a ranked list R_n comprising k_n candidates to the subsequent stage, with the apparent requirement that $k_n \leq k_{n-1}$. The ranked list generated by the last stage H_N in the pipeline is designated for final consumption, i.e., shown to the human searcher or fed to a downstream application. In this context, k_N is the desired k in the original ranking task; all the other k_n ’s $n \in \{0 \dots N - 1\}$ are internal system settings. As a simple example, for the MS MARCO passage ranking task [13], where the official metric is MRR@10 (that is, the evaluation

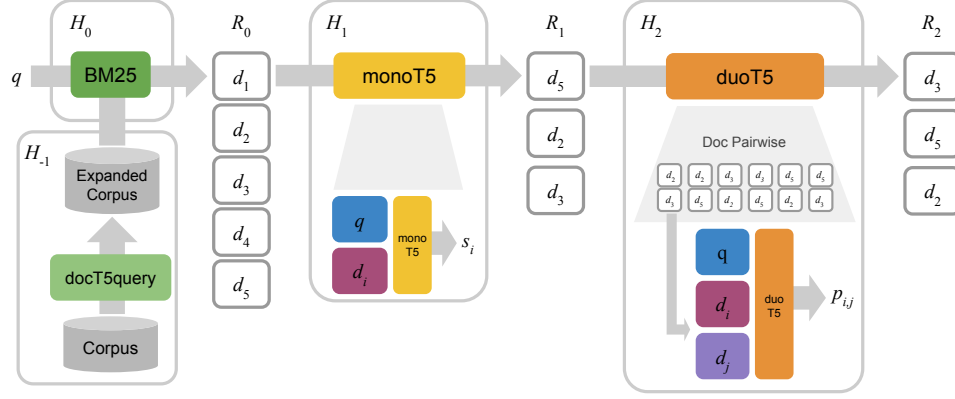


Figure 3.1: Illustration of our multi-stage ranking architecture. Before indexing, we perform document expansion, denoted H_{-1} . In stage H_0 , given a query q , the top k_0 ($= 5$ in the figure) candidate documents R_0 are retrieved using BM25. In stage H_1 , monoT5 produces a relevance score s_i for each pair of query q and candidate $d_i \in R_0$. The top k_1 ($= 3$ in the figure) candidates with respect to these relevance scores are passed to stage H_2 , in which duoT5 computes a relevance score $p_{i,j}$ for each triple (q, d_i, d_j) . The final output R_2 is formed by reranking the candidates according to these scores (see Section 3.1.4 for how these pairwise scores are aggregated).

metric only considers the first ten hits), a popular setting is to retrieve 1000 hits using keyword search, which are then fed to a reranker to produce the top 10 final results [154].

Before building the inverted index that feeds first-stage retrieval H_0 , with “Expando”, we first perform document expansion on the input corpus to enrich its representation; we denote this as the H_{-1} stage. The augmented documents are then indexed exactly as before; the only impact of document expansion is to (hopefully) provide a richer set of candidate documents for downstream rerankers to process. The output of first-stage retrieval H_0 is then passed to the two-stage reranking pipeline comprised of monoT5 H_1 (“Mono”) and duoT5 H_2 (“Duo”). We describe each component of the overall architecture (see Figure 3.1) in detail below. Here, our narrative focuses on the overall design, and we defer details of our experimental settings for each task to Section 3.2. To support replicability, all our implementations are open-sourced in the Pyserini IR toolkit [119]¹

¹<http://pyserini.io/>

and the neural reranking toolkit RankLLM [200].²

3.1.1 H_{-1} : Document Expansion with T5

The idea behind document expansion is to enrich each document (more generally, each text from the corpus) with additional text representative of its content for the purposes of retrieval. In our particular implementation, we leverage a corpus of query–relevant passage pairs to train a sequence-to-sequence model that maps passages to queries. That is, given a segment of text, the model predicts queries that the text can potentially answer. For example, consider the following passage:

July is the hottest month in Washington DC with an average temperature of 27°C (80°F) and the coldest is January at 4°C (38°F) with the most daily sunshine hours at 9 in July. The wettest month is May with an average of 100mm of rain.

In this example, the “target query” (i.e., from our training data) is “What is the temperature in Washington?” In the MS MARCO dataset we used for training, the queries tend to be relatively well-formed natural language utterances, but for generality, we refer to model output as queries. After learning from a large corpus of such examples (details later), our model is able to predict the query “What is the weather in Washington DC?”

These predictions are then directly appended to the original (source) document; once this expansion has been performed for every document, the new augmented corpus is indexed, as before. Since document expansion occurs before indexing, we refer to this as the H_{-1} stage. Note that in this case, the predicted query contains a term (“weather”) that is not present in the original passage. This has the effect of enriching the passage and increasing the likelihood of matching a broader range of queries for which this passage would be relevant. Even in the case where the predicted queries repeat words that are already present in the text, it achieves the effect of increasing the weight on the term, highlighting its importance.

This idea, dubbed “doc2query”, was first proposed in Nogueira et al. [157], but in this work we take advantage of the pretrained sequence-to-sequence model T5, as described in Nogueira and Lin [155]. Here, we refer to the document expansion model with T5 as doc2query-T5, to disambiguate it from the original doc2query proposal [157], which used vanilla (not pre-trained) transformers.

²<http://rankllm.ai/>

3.1.2 H_0 : Keyword Retrieval

The stage H_0 receives as input the user query q and produces the top k_0 candidates R_0 . In multi-stage ranking architectures, this is called first-stage retrieval or candidate generation. In our implementation, the query is treated as a bag of words for ranking documents from the corpus using a standard inverted index based on BM25 [192]. Depending on the exact setting and task (see Section 3.2), keyword search may additionally exploit query expansion using pseudo-relevance feedback. All our experiments used the Pyserini IR toolkit,³ which is the Python wrapper to Anserini [245, 246],⁴ itself built on the popular open-source Lucene search engine.

3.1.3 H_1 : Pointwise Reranking with monoT5

In stage H_1 , documents retrieved in H_0 are reranked by a pointwise reranker called monoT5. The model estimates a score s_i quantifying how relevant a candidate $d_i \in R_{n-1}$ is to a query q . That is:

$$P(\text{Relevant} = 1 | d_i, q). \quad (3.1)$$

This approach is called relevance classification [120], or alternatively, a pointwise approach in the parlance of learning to rank [128, 112]. Naturally, we expect that the ranking induced by these scores yields a higher metric (e.g., MAP or MRR) than the scores from the input ranking (i.e., the output of H_0). At a high level, monoT5 is a sequence-to-sequence adaptation of the monoBERT model proposed by Nogueira and Cho [154] and further detailed in Lin et al. [120].

Unlike monoBERT [154], monoT5 uses T5 [186, 158], a popular pretrained sequence-to-sequence transformer model. In this model, all target tasks are cast as sequence-to-sequence tasks. Specifically, ranking is performed using the following input sequence template:

$$\text{Query: } q \quad \text{Document: } d \quad \text{Relevant:} \quad (3.2)$$

where q and d are the query and document texts, respectively. The model is fine-tuned to produce the tokens “true” or “false” depending on whether the document is relevant or not to the query. That is, “true” and “false” are the “target tokens” (i.e., ground truth predictions in the sequence-to-sequence transformation).

³<http://pyserini.io/>

⁴<http://anserini.io/>

At inference time, to compute probabilities for each query–document pair (in a reranking setting), we apply a softmax only on the logits of the “true” and “false” tokens. Following Nogueira et al. [158], we rerank the documents according to the probabilities assigned to the “true” token. Note that while H_0 uses a corpus enriched by document expansion, documents in R_0 consist of original texts that do not include the predicted queries.

As discussed in Lin et al. [120], one recurring theme in the application of early transformer models to text ranking is the handling of texts that are longer than the input sequences that the models were designed to handle (typically, 512 tokens circa 2020). Building on previous work [80, 25, 47], we were able to devise simple yet effective solutions to address this issue. Since these solutions are corpus dependent, we save detailed discussion for Section 3.2.

3.1.4 H_2 : Pairwise Reranking with duoT5

The output R_1 from the previous stage is used as input to the pairwise reranker we call duoT5. In this pairwise approach, the reranker considers a pair of documents (d_i, d_j) and estimates the probability $p_{i,j}$ that candidate d_i is more relevant than d_j to query q :

$$P(d_i \succ d_j | d_i, d_j, q), \quad (3.3)$$

where $d_i \succ d_j$ is a commonly adopted notation in IR for stating that d_i is *more relevant* than d_j (with respect to the query q).

The basic idea behind duoT5 was originally developed in Nogueira et al. [156], but in the context of BERT (not surprisingly, called duoBERT). As the name suggests, duoT5 is also based on T5. In this case, the reranker takes as input the sequence:

$$\text{Query: } q \quad \text{Document0: } d_i \quad \text{Document1: } d_j \quad \text{Relevant:} \quad (3.4)$$

The pairwise sequence-to-sequence model is fine-tuned to produce the token “true” if document d_i is more relevant than d_j , and “false” otherwise.

At inference time, we aggregate the pairwise scores $p_{i,j}$ so that each document receives a single score s_i . We investigated four different aggregation techniques:

$$\text{SUM : } s_i = \sum_{j \in J_i} p_{i,j}, \quad (3.5)$$

$$\text{SUM-LOG : } s_i = \sum_{j \in J_i} \log p_{i,j}, \quad (3.6)$$

$$\text{SYM-SUM} : s_i = \sum_{j \in J_i} (p_{i,j} + (1 - p_{j,i})), \quad (3.7)$$

$$\text{SYM-SUM-LOG} : s_i = \sum_{j \in J_i} (\log p_{i,j} + \log (1 - p_{j,i})), \quad (3.8)$$

where $J_i = \{0 \leq j < k_1, j \neq i\}$.

The SUM method measures the pairwise agreement that candidate d_i is more relevant than the rest of the candidates $\{d_j\}_{j \neq i}$. The SUM-LOG method is a variant of SUM that penalizes smaller values of $p_{i,j}$ by virtue of the logarithm. The methods SYM-SUM and SYM-SUM-LOG are variants of SUM and SUM-LOG, respectively. In these two cases, we also add the score corresponding to the probability $p_{j,i}$ assigned to the “false” token. The goal of these variants is to ensure a more stable and accurate scoring. We empirically compared these variants and examined the impact of different parameter settings in [Section 3.3.1](#).

The candidates in R_1 (i.e., the output from monoT5 stage H_1) are reranked according to their scores s_i to obtain the final list of candidates R_2 . Note that duoT5 is computationally expensive, where the running time is dominated by the need to perform inference $k_1 \times (k_1 - 1)$ times. Thus, in practice, it is tractable to run duoT5 reranking only on a relatively small number of candidates k_1 (detailed settings are presented in [Section 3.2](#)). The implication of this design is that duoT5 is primarily aimed at improving early precision, i.e., the quality of results high in the ranked list.

While in principle multi-stage ranking architectures can have an arbitrary number of stages, in our current design, the output R_2 is intended for final consumption, e.g., by a human searcher or fed to a downstream application. In our evaluations, we report metrics computed over the output of duoT5.

3.2 Experimental Settings

We have empirically validated the Expando-Mono-Duo design pattern in five different settings: the MS MARCO passage and document ranking tasks, the passage and document ranking tasks in the TREC 2020 Deep Learning Track, and the TREC-COVID challenge. In this section, we describe detailed experimental settings for each task, which correspond to how our Expando-Mono-Duo pattern is “instantiated” in different contexts. We do a deeper dive here to serve as valuable context for later chapters that also build on these datasets.

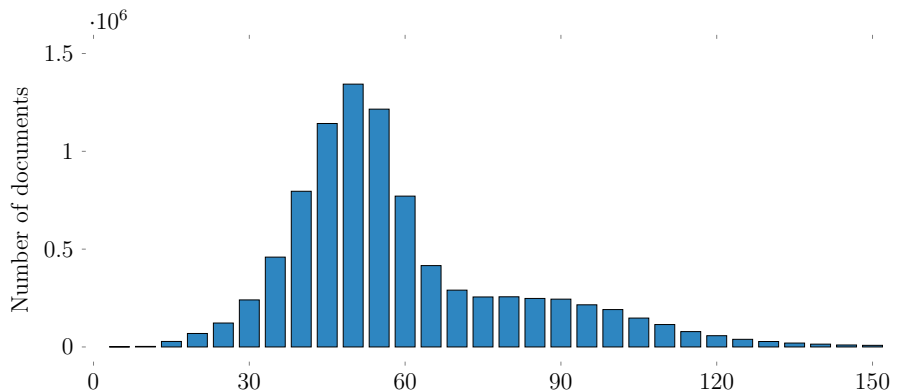


Figure 3.2: Distribution of passage lengths in the MS MARCO passage corpus.

3.2.1 MS MARCO Passage Ranking

We trained and evaluated our models on the MS MARCO passage ranking dataset [13], which provides a corpus comprised of 8.8M passages gathered from Bing search engine results. The mean length of each passage is 56 tokens (median: 50, max: 362), and the distribution of passage lengths is shown in Figure 3.2.⁵ The training set contains approximately 532.7K query–relevant passage pairs, where each query has one relevant passage on average. The development set contains 6980 queries, and the test set contains 6837 queries, but relevance labels are only publicly available for the development set; evaluation on the held-out test set requires submission to the leaderboard.⁶

We also evaluated our models on the passage ranking task of the TREC 2020 Deep Learning Track [42], which has 54 queries with graded relevance judgements (on a four-point scale) gathered via traditional pooling techniques [225] (with ~ 210 judgments per query). The TREC evaluation used the same corpus as the MS MARCO passage ranking task, and thus the two evaluations differed primarily in the number of queries and the abundance of relevance judgments per query. The MS MARCO relevance judgments (with around one passage labeled relevant per query) sometimes referred to as “sparse” judgments, contrasting with the richer “dense” judgments gathered via pooling by TREC assessors. We adopt this terminology in our discussions.

Note that in our experiments, the same exact settings were used for both the MS

⁵Tokenization here is performed using Python’s string `split` method. Note that T5 uses different tokenization, and therefore these lengths do not correspond to model input lengths.

⁶<http://www.msmarco.org/>

MARCO passage ranking task and the TREC 2020 Deep Learning Track. Specifically, we did not take advantage of relevance judgments from the TREC 2019 Deep Learning Track for either evaluation.

“Expando” and Keyword Retrieval Settings. In the first stage of our pipeline, we expanded all documents in the MS MARCO passage corpus with queries predicted by our doc2query-T5 model [155], which was prepared as follows: Starting with a publicly available checkpoint, we fine-tuned T5 with (query, relevant passage) pairs from the MS MARCO passage ranking dataset with a constant learning rate of 10^{-3} for 4K iterations with batches of 256, which corresponds to two epochs with the training data. We used a maximum of 512 input tokens and 64 output tokens. In the MS MARCO dataset, none of the inputs or outputs had to be truncated using these length settings (see Figure 3.2). Similar to Nogueira and Lin [155], we found that the top- k sampling decoder [54] produced better queries (i.e., leading to higher ranking effectiveness) than beam search. We used $k = 10$ and sampled 40 queries per document with T5-base; the large model variant did not appear to yield any improvements in retrieval effectiveness. Our online replication guide details all these settings.⁷

We used a Google TPU v3-8 to fine-tune the model and perform inference. Training took less than 1.5 hours on a single TPU. For inference, sampling five queries for each of the 8.8M passages in the corpus took approximately 40 hours on a single TPU. Note that inference is trivially parallelizable, and the inference time is linear with respect to the number of samples.

All expanded texts were then indexed with the Anserini IR toolkit. The predicted queries were appended to the original passages, without any intervening delimiters. Retrieval was performed using BM25 with the parameters $k_1 = 0.82$, $b = 0.68$, based on tuning on the development set via simple grid search. For the TREC 2020 Deep Learning Track topics, we additionally applied the BM25 + RM3 model for query expansion using pseudo-relevance feedback (with all default parameters in Anserini). This approach was described in Yang et al. [247] and has been shown to be a strong baseline, especially compared to pre-BERT neural ranking models, when evaluated on traditional TREC topics with “dense” judgements.

“Mono” Settings. As part of the MS MARCO passage dataset, the creators provided (query, relevant passage, non-relevant passage) triples, where the negative evidence came from sampling. For training monoT5, we used the “small” triples file containing 39.8M records. We fine-tuned the base, large, and 3B variants of our monoT5 models with a

⁷<http://doc2query.ai/>

constant learning rate of 10^{-3} for 100K iterations with class-balanced batches of size 128. We used a maximum of 512 input tokens and two output tokens (one for the true/false token and another for the end-of-sequence token); none of the inputs had to be truncated when using this length (see [Figure 3.2](#)). Training the base, large, and 3B models took approximately 12, 48, and 160 hours on a single Google TPU v3-8, respectively.

At reranking time, our model considered the top 1000 hits from first-stage retrieval (i.e., $k_0 = 1000$). This is a commonly used setting first proposed in Nogueira and Cho [154]. Note that the input to our reranker, here and in all our experiments, did not include the predicted (expansion) queries; that is, the goal of document expansion is to improve first-stage retrieval only. There are two reasons for this design decision: First, the predicted queries are sometimes noisy and can potentially mislead the model. Second, they add to the length of each passage, thus overflowing the model input length limitations for inference.

“Duo” Settings. We fine-tuned duoT5 using the monoT5 model trained on the MS MARCO passage data as a starting checkpoint, using the same hyperparameters as those used for training monoT5. In more detail, we also used the “small” triples file provided by the MS MARCO organizers, converted into duoT5’s input format ([Equation 3.4](#)) and fed to the model. Based on initial experiments with duoT5-base, we found that model effectiveness converged around 50K iterations. Hence, we trained duoT5-3B for 50K iterations, which took about 80 hours overall on a single Google TPU v3-8.

On the development set, we examined different numbers of candidates k_1 (i.e., output from the mono stage) that are reranked by the pairwise ranker. Note that the duo model is computationally expensive, requiring a number of inferences per query that is quadratic with respect to the number of candidate pairs. Thus, for tractability, we only ran experiments where k_1 ranged from 10 to 50, in increments of 10. The results of these explorations are described later (see [Figure 3.4](#)); in our final experiments, we used $k_1 = 50$ with SYM-SUM as the pairwise aggregation method. Note that if more than k_1 hits are requested as the output of duoT5, we simply take additional ranked output from monoT5. For example, if the user (or downstream application) requires 1000 hits, then the first 50 will come from duoT5 (assuming $k_1 = 50$), while the remaining results (rank positions 51–1000) will be the unaltered rankings from monoT5. Thus, as explained in [Section 3.1.4](#), our pairwise reranker emphasizes early precision.

3.2.2 MS MARCO Document Ranking

We also evaluated our models on the MS MARCO document ranking dataset, which uses a corpus comprised of 3.2M documents, derived from the MS MARCO passage corpus.

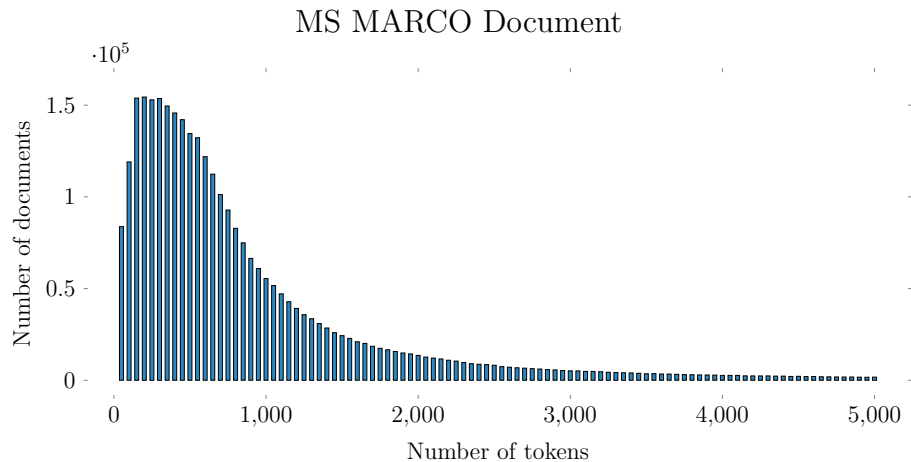


Figure 3.3: Distribution of document lengths in the MS MARCO document corpus.

The mean length of each passage is 1131 tokens (median: 584, max: 333757), and the distribution of document lengths⁸ is shown in Figure 3.3. The general setup of this task is similar to the passage ranking task, with the primary difference being the length of texts in the corpus (passages vs. full-length web documents). We see that the median length of documents is longer than the typical 512 token limit of transformer models, which presents a technical challenge for researchers to overcome. In total, the dataset contains 367K training queries and 5193 development queries; each query has exactly one relevance judgment. There are 5793 test queries, but as with the passage ranking task, relevance judgments are only publicly available for the development set queries; obtaining scores for the held-out test set requires making a submission to the leaderboard.

Similar to the passage ranking task, relevance judgments for the document ranking task are sparse. Thus, we also evaluated our models on the document ranking task of the TREC 2020 Deep Learning Track [42], which uses the same corpus but has 45 queries with judgements (on a four-point scale), gathered via traditional pooling techniques (with ~ 200 judgments per query).

For both the MS MARCO document ranking task and the TREC 2020 Deep Learning Track, we used models trained only on the MS MARCO passage ranking dataset in a zero-shot setting. That is, we did not use any data from the MS MARCO document ranking dataset to train our models; in addition, no training data from the TREC 2019 Deep

⁸Also tokenized using Python’s string `split` method.

Learning Track were used (both passage and document). However, we are clearly taking advantage of the close relationship between the document and passage datasets, and so it cannot be claimed that we are performing much meaningful transfer learning in this setup. Our motivation for reusing models from the passage ranking task was to eliminate the need to perform costly fine-tuning and to simplify experimental variables.

“Expando” and Keyword Retrieval Settings. We performed document expansion on the MS MARCO document corpus with queries predicted by the doc2query-T5 model trained on the MS MARCO passage ranking task. However, the expansion procedure differed slightly for the document corpus because many documents exceeded T5’s input length restriction of 512 tokens (unlike with the passage corpus, where all passages fall under the length limit). To address this issue, we first segmented each document into passages by applying a sliding window of ten sentences with a stride of five. Each passage was then prepended with the title of the document. Finally, we performed inference on these passages with doc2query-T5 using a top- k sampling decoder and generated ten queries (i.e., expansions) per passage.

From this, we prepared two separate indexes with Anserini:

- Per-document expansion. Each document from the original corpus was appended with all its predicted queries, and each of these augmented documents served as a unit of indexing.
- Per-passage expansion. Each passage above, with its predicted queries, served as a unit of indexing. In this case, each document from the original corpus yielded multiple “retrievable units” in the index.

For ablation purposes, we also prepared non-expanded counterparts of the above indexes: a simple document index and a passage index, segmented in the same manner as above, but without expansions.

In all cases, retrieval was performed with BM25 using Anserini’s default parameters $k_1 = 0.9$, $b = 0.4$. For the TREC 2020 Deep Learning Track topics, we also applied the BM25 + RM3 model (same as in the passage ranking case, using all Anserini default settings). The results of the keyword queries provided the candidates that fed our mono/duo rerankers, described in detail below. When evaluating document ranking in isolation, for the per-passage indexes, we constructed a document ranking by retaining the passage from each document with the highest score. Thus, to obtain a top k document ranking, we

retrieve top k' passages from the passage index (where $k' \gg k$), and then apply per-document max passage filtering to retain the top k documents. This simple yet effective approach draws from passage retrieval techniques that date back to the 1990s [80, 25].

“Mono” Settings. In the first reranker stage, we simply used the monoT5 model trained on the MS MARCO passage ranking dataset. However, it is clear from Figure 3.3 that MS MARCO documents are too long to be directly fed into our models. Instead, we applied the reranker in two different ways:

For the MS MARCO document ranking task, we used only the per-passage indexes (with and without expansion, for ablation purpose). At reranking time, we performed monoT5 inference on the top 10000 passages, since each “retrievable unit” was, in essence, already pre-segmented and could be directly fed to monoT5. In other words, we only applied inference to the passages identified by keyword search as being potentially relevant. As with the passage condition, input to our reranking pipeline did not include the predicted queries.

For the TREC 2020 Deep Learning Track, we only used results from the per-document indexes (with and without expansion, for ablation purposes). At reranking time, we first segmented each document into passages using the same technique as above (ten sentence sliding window, five sentence stride, prepending title). Note that monoT5 was fed passages from the original documents (i.e., without predicted queries). We obtained a probability of relevance for each passage by performing inference on it independently, and then selected the highest probability among the passages as the relevance score of the document. This is similar to the MaxP approach of Dai and Callan [47], although our definitions of passages differ. In this configuration, the reranker considered the top 1000 keyword search results ($k_0 = 1000$).

“Duo” Settings. As with monoT5, we used the duoT5 model trained on the MS MARCO passage ranking dataset with default parameters ($k_1 = 50$ and SYM-SUM as the aggregation method). At reranking time, we used the highest-scoring monoT5 passage as the representative passage for each document. The duoT5 model was fed pairs of representative passages from the documents under consideration to compute the pairwise scores, which were then aggregated to yield the relevance scores of each document.

With duoT5, we increased the maximum input tokens from 512 to 1024 to account for pairs of passages that were longer than the default limit of 512 tokens. We were able to do so in T5 since the models were trained with relative positional encodings [202] and thus can (hopefully) generalize to contexts larger than those seen during training. This modification, however, imposed additional computational costs that come from the model

needing to attend to twice the number of tokens; transformers exhibit quadratic complexity in both time and space with respect to input length [100].⁹

3.2.3 TREC-COVID

As one response from the information retrieval community to the global COVID-19 pandemic, the U.S. National Institute for Standards and Technology (NIST) organized the TREC-COVID challenge [226, 190], with the goal of evaluating systems that help stakeholders access reliable scientific evidence about the virus. The series of evaluations, which ran from mid-April to late-August 2020, used the COVID-19 Open Research Dataset (CORD-19) [231], curated by the Allen Institute for AI (AI2).

Due to the evolving nature of both the scientific literature and information needs, the evaluation was organized into a series of “rounds” (five in total), each of which used CORD-19 at a snapshot in time. Each round essentially comprised a separate TREC-style evaluation, with both new information needs (i.e., topics) as well as holdover information needs from previous rounds; the first round had 35 topics, and each subsequent round added five additional, yielding a set of 50 topics by round 5. One example is “serological tests that detect antibodies of COVID-19”. For each round, relevance judgments were gathered using standard pooling methodology from participant runs. Since each round used a corpus that was, to an approximation, a superset of the previous (reflecting the growing literature),¹⁰ NIST adopted a residual evaluation methodology. That is, at each round, all previously judged documents were removed entirely from consideration (this was operationalized by removing the DocIDs of previously judged documents from all new run submissions).

Our participation in TREC-COVID is detailed in Zhang et al. [255]; here we focus only on the aspects that are relevant to our Expando-Mono-Duo design pattern, as opposed to a general overview of our team’s efforts.

“Expando” and Anserini Settings. For TREC-COVID, we used the model trained for the MS MARCO passage ranking task. That is, document expansion was performed in

⁹Note that increasing the length to 1024 tokens was sufficient in this case. However, for monoT5, such an increase would still not have been sufficient to perform inference on a complete document.

¹⁰While this characterization is conceptually accurate, articles were both added and removed in each round. There were also significant complexities involving near-duplicate documents in the corpus and unstable document identifiers, for example, reflecting cases where a preprint was later published in a peer-reviewed venue.

a zero-shot manner, in that the model had not been fine-tuned on data from the target corpus (CORD-19).

Due to limited computational resources, we only generated expansions from the article abstracts. However, even the abstracts alone often exceeded T5’s input length restriction of 512 tokens. To address this issue, we used the same sliding window approach described in [Section 3.2.2](#) (window of ten sentences with a stride of five, passages prepended with the title of the article). Inference was performed on these passages using a top- k sampling decoder that generated 40 queries (i.e., expansions) per abstract passage. Interestingly, based on manual examination of the predicted queries, we found their quality to be quite good, at least to untrained experts. This was a surprising finding given the specialized domain: while it is true that T5 had been exposed to scientific text during pre-training, the model was *not* fine-tuned with any data that could be considered “in domain”.

Previous work on searching full-text articles [\[114\]](#) and results from the first round of TREC-COVID showed value in fusion approaches that combined multiple sources of evidence. Thus, starting in round 2, we adopted an approach that fused results from three different indexes:

- An index where each “document” is comprised of the title and abstract of an article in the corpus.
- An index where each “document” is comprised of the full text of an article in the corpus (including title and abstract). For articles in the corpus that do not have full text, the “document” contains only the title and abstract.
- A paragraph-level index structured as follows: each full-text article was segmented into paragraphs, based on the markup provided in the corpus, and for *each* paragraph, we created a “document” comprising the title, abstract, and that paragraph. The title and abstract alone comprised an additional “document”. Thus, a full-text article with n paragraphs yielded $n + 1$ separate retrieval units. With this index, a query is likely to retrieve multiple paragraphs from the same article; we selected the highest-scoring paragraph as the score of that article.

First-stage retrieval (i.e., the input to our reranking pipeline) comprised results combined from all three indexes using reciprocal rank fusion [\[40\]](#).

There is an additional complexity that needs explaining: one of the lessons learned in the first round of TREC-COVID was the importance of query formulation. TREC-COVID information needs (i.e., topics) were modeled after standard *ad hoc* retrieval topics,

comprising “query”, “question”, and “narrative” fields.¹¹ Each of the fields described the information needs in increasing detail: a few keywords, a sentence, and a short paragraph, respectively. Of the three fields, taking the combination of the “query” and “question” (i.e., concatenating the contents of both fields) was found to be the most effective based on empirical results from round 1 (which is consistent with the literature).

Beyond this simple approach, researchers from the University of Delaware discovered an effective technique for generating queries from the TREC-COVID topics, based on results from round 1. They began with non-stopwords from the query field and then further added named entities extracted from the question field using ScispaCy [150]. Starting in round 2, keyword queries generated using this technique (dubbed the UDel query generator) were shared with all participants *prior* to each round’s deadline. Since the query generator was not further modified, it is accurate to claim that this technique was used in rounds 2 through 5 in a blind manner.

For TREC-COVID, there were thus two commonly adopted query generation approaches. Combined with the three different index conditions discussed above, this led to two different fusion runs:

- **fusion1**: reciprocal rank fusion of results from the abstract, full-text, and paragraph indexes with the “query” and “question” fields (concatenated together) as the search query.
- **fusion2**: reciprocal rank fusion of results from the abstract, full-text, and paragraph indexes with queries generated by the UDel query generator.

Starting in round 2, these two fusion runs, **fusion1** and **fusion2**, were provided as community baselines by our team, made available for downloading prior to the deadline of each round so that participants could build on them (and indeed many did). The experimental runs we describe in this report took **fusion1** and **fusion2** as independent first-stage retrievers, reranked their output, and then combined their results using reciprocal rank fusion (details in [Section 3.3.5](#)).

Document expansion was applied in the following manner: for each of the index conditions above, we expanded the “documents” by appending all the expansion queries to form a total of three more augmented indexes enhanced by doc2query-T5. First-stage retrieval then used these augmented indexes, exactly as before (i.e., reciprocal rank fusion, as described above). As with the other tasks described in this section, downstream rerankers

¹¹These fields parallel the “title”, “description”, and “narrative” structure of standard TREC topics.

consumed the original text, i.e., we did *not* feed any of the expansion queries into the model for inference. The same two query generation approaches described above could also be applied, yielding variants of `fusion1` and `fusion2` based on the augmented indexes.

In summary, there are four possible inputs to our reranking pipeline: results from `fusion1`, `fusion2`, and the corresponding expanded versions based on doc2query-T5. We will further clarify the exact experimental conditions evaluated in [Section 3.3.5](#) when we present results.

“Mono” Settings. We used the monoT5 model fine-tuned on the MS MARCO passage dataset and then fine-tuned again on Med-MARCO, which is a subset of the MS MARCO passage dataset where only queries containing medical terms are kept [137]. For this second fine-tuning, we trained for 1K steps using batches of 128 examples and a learning rate of 10^{-4} . Although Med-MARCO is ten times smaller than the full MS MARCO passage dataset in terms of (query, relevant passage) pairs for training, we noticed a slight gain in effectiveness when fine-tuning on it (additionally).

To be clear, the only adaptation of monoT5 for TREC-COVID was the additional fine-tuning with Med-MARCO. Specifically, the model was not provided (for fine-tuning or otherwise) task-specific data (e.g., relevance judgments), even though such data became available in the later rounds.

At reranking time, we only considered article titles and abstracts from CORD-19, primarily due to length limitations already discussed. We used the same sliding window approach described above, and used the maximum score of a passage as the overall score of that article, just as with our MS MARCO document ranking experiments. Only the “question” field of the topic was used for the query text q in the input sequence template of our rerankers.

“Duo” Settings. Similar to monoT5, we used duoT5 trained on the MS MARCO passage dataset and then fine-tuned again on Med-MARCO. In this second fine-tuning, we trained for 1K steps using batches of 128 examples and a learning rate of 10^{-4} . We used the default parameters ($k_1 = 50$ and SYM-SUM as the pairwise aggregation method). At reranking time, we used the highest-scoring monoT5 passage (eight-sentence sliding window, four-sentence stride, prepending title) as the representative passage for each document. Like the MS MARCO document ranking task, we increased the maximum input tokens to 1024 to account for pairs of passages that were longer than the default limit of 512 tokens.

3.3 Results

We present experimental results from applying our Expando-Mono-Duo pattern to five *ad hoc* retrieval tasks in different domains. Empirically, we validated our approach to be at or near the state of the art across all of these tasks in 2020.

3.3.1 MS MARCO Passage Ranking

For the MS MARCO passage ranking task, [Table 3.1](#) reports MRR@10, the official metric, for various combinations of our techniques, as shown in the H_{-1} , H_0 , H_1 , H_2 columns: these provide ablations to show individual component contributions. Note that obtaining a score on the test set requires submitting a run to the leaderboard organizers, which we did not do for less interesting conditions. Overly frequent submissions to the leaderboard are actively discouraged by the organizers, and we wished to minimize “probes” of the blind held out test set. Thus, for some of the ablation and contrastive conditions, we only report scores on the development set.

At the time of submission (May 2020), the complete Expando-Mono-Duo configuration, condition (7), was the best result atop the very competitive leaderboard (which had received around one hundred submissions at the time). For comparison, we provide scores from two other runs: BM25 + BERT-large, condition (10), can be characterized as the “baseline” of transformer-based techniques [\[120\]](#). The reported effectiveness is copied from Nogueira et al. [\[156\]](#), but the general approach dates back to Nogueira and Cho [\[154\]](#). TFR-BERT [\[78\]](#), condition (11), was the best model just before our leaderboard submission. As of January 2021, our submission ranked fourth on the leaderboard; the top spot is occupied by RocketQA [\[182\]](#), condition (12). While our run is no longer the best known result, Expando-Mono-Duo remains near the state of the art in terms of effectiveness on this task, even to date.

In summary, to answer **RQ1**:

RQ1 We demonstrate that a single T5 sequence-to-sequence architecture can be successfully adapted to perform three distinct ranking tasks through prompting and fine-tuning: (i) document expansion by generating queries from passages, (ii) pointwise reranking by predicting “true” or “false” relevance labels, and (iii) pairwise reranking by predicting which of two documents is more relevant. The flexibility of the sequence-to-sequence framework allows all three components to share the same underlying architecture while serving different roles in the pipeline.

Model					Dev	Test
Expando-Mono-Duo Variants						
	H_{-1}	H_0	H_1	H_2		
(1)	-	BM25	-	-	0.184	0.184
(2)	-	BM25	monoT5-base	-	0.381	-
(3)	-	BM25	monoT5-large	-	0.393	-
(4)	-	BM25	monoT5-3B	-	0.398	-
(5)	doc2query-T5	BM25	-	-	0.277	0.272
(6)	doc2query-T5	BM25	monoT5-3B	-	0.409	-
(7)	doc2query-T5	BM25	monoT5-3B	duoT5-3B	0.420	0.408
(8)	-	BM25	monoT5-base	duoT5-base	0.392	-
(9)	-	BM25	monoT5-3B	duoT5-base	0.409	-
(10)	BM25 + BERT-large [156]				0.372	0.365
(11)	TFR-BERT Ensemble [78]				0.405	0.395
(12)	RocketQA [182]				0.439	0.426

Table 3.1: Results of Expando-Mono-Duo Variants on the MS MARCO passage ranking task, showing the official metric MRR@10 on the development and test sets. Note that scores on the test set are only available via a leaderboard submission.

Conditions (2), (3), and (4) focus on single-stage reranking using monoT5, without document expansion; these differ in the size of the underlying T5 model (monoT5-base vs. monoT5-large vs. monoT5-3B). It is apparent that increasing model sizes yields improvements in effectiveness, as expected. These results show that T5 is more effective than BERT, even factoring in model size differences, bolstering the arguments of Nogueira et al. [158] in advocating ranking with sequence-to-sequence models. In fact, a single large model, monoT5-3B, condition (4), approaches TFR-BERT, which is an ensemble of 15 BERT-large models that reranked the top 1000 passages from both BM25 and DeepCT [46].

Conditions (5), (6), and (7) include document expansion, using models based on T5-3B for reranking. We see that adding document expansion (stage H_{-1}) to enhance first-stage retrieval (stage H_0), condition (5) vs. (1), yields a large boost on the test set (a nearly 50% relative improvement); these gains carry over to the held-out test set. Note that this gain comes without the need to perform any (computationally expensive) neural inference at query time, although first-stage retrieval latency increases modestly since the expanded documents are longer.

On top of doc2query-T5, adding monoT5-3B, condition (6), and duoT5-3B, condition (7), each contributes additional cumulative gains, and the combination gives us the best score, corresponding to our best entry on the leaderboard, condition (7). Each of these gains are statistically significant, based on the t -test (at $p < 0.01$), on the development set (the only split we have access to). In other words, every component in our multi-stage ranking pipeline contributes significantly to end-to-end effectiveness.

To answer **RQ2**:

RQ2 Our ablation studies reveal that each component makes cumulative contributions to overall effectiveness. Document expansion (doc2query-T5) alone yields a nearly 50% relative improvement over the BM25 baseline on the test set. Adding pointwise reranking provides an additional significant boost. Finally, pairwise reranking contributes further gains, with all improvements being statistically significant at $p < 0.01$.

Condition (8) represents an ablation study that is intended to be contrasted with condition (4), designed to answer the question: What’s more important, the size of the sequence-to-sequence model, or the monoT5/duoT5 reranking design? The results are clear: a single-stage reranker with a larger model beats the pointwise/pairwise combination with a smaller model. However, as the comparison with condition (2) clearly shows, the addition of “Duo” improves over “Mono” alone, even with the smaller T5-base model. Finally, condition (9) shows that we need a “Duo” that is at least as large as “Mono” to be effective. That is, if “Duo” uses a smaller model than “Mono”, it does not appear to improve over “Mono” alone, see condition (6) vs. condition (9).

In [Figure 3.4](#), we examine the effectiveness of duoT5-3B on the development set when reranking different numbers of candidates k_1 from the output of monoT5-3B. In these experiments, we considered $k_1 \in \{0, 10, 20, 30, 40, 50\}$, where $k_1 = 0$ corresponds to using only monoT5-3B; in all cases, k_0 is set to 1000, i.e., monoT5 considers the top 1000 candidates from first-stage retrieval. The values in the x -axis are computed as $k_1 \times (k_1 - 1)$, which is the total number of inferences needed at the duo stage. We find that SYM-SUM and SYM-SUM-LOG performs best, indicating that combining scores from both (d_i, d_j) and (d_j, d_i) improves reranking. Interestingly, most of the impact of pairwise reranking comes from a small value of k_1 , which means that the gains in MRR come from reshuffling results in the top ranks. In fact, we see that in all but SYM-SUM, model effectiveness drops slightly as we increase k_1 . Note that our leaderboard submissions used $k_1 = 50$ with SYM-SUM.

In summary, addressing **RQ3**:

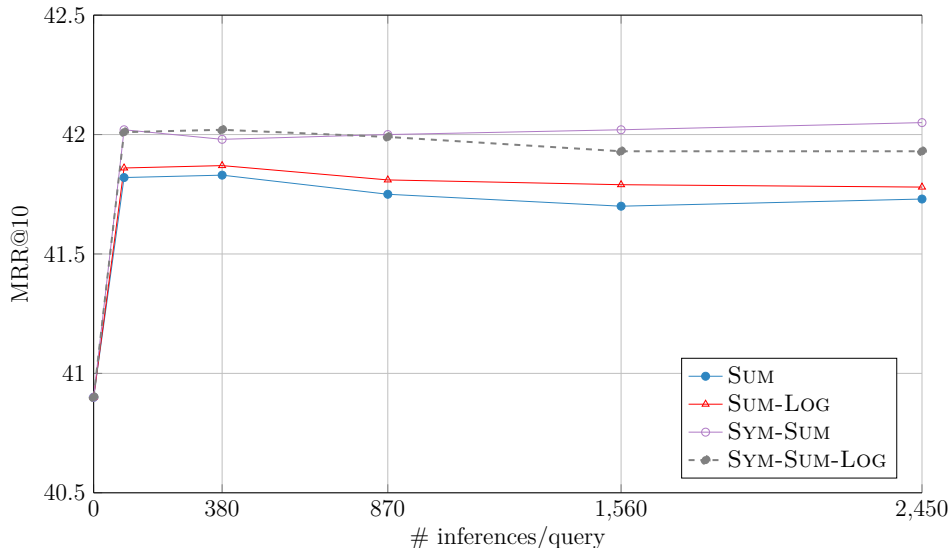


Figure 3.4: Number of inferences per query vs. the effectiveness of duoT5-3B on the MS MARCO passage development set when varying the number of candidates k_1 . We evaluated at $k_1 \in \{0, 10, 20, 30, 40, 50\}$, where $k_1 = 0$ corresponds to using only monoT5-3B. The values in the x -axis are computed as $k_1 \times (k_1 - 1)$.

RQ3 The complete Expando-Mono-Duo pattern achieved state-of-the-art effectiveness at the time of submission (May 2020), reaching the top position on the highly competitive MS MARCO passage ranking leaderboard with an MRR@10 of 0.408 on the test set. This surpassed strong baselines, including Google’s TFR-BERT Ensemble (0.395 MRR@10), demonstrating that our design pattern with T5-3B models can match or exceed ensemble-based approaches.

3.3.2 TREC 2020 Deep Learning Track Passage Ranking

Table 3.2 presents results from the passage ranking condition of the TREC 2020 Deep Learning Track; the columns H_{-1} , H_0 , H_1 , H_2 denote different settings of our Expando-Mono-Duo design. All of these results represent official submissions to the evaluation. It is worthwhile to emphasize that the underlying models did *not* take advantage of relevance judgments from the TREC 2019 Deep Learning Track. Our best run in terms of nDCG@10, condition (5), was the second-best run (on a per-team basis) submitted to the evaluation. The first block of the table, conditions (1)–(3), presents results without doc-

	H_{-1}	Expando-Mono-Duo Variants			MAP	nDCG@10	nDCG@1K	MRR	R@1K
		H_0	H_1	H_2					
(1)	-	BM25	-	-	0.2856	0.4796	0.5830	0.6585	0.7863
(2)	-	BM25 + RM3	-	-	0.3019	0.4821	0.6046	0.6360	0.8217
(3)	-	BM25 + RM3	monoT5-3B	duoT5-3B	0.5355	0.7583	0.7387	0.8759	0.8217
(4)	doc2query-T5	BM25	-	-	0.4074	0.6187	0.6840	0.7326	0.8452
(5)	doc2query-T5	BM25	monoT5-3B	duoT5-3B	0.5609	0.7837	0.7539	0.8798	0.8452
(6)	doc2query-T5	BM25 + RM3	-	-	0.4295	0.6172	0.7041	0.7424	0.8699
(7)	doc2query-T5	BM25 + RM3	monoT5-3B	duoT5-3B	0.5643	0.7821	0.7732	0.8798	0.8699

Table 3.2: Results of Expando-Mono-Duo Variants on the TREC 2020 Deep Learning Track passage ranking task.

ument expansion, and the second block of the table, conditions (4)–(7), presents results with doc2query-T5.

Let us first examine the impact of query expansion using pseudo-relevance feedback. The first two rows, conditions (1) and (2), represent standard bag-of-words baselines, with BM25 and BM25 + RM3, respectively (see [Section 3.2.1](#)). As expected, pseudo-relevance feedback (BM25 + RM3) increases effectiveness, which is consistent with decades of information retrieval research. Note that confirmation of this finding is possible only with the “dense” judgments available in TREC. As demonstrated by Lin et al. [120], with the “sparse” judgments from the MS MARCO passage dataset, BM25 + RM3 actually scores lower, due to the inability of the MS MARCO passage judgments (on average, only one relevant passage per query) to properly capture (i.e., reward) the effects of query expansion. We similarly observe the beneficial effects of RM3 when applied on top of the doc2query-T5 expanded index, condition (4) vs. condition (6), in terms of MAP, MRR, and R@1K, but not in terms of nDCG@10.

Conditions (2) and (3), (4) and (5), and (6) and (7) form contrastive minimal pairs with and without mono/duoT5 reranking using T5-3B. Note that these pairs have the same recall, since for each pair, the latter reranks the output from the former and thus cannot find any additional relevant documents. In each case, we see that our mono/duoT5 reranking pipeline yields a large increase in effectiveness in all first-stage configurations: BM25 + RM3, BM25 with doc2query-T5 expansion, and BM25 + RM3 with doc2query-T5 expansion. These findings are consistent with the results on the MS MARCO passage dataset. Conditions (5) and (7) demonstrate that while pseudo-relevance feedback improves recall, it is not clear if it improves end-to-end effectiveness in terms of metrics like MAP, nDCG@10, and MRR after reranking. In other words, in a pipeline with doc2query-T5, monoT5, and duoT5, it is unclear if pseudo-relevance feedback is still necessary.

Here, we see an interesting relationship between document expansion and pseudo-relevance feedback (basically query expansion) from the end-to-end perspective (with reranking). Put differently, the comparison between conditions (5) and (7) suggests that with document expansion, query expansion does not seem to matter much. In fact, our highest nDCG@10 score, condition (5), and also the best submission to the TREC 2020 Deep Learning Track, did *not* use query expansion. However, the comparison between conditions (3) and (7) suggests that with query expansion, document expansion still helps, i.e., the latter beats the former in all metrics. Furthermore, between the two techniques separately, document expansion alone, condition (5), is more effective than query expansion alone, condition (3). Since BM25 + RM3 is only a single query expansion technique (and does not exploit neural networks), this is not an entirely fair comparison between document expansion and query expansion methods *in general*. However, we do note that document expansion techniques can take advantage of longer texts as input (hence, more context) than query expansion techniques, since queries are usually much shorter.

3.3.3 MS MARCO Document Ranking

For the MS MARCO document ranking task, [Table 3.3](#) reports MRR@100, the official metric, for certain combinations of our techniques, as shown in the H_{-1} , H_0 , H_1 , H_2 columns: these provide ablations to show individual component contributions. As with the passage ranking task, obtaining a score on the test set requires submitting a run to the leaderboard. Hence, we focused on more interesting experimental conditions and refer readers to the more thorough ablation analysis on the MS MARCO passage ranking dataset in [Section 3.3.1](#).

Note that in condition (1), the only one that doesn’t use doc2query-T5, we used the per-passage index, while in the rest of the conditions, we used the per-passage expansion index, both as described in [Section 3.2.2](#). For the reranking conditions, conditions (3) and (4), the top 10000 passages from keyword search were processed (compared to the top 1000 for the passage ranking task). Thus, these runs required a significantly larger (around 10 times larger) compute budget compared to the MS MARCO passage ranking runs.

At the time of submission (September 2020), the full Expando-Mono-Duo configuration, condition (4), was the best result atop the very competitive leaderboard. The best run on the leaderboard (as of January 2021) is shown as condition (5) in [Table 3.3](#). While it is no longer the best-known result, Expando-Mono-Duo remains near the state of the art in terms of effectiveness on this task, especially given that our pipeline is zero-shot. In contrast, most of the top submissions involve ensembles and training directly on the MS

Model					Dev	Test
Expando-Mono-Duo Variants						
	H_{-1}	H_0	H_1	H_2		
(1)	-	BM25	-	-	0.268	-
(2)	doc2query-T5	BM25	-	-	0.318	0.284
(3)	doc2query-T5	BM25	monoT5-3B	-	0.411	0.362
(4)	doc2query-T5	BM25	monoT5-3B	duoT5-3B	0.426	0.370
(5)	PROP_step400K base (ensemble v0.1)				0.455	0.401

Table 3.3: Results of Expando-Mono-Duo Variants on the MS MARCO document ranking task, showing the official metric MRR@100 on the development and test sets. Note that scores on the test set are only available via a leaderboard submission.

MARCO document ranking dataset. As with the passage ranking leaderboard, we have not worked on this task since our submission, and it is possible that recent innovations can be incorporated into our approach to further improve the effectiveness of Expando-Mono-Duo.

We see that adding document expansion (stage H_{-1}) to enhance first-stage retrieval (stage H_0), condition (2) vs. (1), yields a boost on the development set. This is consistent with results from the passage ranking task. Note here that, as with the passage ranking task, this gain comes at only a modest increase in first-stage retrieval latency since the expanded documents are longer; computationally expensive neural inference is not required at query time.

On top of doc2query-T5, adding monoT5-3B, condition (3), and duoT5-3B, condition (4), each contributes additional cumulative gains, and the combination gives us the best score, corresponding to our best entry on the leaderboard, condition (4). Each of these gains are statistically significant, based on the t -test (at $p < 0.01$), on the development set (note that we do not have access to the test set). In other words, every component in our multi-stage ranking pipeline contributes significantly to end-to-end effectiveness.

To summarize our initial answer to **RQ4**:

RQ4 The Expando-Mono-Duo pattern demonstrates strong zero-shot effectiveness when applied to document ranking. Using only models trained on MS MARCO passage data (without any fine-tuning on document ranking data), our approach achieved the top position on the MS MARCO document ranking leaderboard at submission time (September 2020) with an MRR@100 of 0.370 on the test set. This zero-shot transfer

	H_{-1}	Expando-Mono-Duo Variants			MAP	nDCG@10	NDCG@1K	MRR	R@1K
		H_0	H_1	H_2					
(1)	-	BM25	-	-	0.3791	0.5271	0.5647	0.8521	0.8085
(2)	-	BM25 + RM3	-	-	0.4006	0.5248	0.5726	0.8541	0.8260
(3)	-	BM25 + RM3	monoT5-3B	duoT5-3B	0.5270	0.6794	0.6929	0.9476	0.8260
(4)	doc2query-T5	BM25	-	-	0.4230	0.5885	0.6115	0.9369	0.8403
(5)	doc2query-T5	BM25	monoT5-3B	duoT5-3B	0.5422	0.6934	0.7089	0.9476	0.8403
(6)	doc2query-T5	BM25 + RM3	-	-	0.4228	0.5407	0.5902	0.8147	0.8596
(7)	doc2query-T5	BM25 + RM3	monoT5-3B	duoT5-3B	0.5427	0.6900	0.7122	0.9476	0.8596

Table 3.4: Results of Expando-Mono-Duo Variants on the TREC 2020 Deep Learning Track document ranking task.

is particularly notable given that most competitive submissions employed ensembles and direct training on the target document collection.

3.3.4 TREC 2020 Deep Learning Track Document Ranking

Table 3.4 presents results from the document ranking condition of the TREC 2020 Deep Learning Track; the columns H_{-1} , H_0 , H_1 , H_2 denote different settings of our Expando-Mono-Duo setting. All of these results represent official submissions for evaluation. It is worthwhile to emphasize that the underlying models did *not* take advantage of relevance judgments from the TREC 2019 Deep Learning Track. Our best run in terms of nDCG@10, condition (5), was the best run submitted to TREC 2020. The first block of the table, conditions (1)–(3), presents results without document expansion. Due to limits on the number of submissions allowed, our runs for conditions (4)–(7) only used per-document expansions.

The first two rows represent standard bag-of-words baselines, with BM25 and BM25 + RM3, respectively. This parallels the conditions for the passage ranking task. Pseudo-relevance feedback increases effectiveness in terms of R@1K, the most crucial metric for first-stage retrieval since it sets the effectiveness upper bound for the entire reranking pipeline. We see that MAP increases as well, but nDCG@10 and MRR are essentially unchanged (since they are not recall-oriented metrics). The finding here is consistent with results from the passage ranking task. The comparison between condition (4) and condition (6) illustrates the effects of applying pseudo-relevance feedback on top of doc2query-T5 (without reranking). We see that R@1K improves, but the other metrics are either essentially unchanged or actually degrade. Thus, the gains from doc2query-T5 and RM3 do *not*

appear to be additive.

Note that conditions (2) and (3), (4) and (5), and (6) and (7) have the same recall, since for each pair, the latter reranks output from the former and thus cannot find any additional relevant documents. However, for each of the pairs, the mono/duo reranking pipeline substantially increases effectiveness. These findings are consistent with previous results.

As with the passage ranking case, the comparisons between conditions (3), (5), and (7) illustrate the effects of document expansion vs. query expansion (pseudo-relevance feedback with RM3) in an end-to-end setting (with full mono/duoT5 reranking). Comparing conditions (5) and (7), we see that with document expansion, query expansion doesn’t add much. However, starting with query expansion, document expansion improves effectiveness; this is seen by comparing condition (3) and condition (7). Individually, document expansion (alone) appears to be more effective than query expansion (alone), based on comparing condition (3) and condition (5). These findings are consistent with results in [Section 3.3.2](#) on passage ranking.

3.3.5 TREC-COVID

Official results from TREC-COVID are shown in [Table 3.5](#). The official evaluation metric was nDCG, at a rank cutoff of 10 for the first three rounds, increased to 20 for rounds 4 and 5. NIST also reported a few other metrics, including precision at a fixed ranked cutoff and average precision (AP) to the standard rank depth of 1000. It is worth emphasizing that due to the residual collection evaluation methodology, and the fact that the corpora and topics were different, scores across rounds *are not* comparable.

A thorough discussion of all results from the TREC-COVID challenge is beyond the scope of this report, and thus, we focus only on runs that directly pertain to the Expando-Mono-Duo pattern. [Table 3.5](#) shows only “automatic” runs that were formally submitted to the evaluation. In an automatic run, manual intervention was not allowed. This contrasted with two other categories of runs that were accepted: “feedback” runs, which could leverage relevance judgments from previous rounds but otherwise could not involve human intervention, and “manual” runs, where any human intervention was allowed.

The “Run” column of [Table 3.5](#) shows the run identifier for readers interested in matching our submissions with official results, and the “Description” column provides a short description in the Expando-Mono-Duo context. In rounds 1, 2, and 3, we present our own runs as well as the best automatic run from that round—these are noted as conditions (1a), (2a), and (3a), respectively. In rounds 2 and 3, we submitted the second-best runs (on a

	Team	Run	Description	Type	nDCG@10	P@5	MAP	Judged@5
Round 1: 30 topics								
(1a)	sabir	sabir.meta.docs	monoT5-3B	automatic	0.6080	0.7800	0.3128	1.0000
(1b)	covidex	T5R1		automatic	0.5223	0.6467	0.1919	1.0000
Round 2: 35 topics								
(2a)	GUIR_S2	GUIR_S2_run1	monoT5-3B	automatic	0.6251	0.7486	0.2842	0.9885
(2b)	covidex	covidex.t5		automatic	0.6250	0.7314	0.2880	1.0000
(2c)	anserini	r2.fusion2		automatic	0.5553	0.6800	0.2725	1.0000
(2d)	anserini	r2.fusion1		automatic	0.4827	0.6114	0.2418	1.0000
Round 3: 40 topics								
(3a)	SFDC	SFDC-fus12-enc23-tf3	monoT5-3B + duoT5-3B	automatic	0.6867	0.7800	0.3160	0.9850
(3b)	covidex	r3.duot5		automatic	0.6626	0.7700	0.2676	0.8850
(3c)	covidex	r3.monot5	monoT5-3B	automatic	0.6596	0.7800	0.2635	0.9300
(3d)	anserini	r3.fusion2		automatic	0.6100	0.7150	0.2641	0.9550
(3e)	anserini	r3.fusion1		automatic	0.5359	0.6100	0.2293	0.8950
	Team	Run	Description	Type	nDCG@20	P@20	AP	Judged@5
Round 4: 45 topics								
(4a)	covidex	covidex.r4.d2q.duot5	doc2query-T5 + monoT5-3B + duoT5-3B	automatic	0.7219	0.7267	0.3122	
(4b)	covidex	covidex.r4.duot5	monoT5-3B + duoT5-3B	automatic	0.6877	0.6922	0.3283	
(4c)	uogTr	uogTrDPH_QE_SCB1		automatic	0.6820	0.7144	0.3457	
(4d)	anserini	r4.fusion2		automatic	0.6089	0.6589	0.3088	
(4e)	anserini	r4.fusion1		automatic	0.5244	0.5611	0.2666	
Round 5: 50 topics								
(5a)	covidex	covidex.r5.d2q.2s	doc2query-T5 + monoT5-3B + duoT5-3B	automatic	0.7539	0.7700	0.3227	
(5b)	covidex	covidex.r5.2s	monoT5-3B + duoT5-3B	automatic	0.7457	0.7610	0.3212	
(5c)	uogTr	uogTrDPH_QE_SB_CB		automatic	0.7427	0.7910	0.3305	
(5d)	covidex	covidex.r5.d2q.1s	doc2query-T5 + monoT5-3B	automatic	0.7121	0.7320	0.3150	
(5e)	covidex	covidex.r5.1s	monoT5-3B	automatic	0.6960	0.7070	0.3119	
(5f)	anserini	r5.fusion2		automatic	0.6007	0.6440	0.2734	
(5g)	anserini	r5.fusion1		automatic	0.5313	0.5840	0.2314	

Table 3.5: Selected TREC-COVID results. Rows show combinations of Expando-Mono-Duo as well as baselines and submissions from other teams for comparison. Note that the metrics used in the first three rounds are different from those used in the final two rounds.

per-team basis). In rounds 4 and 5, we submitted the best automatic run; for comparison, we show the second-best runs, marked conditions (4c) and (5c), respectively.

Within each round, our submissions provided limited contrastive and ablation runs that highlight different aspects of the Expando-Mono-Duo pattern. Since each team was only allowed to submit a limited number of runs (three in rounds 1 through 4 and eight in round 5), it was not possible to examine all interesting conditions. Also, due to the rapid pace of the evaluation, we were only able to “roll out” components in our Expando-Mono-Duo design pattern incrementally. We discuss results from each round below:

Round 1. Our monoT5-3B reranker was introduced in round 1, shown as condition (1b), although its effectiveness was still quite far behind the state of the art. The best submission in round 1 was a fusion-based run using the vector space model, involving no

neural techniques at all. A post-hoc analysis diagnosed the issue with our run to be with first-stage retrieval: we had used the paragraph index, but simply replacing our pipeline with the abstract index improved nDCG@10 from 0.5223 to 0.5702. It is unclear whether this difference was due to an evaluation artifact.

Round 2. Starting in round 2 for the remaining rounds, the `fusion1` and `fusion2` baselines described in [Section 3.2.3](#) were made available to the community to serve as first-stage retrieval. We took advantage of these strong keyword baselines as the input to our reranking pipeline. In condition (2b), we reranked `fusion1` and `fusion2` with `monoT5-3B`, and then combined the reranked results using reciprocal rank fusion [40]. This yielded effectiveness that was, for all practical purposes, the state of the art (behind the top-scoring run by only 0.001).

Round 3. As the rounds progressed in TREC-COVID, we introduced more components in our Expando-Mono-Duo design. Our `duoT5-3B` reranker was introduced in round 3, shown as condition (3b): similar to the round 2 configuration, we reranked `fusion1` and `fusion2` with the two-stage `monoT5/duoT5` pipeline. Then we combined the reranked results using reciprocal rank fusion. Condition (3c) represents the same technique as condition (2b), and thus conditions (3b) and (3c) isolate the effects of the `duoT5` reranker. We see that `duoT5` contributes to small increases in nDCG@10 and average precision, but a slight decrease in precision at rank 5.

The highest-scoring automatic run submitted in round 3 was `SFDC-fus12-enc23-tf3`, which is 2.4 points higher than our best run; condition (3b) represented the second-highest automatic run. Thus, it would be fair to characterize the `monoT5/duoT5` pipeline as close to the state of the art for this round.

Round 4. Our `doc2query-T5` document expansion technique was introduced in round 3, thereby completing our Expando-Mono-Duo design pattern. Row (4b) represents the same technique as row (3b), and thus conditions (4a) and (4b) isolate the effects of document expansion. We see that document expansion clearly contributes to a big increase in terms of nDCG@20 and precision at rank 20, although, interestingly, average precision decreased.

Round 5. Since we were allowed eight submissions in this final round, we submitted two more ablations on our design pattern. Row (5a), (5b), and (5e) represent the same techniques as row (4a), (4b), and (3c), respectively. Row (5d) added the new condition where we just ran the “Expando-Mono” part of our design pattern. Comparing conditions (5a) and (5d) vs. (5b) and (5e), we see the effects of document expansion in isolation (i.e., fixing the other parts of the pattern). Consistent with results from the other retrieval tasks, we see considerable improvements in effectiveness. Comparing (5a) and (5b) vs. (5b) and

(5e), we see that pairwise reranking with duoT5-3B contributes effectiveness gains over just reranking with monoT5. Again, this is consistent with results from other tasks.

In conclusion, to answer **RQ4** in a cross-domain setting:

RQ4 Beyond zero-shot document ranking, TREC-COVID demonstrates the robustness of our design pattern in cross-domain transfer. Using models trained only on MS MARCO passage data with minimal domain adaptation (fine-tuning on Med-MARCO, a subset of MS MARCO), the Expando-Mono-Duo pipeline achieved the best automatic run in TREC-COVID rounds 4 and 5, ahead of industry-leading large ensembles. Remarkably, doc2query-T5 generated high-quality expansions for scientific abstracts despite never being fine-tuned on in-domain data, showing that the pattern generalizes effectively to specialized domains.

3.4 Concluding Thoughts

At the micro level, this chapter has successfully demonstrated the viability of the Expando-Mono-Duo design pattern. We have shown how a single sequence-to-sequence architecture can be effectively adapted to serve three distinct roles within a ranking pipeline: document expansion, pointwise reranking, and pairwise reranking. Our ablation studies confirmed that each component—Expando, Mono, and Duo—provides a considerable cumulative boost to effectiveness. The final pipeline achieved results that were state-of-the-art on several competitive benchmarks at the time, answering our initial research questions affirmatively. In fact, this pipeline continues to serve as a strong baseline for the information retrieval community.

Zooming out to the macro picture, the Expando-Mono-Duo pattern represents a pinnacle of the supervised ranking paradigm. The paper has over 200 citations and continues to be used as a strong baseline in several TREC tracks and beyond [169, 106, 170, 173, 179]. It is a simple yet powerful and highly optimized recipe for building search systems. However, there were some limitations of this paradigm. The reliance on large-scale supervised datasets for training multiple components, coupled with the substantial computational cost of a multi-stage inference process (especially the quadratic complexity of the Duo stage), makes such systems expensive to develop, deploy, and maintain.

This macro-level view reveals a clear need for a paradigm shift towards more streamlined and less human-curated data-dependent approaches. The challenges inherent in the

Expando-Mono-Duo architecture motivate the work that follows. The next chapter explores how we can squeeze effectiveness out of a smaller BERT-based model using advancements in both the data and modeling dimensions. Chapter 6 confronts issues related to the need for supervised data by exploring the potential of LLMs to perform listwise reranking in a zero-shot setting, asking whether a single, powerful model distilled from a larger teacher model can simplify the ranking pipeline while matching the hard-won effectiveness of such supervised systems.

Chapter 4

Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking

In [Chapter 3](#), we improved overall system effectiveness with an elaborate multi-stage pipeline, but a better pointwise reranker is still always a win. While first-stage retrievers cast a wide net and focus on recall, it is the cross-encoders that provide the final, high-precision ordering. Despite being a well-established component since its introduction in 2019 by Nogueira and Cho [\[154\]](#), innovation in supervised cross-encoders, specifically pointwise rerankers, had slowed, with many researchers treating them as a largely “solved” problem. This chapter challenges that assumption, arguing that significant, untapped potential remains within these foundational components.

Here, we present a systematic investigation into the optimization of pointwise rerankers by combining different model backbones, loss functions, and negative mining strategies. We revisit the basic cross-encoder design and demonstrate that a series of careful and methodical improvements can yield substantial gains in pointwise reranking effectiveness. This work, conducted in 2021, was guided by the following research questions:

- (RQ1) Can combining alternative model weights with more effective first-stage retrievers for negative mining significantly improve the effectiveness of pointwise rerankers?

- (RQ2) How do different loss functions—specifically Hinge, Cross-Entropy, and the Localized Contrastive Estimation (LCE) loss from Gao et al. [67]—compare when training these models?
- (RQ3) How critical is the alignment between the retriever used for hard-negative mining during training and the retriever used in evaluation?
- (RQ4) What is the impact of the number of negative samples on ranking effectiveness when using the LCE loss?

Through a comprehensive study, we show that even a well-established encoder-only model can be pushed to state-of-the-art effectiveness, a model we dub monoELECTRA. This work underscores the value of continuous refinement of core system components and provides a highly optimized, powerful building block for retrieval pipelines.

4.1 Methodology

First, we review standard loss functions (cross-entropy and hinge loss) used in cross-encoder fine-tuning (for encoder-only models) and then describe the LCE loss function proposed by Gao et al. [67].

We begin with a quick review of how encoder-only pointwise models typically compute the relevance score, given a query q and a document d , using the formulation from Lin et al. [120]:

$$z_{q,d} = T_{[CLS]}W + b \quad (4.1)$$

where $z_{q,d}$ is the relevance score of the query–document pair, $T_{[CLS]}$ stands for the representation of the final layer’s [CLS] token, and W and b are the final classification layer’s weight and the bias. The W and b dimensions vary according to the loss function — when the model is fine-tuned with the cross-entropy loss, $W \in \mathbb{R}^{D \times 2}$ and $b \in \mathbb{R}^2$, whereas $W \in \mathbb{R}^D$ and $b \in \mathbb{R}$ with hinge or LCE loss. That is, the output has two dimensions when we use the cross-entropy loss, one each for the relevant and non-relevant classes, while with the other two losses, the output has only one dimension, for the relevant class only.

4.1.1 Cross-Entropy and Hinge Loss

Early cross-encoders fine-tune BERT using the cross-entropy loss, as recommended by Devlin et al. [50]:

$$s_{q,d} = \text{softmax}(z_{q,d})_1 \quad (4.2)$$

$$L_{CE} = -\sum \log(s_{q,d^+}) - \sum \log(1 - s_{q,d^-}) \quad (4.3)$$

where $\text{softmax}(\cdot)_1$ corresponds to the softmax score of the relevant label, d^+ indicates a relevant document, and d^- indicates a non-relevant document. We will leverage this notation from now on.

Later, MacAvaney et al. [136] fine-tuned cross-encoders with the hinge loss, which is more commonly used in pre-BERT neural rerankers [240, 74, 86]:

$$L_{\text{hinge}} = \max(0, 1 - z_{q,d^+} + z_{q,d^-}) \quad (4.4)$$

In the literature, cross-entropy loss and hinge loss represent the two basic ways of training cross-encoders.

4.1.2 Localized Contrastive Estimation

Gao et al. [67] note that the above cross-entropy loss computation considers only one document per batch per query. While not discussed in the original paper, the hinge loss is limited by not being able to use multiple negatives per positive unless done in a pairwise independent fashion. They also note it is essential that negative examples be true negatives, especially on datasets like MS MARCO passage, where many relevant passages remain unlabelled for each query. Gao et al. [67] propose the Localized Contrastive Estimation (LCE) loss to address these issues:

$$L_{LCE_q} := -\log \frac{\exp(z_{q,d^+})}{\sum_{d \in G_q} \exp(z_{q,d})} \quad (4.5)$$

$$L_{LCE} := \frac{1}{|Q|} \sum_{q \in Q, G_q \sim R_q^m} L_{LCE_q} \quad (4.6)$$

where R_q^m is the list of documents top-ranked by some first-stage retriever for query q , and G_q refers to a group of documents for query q , which consists of a relevant document d^+ and $n - 1$ non-relevant documents d^- sampled from R_q^m , where n is the group size.

The LCE loss augments the Noise Contrastive Estimation (NCE) [75] loss (used in, for example, Karpukhin et al. [95]) with “localized” selection of negative examples. In the NCE loss, the model scores the positive instance and multiple negative instances, normalizes all of them into probabilities by passing them through the softmax function, and then encourages the model to score the positive instance higher than the negatives. LCE “localizes” this loss by sampling negative examples from the top-ranked documents

produced by the first-stage retriever. In combination, this loss should produce a reranker that succeeds at handling the top-ranked documents specific to a first-stage retriever while also not collapsing to match based on the confounding characteristics in the retriever’s hard negative samples.

4.1.3 Experimental Setup

In this section, we describe the data and experimental configurations used in our replication. Note that *replication* indicates using a different experimental setup to generalize findings from the original paper, whereas *reproduction* indicates verifying the original paper’s findings using the same experimental setup.¹

4.1.4 Data

We use the MS MARCO *passage* ranking dataset [13] (MS MARCO for later reference), a large-scale *ad hoc* retrieval dataset constructed from the Bing search log. It contains 8.8 million passages with around 800K queries for training, where most of the queries have a single passage labelled relevant. These do not necessarily represent *all* true relevant passages, as it is likely that several queries in the dataset have more than one relevant passage, the sparse setting discussed in the previous chapter. On the evaluation side, there is a small development set with 6980 queries and a blind test set with 6837 queries, both of which are similarly sparsely labelled.

We report MRR@10, the official metric, and Recall@1K (R@1K) on the small development set for all our experiments. Evaluating on the blind test set requires submitting the runs to the organizers’ official leaderboard. To avoid probing test sets across all settings considered, we chose to submit only the test set run produced by the most effective system based on development set scores.

Note that Gao et al. [67] use the MS MARCO *document* ranking dataset. Thus, our experiments generalize their findings to cover the MS MARCO *passage* ranking dataset and additionally thoroughly explore certain parts in the hyperparameter space.

First-stage rankings (runs) are generated for MS MARCO’s training, development, and test query sets with two retrievers: BM25 and TCT-ColBERTv2 [123], a state-of-the-art

¹The terms replication and reproduction are used in the sense articulated by the [ACM Artifact Review and Badging \(v1.1\) policy](#); note that the definitions of the two terms are swapped in [Artifact Review and Badging \(v1.0\)](#).

dense retriever in 2021. We use the Anserini IR toolkit [245] to generate the BM25 runs. The parameters k_1 and b were found with grid search over the ranges $[0.6, 1.2]$ and $[0.5, 0.9]$, respectively, both using step size 0.1. The tuning is based on five different randomly prepared query subsets, optimizing Recall@1K, following the reproduction documentation in Anserini.² We use the Pyserini IR toolkit [119] to generate the TCT-ColBERTv2 runs following the reproduction documentation in Pyserini.³ We use the model trained with the HN+ setting as it optimizes the effectiveness of the primary metrics.

We first retrieve the top-200 passages for the training queries, from which we randomly sample negative examples without replacement following Gao et al. [67]. This method of curating the training set differs from the then-standard approach of training cross-encoders, as we saw in Chapter 3, of instead using the official small triples training file provided by the organizers.

We retrieve 1K passages for all queries in the development set and test set. They form the base first-stage retriever runs, which are later reranked by the cross-encoders.

4.1.5 Training and Inference

Our cross-encoder training and inference experiments are run on Capreolus [251, 252], a toolkit for end-to-end neural ad hoc retrieval. We take advantage of its support for the MS MARCO passage, the monoBERT cross-encoder, and the training, reranking, and evaluation pipeline. We use the provided hinge and cross-entropy loss functions and incorporate the LCE loss into the toolkit. Since we released monoELECTRA, we’ve also added inference support for the models to RankLLM [200].

The maximum number of tokens for the query and the entire input sequence (“[CLS] *query* [SEP] *passage* [SEP]”) are set to 50 and 256, respectively. For all experiments, we initialize monoBERT with ELECTRA_{Base}, using the checkpoint released on HuggingFace [238].⁴ We choose ELECTRA_{Base} as the starting point for fine-tuning as it was found to be a stable and effective pLM by Zhang et al. [257]. ELECTRA_{Base} uses the same Transformer architecture as BERT-Base, but is pre-trained with a replaced token detection objective rather than masked language modeling [37].

We train all monoELECTRA variants for 300K steps using a batch size of 16. We use the Adam optimizer [99] with a learning rate of $1e-5$, apply linear warm-up for 30K steps,

²<https://github.com/castorini/anserini/blob/master/docs/experiments-msmarco-passage.md>

³https://github.com/castorini/pyserini/blob/master/docs/experiments-tct_colbert-v2.md

⁴[google/electra-base-discriminator](https://github.com/google/electra-base-discriminator)

	HN+ First Stage	Loss	n	MRR@10	R@1K
Baselines					
(a) BM25		–	–	0.187	0.857
(b) TCT-ColBERTv2		–	–	0.359	0.969
Prior cross-encoder work					
(c) monoBERT _{Base} [154]	BM25	CE	1	0.347	–
(d) monoBERT _{Base} [156]	BM25	CE	1	0.348	–
(e) monoBERT _{Base} [64]	BM25	CE	1	0.353	–
(f) monoBERT _{Base} [82]	BM25	CE	1	0.376	–
(g) monoBERT _{Base} [154]	BM25	CE	1	0.365	–
(h) monoBERT _{Large} [82]	BM25	CE	1	0.366	–
(i) monoBERT _{Large} [158]	BM25	CE	1	0.372	–
(j) monoT5 _{Base} [158]	BM25	CE	1	0.381	–
Capreolus cross-encoders					
(1) monoELECTRA _{Base}	BM25	CE	1	0.378 ^{ab}	0.857
(2) monoELECTRA _{Base}		Hinge	2	0.379 ^{ab5}	
(3) monoELECTRA _{Base}		LCE	2	0.378 ^{ab5}	
(4) monoELECTRA _{Base}		LCE	8	0.391 ^{ab12356}	
(5) monoELECTRA _{Base}	TCT-ColBERTv2	CE	1	0.365 ^a	0.969
(6) monoELECTRA _{Base}		Hinge	2	0.375 ^{ab}	
(7) monoELECTRA _{Base}		LCE	2	0.393 ^{ab12356}	
(8) monoELECTRA _{Base}		LCE	8	0.401 ^{ab123456}	

Table 4.1: MRR@10 and Recall@1K with different loss functions when using BM25/TCT-ColBERTv2 as the source of hard negative and first-stage runfile. The n in the table indicates the group size. For hinge and LCE, each group always contains a positive example and $n - 1$ negative examples. For CE, each group only contains one data point, which could be either a positive or negative example. “–” indicates not applicable or the score was not reported in the original papers. Superscripts indicate significantly higher results ($p < 0.01$ with paired t -tests) after Bonferroni correction, e.g., ^a indicates the entry is significantly higher than the results in row (a).

following it up with a linear decay. All experiments are run on Quadro RTX 8000 GPUs with TensorFlow 2.3.0. We use mixed-precision training in all the experiments.

4.2 Results and Discussion

In this section, we first compare the results of the three loss functions (hinge, cross-entropy, and LCE) when using BM25 and TCT-ColBERTv2 first-stage retrievers. Here, we aim to show that the cross-encoders trained with the LCE loss outperform those trained with the other two losses on the MS MARCO passage. Next, we compare the effectiveness as

we vary both the source of negatives during training and the first-stage retriever during evaluation. To conclude, we show that the effectiveness of cross-encoders trained with the LCE loss increases with group size, which also means it increases with more negative samples.

4.2.1 Loss Functions

[Table 4.1](#) reports results with different loss functions and first-stage retrievers used during training and evaluation. In the first block, we report the scores of BM25 and TCT-ColBERTv2, which form the two baseline first-stage retrieval runs we consider for all the Capreolus rerankers.

The second block reports the scores of various comparable cross-encoders from various groups reported in the literature. We copy over the monoBERT and monoT5 scores from the original papers, rows (c), (g), and (j), respectively. In addition, we include other monoBERT results reported by different groups because we observe a considerable variance in reported scores. This could be due to one of many reasons: different BM25 implementations, monoBERT training hyperparameters, and the number of passages reranked, to name a few.

The third block of the table, rows (1-4), shows the scores when our pointwise reranker is trained on BM25-sourced hard negatives and reranks the BM25 runfile. The fourth block, rows (5-8), shows the scores when the cross-encoder is trained on TCT-ColBERTv2-sourced hard negatives and reranks the TCT-ColBERTv2 runfile.

Gao et al. [67] only compare the cross-entropy loss to the LCE loss with a fixed group size of 8. We generalize this by additionally considering a group size of 2 (and more later) with LCE loss and including hinge loss, which can also be viewed as having a group size of 2 but has a different loss formulation. As negative examples are sampled from the same groups of top-ranked passages, both losses benefit from the “localized” effect, and the formulation is the key difference.

Cross-entropy loss scores on par with the hinge loss when BM25 is used as the retriever, row (1) vs. (2). However, when using TCT-ColBERTv2 as the retriever, the hinge loss demonstrates improved effectiveness over the cross-entropy loss by a slight margin, row (5) vs. (6). We suspect this is due to the pairwise loss making better use of the harder negative examples provided by TCT-ColBERTv2.

We also note that monoELECTRA with LCE significantly outperforms that with the other two losses when TCT-ColBERTv2 forms the first-stage retriever, even when the group

	HN	First-stage	MRR@10	R@1K
(a)	BM25	BM25	0.391	0.8573
(b)	TCT-ColBERTv2		0.389	
(c)	BM25	TCT-ColBERTv2	0.402 ^{ab}	0.9690
(d)	TCT-ColBERTv2		0.401 ^{ab}	

Table 4.2: MRR@10 and Recall@1K of all combinations of training hard negatives retriever and inference first-stage retriever on the development set of the MS MARCO passage dataset. HN refers to the source of Hard Negatives, i.e., the training retriever. All table entries use LCE with the group size 8 (one positive sample with seven negative samples). Superscripts indicate significantly higher results ($p < 0.01$ with paired t -tests) after Bonferroni correction.

size is 2, which hinge loss uses too, row (5-7). However, LCE and hinge losses perform comparably when using BM25 as the retriever, row (2) vs. (3), and fixing the group size at 2. This indicates that the contrastive loss may itself serve as a better approach to distinguish the relevant passage from the negative ones in the ranking task, compared to the hinge loss. It additionally gains from increasing the group size, rows (3) and (7) vs. rows (4) and (8); this is more carefully examined in [Section 4.2.3](#).

To answer **RQ1** and **RQ2**:

RQ1 We find that combining ELECTRA_{Base} with the stronger TCT-ColBERTv2 retriever for hard negative mining significantly improves effectiveness compared to using BERT_{Base} with BM25. This demonstrates that more effective models with better first-stage retrievers to provide higher-quality negative examples result in better pointwise models.

RQ2 We find that the LCE loss outperforms both hinge loss and cross-entropy loss, particularly when paired with stronger retrievers like TCT-ColBERTv2. The LCE formulation provides a better learning signal for distinguishing relevant passages from hard negatives in the ranking task.

4.2.2 Hard Negatives and Retrieval Alignment

[Table 4.2](#) presents the effectiveness of the reranker when we vary the retriever for preparing training negatives and generating the development runfile. Rows (a) and (d) here

correspond to rows (4) and (8) in Table 4.1, respectively. To obtain row (b), we use the checkpoint of row (d) to rerank the BM25 runfile. Similarly, we use the checkpoint of row (a) to rerank the TCT-ColBERTv2 runfile for row (c).

It is clear that switching out the BM25 first-stage retriever with the dense retriever, TCT-ColBERTv2, results in significant improvement irrespective of the retriever used to mine hard negatives, rows (a-b) vs. (c-d). This is reasonable as there is a difference of around 11% in R@1K, meaning reranking the runfile produced by TCT-ColBERTv2 would more likely pull up the relevant passages.

However, we observe no improvement in aligning the hard-negative mining retriever with the evaluation first-stage retriever. In our experiments, changing only the retriever for generating the training data does not yield noticeable differences in the score when we preserve the first-stage retriever to be the same, row (b) vs. (a) and row (c) vs. (d). This contrasts with findings on the MS MARCO document ranking task as seen by Gao et al. [67]. There are several differences in the experiments that could be responsible for this disagreement. The first is the dataset itself. Although both the MS MARCO passage and the MS MARCO document are from the same *ad hoc* domain, the document length may impact training data quality. Other possible causes include the range from which we sample the hard negatives, the choice of the first-stage retriever, etc. Based on these results, for the rest of the paper, we use TCT-ColBERTv2 as the first-stage retriever during inference.

To summarize, for RQ3:

RQ3 We find that alignment between the retriever used for hard-negative mining during training and the retriever used for candidate generation at inference is not critical for the MS MARCO passage ranking task. Cross-encoders trained on BM25-mined negatives perform comparably when reranking TCT-ColBERTv2 results versus those trained on TCT-ColBERTv2-mined negatives.

4.2.3 LCE Group Size

Here, we explore the effect of the group size in the LCE loss, which we denote by n , on effectiveness. This has been studied by Gao et al. [67] with $n \in \{2, 4, 6, 8\}$, where additional improvement can always be observed when the group size is increased. We investigate the effect of group size in the same manner, but increase the range to $\{2, 4, 8, 16, 32\}$.

Method	Dev MRR@10	Eval MRR@10
(a) coCondenser [62]	0.443	0.428
(b) C-COIL + RoBERTa [66]	0.443	0.427
(c) RocketQA + ERNIE [182]	0.439	0.426
(d) DR-BERT	0.420	0.419
(e) expando-mono-duo-T5 [172]	0.420	0.408
(f) DeepCT + TF-Ranking Ensemble [78]	0.420	0.408
(g) monoELECTRA	0.414	0.404

Table 4.3: MRR@10 on the official MS MARCO passage leaderboard.

HN	Group Size				
	2	4	8	16	32
(a) TCT-ColBERTv2	0.393 _{b₂}	0.400 _{b₂}	0.401 _{b₂}	0.408 _{b_{2,4}} ^{a_{2,4,8}}	0.414 _{b_{2,4,8,16}} ^{a_{2,4,8}}
(b) BM25	0.381	0.397 _{b₂}	0.402 _{b₂}	0.403 _{b₂}	0.407 _{b_{2,4}}

Table 4.4: MRR@10 on the development set of the MS MARCO passage dataset across the choice of group size and retriever used to mine hard negatives. All entries use TCT-ColBERTv2 as the first-stage, which has a Recall@1K of 0.9690, as seen in Table 4.1. Superscripts and subscripts indicate significantly higher results ($p < 0.01$ with paired t -tests) after Bonferroni correction. (e.g., $(\cdot)_{b_2}^{a_{2,4,8}}$ indicates the entry is significantly higher than the results in row (a) with group sizes 2, 4, and 8, and the result in row (b) with group size 2.)

Additionally, we vary the retriever used for mining the hard negatives. We look at the MRR@10 scores across these settings in Table 4.4.

As noted by Gao et al. [67], we observe that the primary metric improves as the number of negatives increases. We do so for both choices of the retriever used for hard negative mining. We surprisingly find that the metric does not seem to plateau even when the group size increases to 32. We did not experiment on larger group sizes due to hardware limitations.⁵

Table 4.2 does not note any improvements aligning the hard negative mining retriever with that used for first-stage retrieval during inference in the case with group size 8.

⁵The experiment involving a group size of 32 requires 4 Quadro RTX 8000 GPUs (48 GB memory each) to train with a batch size of 16.

However, we find improvements in group sizes of 2 and 32. We leave further investigation of this unusual observation as future work.

We submitted the test set run corresponding to our most effective development set configuration to the MS MARCO passage leaderboard.⁶ Table 4.3 reports our scores and the other submissions with higher scores on the test set (at the time of our work). The table shows that our best results are quite competitive to the top results from 2021, which use ensembles of several cross-encoders, rows (a-c,f), or our multi-stage reranking pipeline from Chapter 3, row (e).

To answer RQ4:

RQ4 We find that increasing the group size for both first-stage retrievers consistently improves effectiveness when using the LCE loss.

4.3 Concluding Thoughts

This chapter demonstrated that substantial effectiveness gains can still be squeezed from supervised pointwise rerankers. Through a systematic exploration of model backbones, loss functions, and negative mining strategies, we have shown that even components considered “solved” can be significantly improved. By applying the Localized Contrastive Estimation loss, leveraging stronger first-stage retrievers for hard negative mining, and increasing the number of negatives, we pushed a standard cross-encoder architecture to then state-of-the-art effectiveness.

This work provides the community with a powerful, highly optimized reranking component, reinforcing the idea that continuous, careful refinement of existing models is a vital avenue for progress. In fact, a number of papers through 2025 use monoELECTRA as a competitive baseline, with some even building on top of it. Parry et al. [166] find that monoELECTRA is competitive with LLM-based listwise approaches, which we will discuss in Chapter 6, when tasked to rerank randomly ordered passages from an in-domain collection. Note that the monoELECTRA models are over an order of magnitude smaller than the models that power today’s listwise rerankers, so this finding is substantial. Schlatt et al. [199] leverage RankZephyr reorderings and fine-tune a more effective monoELECTRA. As evident from its continued adoption in recent work, monoELECTRA has remained a strong and influential baseline that actively shapes how researchers design and evaluate modern reranking systems.

⁶<https://microsoft.github.io/MSMARCO-Passage-Ranking-Submissions/leaderboard>

While we have pushed the boundaries of supervised cross-encoders of this scale and for this task, this approach still fundamentally relies on large-scale, human-annotated training data, a limitation that was becoming increasingly apparent in 2021. The logical next step in reranking is to move beyond this requirement by leveraging the zero-shot capabilities of LLMs for listwise reranking. This shift, however, presents its own difficulties, particularly a potential reliance on proprietary, closed-source LLMs, which introduces a new facet of the Component Challenge. The work in Chapter 6 of this thesis confronts this issue by exploring the building of an open-source listwise reranker designed to match the effectiveness of its proprietary counterparts, with the goal of democratizing the next wave of information access systems.

While this chapter focused on refining an existing component, a parallel line of research proposed a more radical transformation of the entire retrieval pipeline. This new paradigm, known as generative retrieval, sought to replace the traditional index-and-retrieve architecture with a single model that learns to generate document identifiers directly from a query. This ambitious approach promised to unify the retrieval process within one model’s parameters, but its viability at a large scale remained a critical, unanswered question. Therefore, before we continue our exploration of advanced reranking techniques, the next chapter takes a necessary detour to evaluate the scalability and practical limits of this emerging retrieval paradigm.

Chapter 5

On the Scalability of Generative Retrieval

Dual encoders have long remained the most popular way to do first-stage retrieval. However, in 2022, generative retrieval emerged as a radical new paradigm, reframing retrieval as a sequence-to-sequence task [211]. Here, a single Transformer model learns to map queries directly to document identifiers, effectively encoding the entire corpus within its parameters.

While early results were promising, they were limited to small corpora of around 100,000 documents. This leaves a critical open question: could this paradigm scale to the millions of documents required for real-world applications?

This chapter addresses this question through the first large-scale empirical study of generative retrieval, systematically evaluating its effectiveness as we scale from 100K to 8.8M passages. We investigate the design choices and modeling techniques that are most critical for effectiveness at scale. The following research questions guide this work:

- (RQ1) What input and output representations are most effective for generative retrieval across scales of corpora?
- (RQ2) Can generative retrieval, with then-available architectures, match the effectiveness of then state-of-the-art dual encoders on large-scale corpora?
- (RQ3) Do architectural modifications like PAWA [234] offer benefits at scale?
- (RQ4) How does retrieval effectiveness change as we scale model size up to 11B parameters on a large corpus?

Through this study, we uncover the practical limits of generative retrieval and identify the key factors that govern its effectiveness at scale. Our findings clarify the state of this emerging paradigm and highlight the challenges that must be overcome for it to become a viable alternative to established retrieval architectures.

5.1 Methods

We revisit the design details of the generative retrieval method, using the Differentiable Search Index (DSI) [211] as the baseline, and describe multiple techniques introduced in subsequent works that we ablate and study in this work.

5.1.1 Background

DSI [211] transforms the retrieval task into sequence-to-sequence modeling, using queries as inputs and relevant document identifiers (DocIDs) as generation targets. The corpus, namely the mapping between the document’s content and its identifier, is encoded using the parameters of the LLM. DSI achieves this by leveraging two sequence-to-sequence tasks: indexing and retrieval. During training, the model learns to generate the DocID given the document content (indexing task) or a relevant query (retrieval task). During inference, the model processes a query and generates a ranked list of identifiers as retrieval results.

5.1.2 Inputs and Targets

In the framework discussed, DSI learns to encode the mapping between the long-form textual representation of a document and its identifier in its parameters, while also learning to fetch the same identifier when it receives a relevant query as input.

Two crucial design choices are how documents are represented (i.e., the inputs in the indexing task) and how DocIDs are defined (i.e., the targets in both indexing and retrieval tasks). One primary consideration is the challenge of encoding long textual sequences with a Transformer [223], making it difficult to index entire documents. Another is that naive identifiers from existing datasets could be sub-optimal due to their lack of semantic meaning.

Document Representations

One straightforward idea is to pick a text span from the document as a representation. DSI considers the first 64 tokens (FirstP) in each document, whereas Wang et al. [234] leverages ten randomly selected chunks of 64 consecutive tokens, a technique they call Document As Query (DaQ). We evaluate the FirstP and DaQ approaches individually and in conjunction when working with lengthy documents in the Natural Questions and TriviaQA datasets. In the case of MS MARCO, which has short passages, FirstP and DaQ are essentially the same, assuming sufficient context length.

Synthetic Query Generation

In training the model for retrieval tasks, the natural baseline uses existing labeled data, i.e., queries from the retrieval dataset as inputs and the DocIDs labeled as relevant as targets. However, as argued in Zhuang et al. [263] and Wang et al. [234], there are two kinds of gaps between the indexing and retrieval tasks. First is the data distribution gap: queries for the retrieval task are short and request specific information, while the documents for the indexing task are long and convey information. Second is the coverage gap: the model is exposed to the entire corpus during the training of the indexing task, while only positive examples have associated queries in the retrieval task. The latter problem is exacerbated in the MS MARCO passage ranking task as only 550K passages have an associated query for training the retrieval task, while the indexing task has to learn to encode all 8.8M passages in the corpus.

To mitigate this gap, they propose generating synthetic queries for each document using models like doc2query-T5 [155], discussed in Chapter 3. They then trained a generative retrieval model to predict the DocIDs given the corresponding synthetic queries. We can also think of these synthetic queries as alternative document representations.

Document Identifiers

In this work, we consider four kinds of different identifiers: the three kinds of document identifiers from DSI [211] — unstructured atomic identifiers (Atomic DocIDs), naive string identifiers (Naive DocIDs), and semantically structured identifiers (Semantic DocIDs), and a new kind from Wang et al. [234] — 2D Semantic DocIDs .

Atomic DocIDs. We treat each DocID as a single, or “atomic” token in this setting. The decoder then only needs to run for a single decoding step; we then sort the logits of the

DocIDs to obtain the ranked document list. Here, each document requires an additional unique token in the model vocabulary. As a result, the model’s parameters increase by the corpus size times the embedding dimension, becoming potentially expensive for large corpora. When considering millions of documents, we apply two optimizations to make the implementation more feasible. First, the embedding table of the encoder is adjusted to include only the standard T5 vocabulary, while the output projection of the decoder corresponds to DocIDs. Second, we take special care to ensure that the output projection is properly sharded across cores to distribute memory cost to allow scaling. We achieve this by setting partitioning rules in the `t5x` framework [189].

Naive DocIDs. Here, the original document identifier from a corpus is directly used and treated as a textual string. For example, a five-digit number “42915” is treated as a string and passed through the SentencePiece vocabulary of T5. Note that Naive DocIDs might sometimes capture semantics influenced by the curation pipelines that may reflect inherent relatedness.

Semantic DocIDs. Following Tay et al. [211], Semantic DocIDs aim to imbue document identifiers with hierarchical semantic information instead of relying on predefined identifiers. Specifically, after encoding documents into dense vectors, a hierarchical k -means algorithm recursively clusters the space into k clusters until individual clusters include no more than c documents. Consequently, all document identifiers form a tree, where non-leaf nodes correspond to superclusters, and leaf nodes have at most c documents each. Semantic DocIDs are formed by composing these cluster identifiers, each from 0 to $k - 1$, tailed by an identifier in the leaf nodes between 0 and $c - 1$. In this work, we use the identifiers generated by Wang et al. [234] for NQ and TriviaQA to perform a fair comparison. They achieve this with a 12-layer BERT model. For MS MARCO, we use SentenceT5-Base [152], and $c = 100$. Since the passage-level corpus is in the order of millions, if a cluster ends up with more than 1M documents, we sample 100K documents first when computing centroids. We used $k = 10$ clusters at each level, corresponding to the ten digits (0...9).

2D Semantic DocIDs. In the Semantic DocID setting, the same tokens represent different semantic meanings at different positions: we use the same set of tokens between 0 to $k - 1$ for all identifiers, but they represent semantic clusters at different levels in the tree. To address this, NCI [234] introduces a 2D variant by adding an extra dimension to encode the positions, making the model aware of levels of clustering when decoding the identifier. To allow better modeling with this variant, they additionally introduce a change to the decoder, Prefix-Aware Weight-Adaptive Decoding, which we describe in the next section.

5.1.3 Model Variants

Besides alternative ways of constructing model inputs and targets, generative retrieval approaches that build on DSI have also investigated novel modeling components. Here, we review three model components introduced by Wang et al. [234] and Bevilacqua et al. [15].

Prefix-Aware Weight-Adaptive (PAWA) Decoder is proposed as a method for decoding 2D Semantic DocIDs. Unlike a standard Transformer decoder, which uses the same matrix to project the hidden representation of the decoder to the vocabulary space for every position, a PAWA decoder uses different projection matrices at each timestep, with the weights of each projection matrix computed adaptively by a separate Transformer decoder. Specifically, in a vanilla decoder, the dense representation $h \in \mathbb{R}^{l \times d}$ from the last decoder layer is projected into the vocabulary space with $W \in \mathbb{R}^{d \times |V|}$, where l denotes the sequence length for decoding. To incorporate the position, the extra decoding module separately processes the input query and the already-decoded DocID tokens to output a projection matrix $W^{pawa} \in \mathbb{R}^{d \times l \times |V|}$, replacing W . This aims to capture that the semantic meaning of a DocID token depends on its position in the output sequence as well as on the DocID prefix preceding it. The experiments in this work use the open-source PAWA decoder implementation provided by the original authors¹ as a reference and build it out on `t5x`. For more details, please refer to NCI [234] and their codebase.

Constrained decoding can be used to avoid generating invalid document identifiers [15, 234]. While we have empirically found that roughly less than 1 in 20 DSI-based generation beams are invalid, we include this method nonetheless, as it is widespread in the literature. In this work, we adopt an exact match approach that leverages a trie to ensure we decode only valid document identifiers.

Consistency loss helps alleviate over-fitting by introducing a regularization term. The basic idea is that the representations generated by two forward passes with different dropout masks should be similar. Wang et al. [234] incorporate this insight into a regularization term that augments the generation loss. We investigate the softmax version as described in the NCI paper (Eq. 5 in [234]) and a KL-divergence version from an early version of NCI (Equation 5.1). They compute the Kullback-Leibler (KL) divergence between the output probabilities of two independent forward passes per position, where $p_{i,1}$ and $p_{i,2}$ are the probability distributions over the vocabulary space from the two forward passes at position i , respectively.

$$\mathcal{L}_{reg} = \frac{1}{2} [D_{KL}(p_{i,1} \parallel p_{i,2}) + D_{KL}(p_{i,2} \parallel p_{i,1})] \quad (5.1)$$

¹<https://github.com/solidsea98/Neural-Corpus-Indexer-NCI>

Dataset	#Docs	% Covered by train query set
NQ100K [234]	110K	98.4%
TriviaQA [234]	74K	57.7%
MSMarco100K	100K	92.9%
MSMarco1M	1M	51.6%
MSMarcoFULL	8.8M	5.8%

Table 5.1: The coverage statistics of the benchmark datasets and their training query sets.

While we closely follow the implementation of the Neural Corpus Indexer codebase, we find that these regularization terms lead to training instability and that the model’s effectiveness often diverges into a *NaN* loss. As a result, we do not include consistency regularization in our final experiments.

5.2 Experimental Setting

We limit ourselves to English retrieval tasks, focusing on the behavior of generative retrieval models at varying corpus scales.

5.2.1 Corpus and Training Data

Following small-scale generative retrieval experiment setups [211, 234, 263, 30], we start with experiments on the Natural Questions [103] and TriviaQA [93] datasets. To better understand how different model configurations perform at scale and in more practical settings, we also experiment with variants of the MS MARCO passage ranking dataset. The MS MARCO passage ranking collection consists of a corpus of 8.8M passages and a training set of 532K queries. From this dataset, we construct three variants, namely, MSMarco100K (100K passages), MSMarco1M (1M passages), and MSMarcoFULL (all 8.8M passages). It is worth noting that most documents in NQ100K and MSMarco100K have at least one relevant query in the training set. However, as we scale to MSMarcoFULL, the fraction of documents with queries in the training set drastically drops to around 6%, leading to a more practical setup. We summarize the statistics of these datasets in Table 5.1.

NQ100K & TriviaQA. To enable comparisons, we reuse the documents, the segmented documents, the training/testing splits, and the generated query sets from Wang et al. [234]. The Natural Questions and TriviaQA datasets have corpora of sizes 109K and 74K,

respectively. Note that Wang et al. [234] refers to NQ100K as NQ320K; we refer to the number of *unique* documents instead of the labeled training data size. Most documents in the NQ100K dataset have at least one relevant question in the training data, while 58% of the TriviaQA dataset has this property.

MSMarco100K, 1M, & FULL. In the same vein as NQ100K and TriviaQA, for MS-Marco100K, we curate a dataset with 100K passages sampled from the full MS MARCO passage ranking dataset. Most passages have at least one positive query for training. We also include passages relevant to the queries in the development dataset (for evaluation). MSMarco1M is $10\times$ larger than MSMarco100K. As with MSMarco100K, we augment the corpus with passages relevant to development queries. We first include all passages relevant to the 533K and 7K queries from the training dataset and development sets, respectively. This results in 516K and 7K unique passages from each set. We randomly sample passages without a query in either set to a total of a million passages. Finally, MSMarcoFULL scales up another order of magnitude in corpus size. As a result, only 5.8% of the passages have a corresponding query in the training set. We are not aware of any previous work that has attempted to apply generative retrieval models to a dataset of this size and complexity.

5.2.2 Synthetic Query Generation

For NQ100K and TriviaQA, we reuse the generated questions from Wang et al. [234] with 20 and 15 generated questions for each document, respectively. For the MS MARCO variants, we use doc2query-T5 [155] to generate questions, with 40 generated questions per passage. We also train a question-generation model using T5-base using training data from DPR [95], a retrieval dataset derived from NQ [103]. We use this model to generate 40 questions per passage, following the configuration of doc2query-T5. We refer to this setting for NQ and TriviaQA as “in-domain D2Q”.

5.2.3 Evaluation Dataset and Metrics

We evaluate using the development set of each dataset. For NQ100K, TriviaQA, and MS-Marco, the set contains 7,830, 7,993, and 6,980 queries, respectively. We use the same evaluation set for all three MS MARCO variants considered. For each query in the evaluation sets, we use the models to generate ranked lists of documents. We report Recall@1, Recall@5, and MRR@10 (Mean Reciprocal Rank) for NQ, TriviaQA, and MSMarco, respectively.

5.2.4 Model Variants

We evaluate all methods using a T5.1.1 backbone [186]. We test variants of labeled vs. synthetic queries, FirstP vs. DaQ document representations, and combinations of multiple representation choices. For each model variant, we ablate all versions of document identifiers when applicable. Modifications in model architecture are introduced in a stacking fashion, starting with the base model and then adding on the PAWA decoder, constrained decoding, and consistency loss in this order. Note, we only evaluate the PAWA decoder with 2D Semantic DocIDs as Wang et al. [234] introduce the decoder to exploit their unique representation hierarchy.

For model scaling experiments, we mainly investigate whether Atomic DocIDs are an effective way to scale to millions of passages, given the parameter cost. As such, we consider larger models with Naive DocIDs and Semantic DocIDs comparable to T5-Base with Atomic DocIDs, which total 7B parameters when scaling to 8.8M DocIDs.

For baselines we provide BM25 [191] and BM25 with doc2query-T5 [155]. For Natural Questions and TriviaQA, we also include the previous results reported for the NCI-variant of NQ (i.e., NQ100K). The baselines include state-of-the-art generative retrieval results like NCI and GenRet [206], and GTR-Base, a then state-of-the-art dual encoder [153]. For the MS MARCO variants we introduce, we provide our own GTR-Base [153] results.

5.3 Experimental Results

We report our results in three parts. First, we ablate all the methods from Section 5.1 using T5-Base on small-scale datasets: NQ100K and TriviaQA. We observe which techniques work best on this small scale with widely studied datasets. Then, we transfer the same techniques and scale up to the entire MS MARCO passage ranking dataset. Finally, to understand whether we can attribute the effectiveness bump from Atomic DocIDs to additional model parameters, we fix the corpus to be MSMarcoFULL and scale the model size up to 11B. Across the tables, At. refers to Atomic DocIDs, Nv. Naive DocIDs, and Sm. Semantic DocIDs.

5.3.1 Ablations over Small Corpora

We report our ablations over NQ100K and TriviaQA in Table 5.2. The best combination of our techniques (row (7)) sets a new state-of-the-art result on the NCI variant of

Model	NQ100K			TriviaQA		
	At.	Nv.	Sm.	At.	Nv.	Sm.
<i>Baselines</i>						
BM25 (via Wang et al. [234])	-	15.1	-	-	56.9	-
BM25 w/ doc2query-T5 (via Wang et al. [234])	-	35.4	-	-	59.7	-
GTR-Base (via Sun et al. [206])	-	56.0	-	-	-	-
NCI [234]	-	62.8	65.9	-	88.8	90.5
GenRet [206]	-	-	68.1	-	-	-
<i>Ours</i>						
(1a) Labeled Queries (No Indexing)	50.7	49.2	49.0	60.9	56.7	61.4
(2a) FirstP + Labeled Queries (DSI)	60.0	58.4	58.7	71.6	75.2	78.9
(2b) DaQ + Labeled Queries	61.4	60.4	60.0	81.0	80.4	77.6
(3a) DaQ + D2Q + Labeled Queries	69.6	67.9	67.9	88.2	85.7	86.3
(3b) FirstP + DaQ + D2Q + Labeled Queries	69.0	68.2	67.2	88.9	86.9	87.4
(4a) 3b + PAWA (w/ 2D Semantic DocIDs)	-	-	66.3	-	-	86.5
(4b) 3b + Constrained Decoding	-	-	67.3	-	-	87.3
(5) 4b + Consistency Loss (NCI)	-	-	66.3	-	-	86.6
(6a) DaQ Only	17.1	18.4	15.6	41.0	31.3	20.6
(6b) D2Q Only	43.6	42.3	42.9	61.9	57.8	57.1
(6c) 6b + PAWA (w/ 2D Semantic DocIDs) + Constrained Decoding	-	-	43.1	-	-	57.7
(7) 3b + in-domain D2Q	70.7	69.7	69.5	90.0	88.0	89.2

Table 5.2: Results on small-scale NQ (Recall@1) and TriviaQA (Recall@5) datasets. We present ablation results in a stacking fashion. Rows (6a-c) isolate the importance of D2Q. Last row revises the best method with in-domain D2Q.

Natural Questions without using sophisticated modeling techniques, architectural changes, and learned DocIDs. The choice of document representation by far dominates the overall effectiveness of the retriever. Using just the training queries provided by the datasets demonstrates poor results due to the low coverage of the documents. With FirstP, we see considerable improvements over this setting. The DaQ representation strategy further improves over FirstP, exposing the model to different views of the same document. However, synthetic queries from doc2query-T5 are essential to high generative retrieval effectiveness, resulting in a gain of more than 7 points. This technique, by far, trumps all other proposed changes. Notably, the *quality* of the synthetic queries is very important, with those from a generator we specifically trained for the question-answering domain outperforming the query generator trained over MSMarcoFULL.

As for other design choices, for this small-scale setup, Naive and Semantic DocIDs compete on par with each other (varying between task configurations), and Atomic DocIDs are consistently the best. We note, though, that on NQ100K, Atomic DocIDs add 80M parameters to a T5-Base model that would otherwise be 220M parameters (a 36% increase). Given the comparable effectiveness in the best configuration (row (7)), these extra

parameters may or may not be worth it. Modeling techniques from [234], i.e., 2D Semantic DocIDs, PAWA, constrained decoding, and consistency loss, do not reliably improve the model over using synthetic queries alone.

At this corpus scale, our best result uses a mixture of FirstP, DaQ, labeled queries, and synthetic queries for training.

To answer **RQ1** at small corpus scales:

RQ1 At the 100K scale, synthetic queries from doc2query-T5 provide the most gains, with domain-specific query generators leading to better effectiveness. FirstP and DaQ representations offer substantial value when indexing document collections. Among identifier schemes, Atomic Doc-IDs consistently perform best, though Naive and Semantic Doc-IDs achieve comparable effectiveness.

5.3.2 Scaling Corpus Size

We now consider variants of the MS MARCO passage ranking task, scaling from 100K to 1M and 8.8M passages. Results are reported in Table 5.3. Perhaps the most striking observation about the transition to the MS MARCO passage ranking collection is the absolute requirement of synthetic queries for strong retrieval effectiveness. Synthetic queries result in a 2-3 $\times$ improvement over the original DSI formulation alone. Using only synthetic queries as the indexing task is the most effective and straightforward training strategy for the MS MARCO collection. This result signifies a notable difference in the transition from NQ and TriviaQA to MS MARCO, where FirstP and DaQ did provide substantial value. The reason may be an artifact of the NQ and TriviaQA datasets leveraging a corpus of Wikipedia articles: the beginnings of Wikipedia documents are informative entity descriptions, and many sentences refer to the entity—which is likely the answer to a requested query.

As we scale the corpus size, DSI effectiveness drops rapidly, with the best result (D2Q only with Atomic DocIDs) falling off from 80.3 to 55.8 and finally 24.2 as we scale to the whole 8.8M passages. The Semantic DocID setting, too, drops off as we scale to the whole corpus, under-performing the Naive DocID setting. We conjecture that this may be due to the potentially increased length of Semantic DocIDs being more difficult to decode than Naive DocIDs, coupled with a noisy partitioning of the semantic space (especially when using an off-the-shelf embedding model such as SentenceT5-Base). However, we observe Semantic DocIDs decoded via the PAWA decoder score higher. We hope to shed some insight into why this might be in the next section, where we examine model size.

Model	MSMarco100K			MSMarco1M			MSMarcoFULL		
	At.	Nv.	Sm.	At.	Nv.	Sm.	At.	Nv.	Sm.
<i>Baselines</i>									
BM25	-	65.3	-	-	41.3	-	-	18.4	-
BM25 (w/ doc2query-T5)	-	80.4	-	-	56.6	-	-	27.2	-
GTR-Base	-	83.2	-	-	60.7	-	-	34.8	-
<i>Ours</i>									
(1a) Labeled Queries (No Indexing)	0.0	1.1	0.0	0.0	0.5	0.0	0.0	0.0	0.0
(2a) FirstP/DaQ + Labeled Queries (DSI)	0.0	23.9	19.2	2.1	12.4	7.4	0.0	7.5	3.1
(3b) FirstP/DaQ + D2Q + Labeled Queries	79.2	77.7	76.8	53.3	48.2	47.1	14.2	13.2	6.4
(4a) 3b + PAWA (w/ 2D Semantic DocIDs)	-	-	77.1	-	-	50.2	-	-	9.0
(5) 4a + Consistency Loss (NCI)	-	-	77.1	-	-	50.2	-	-	9.1
(6b) D2Q only	80.3	78.7	78.5	55.8	55.4	54.0	24.2	13.3	11.8
(4a') 6b + PAWA (w/ 2D Semantic DocIDs)	-	-	78.2	-	-	54.1	-	-	17.3
(4b') 6b + Constrained Decoding	-	-	78.6	-	-	54.0	-	-	12.0
(5') 6b + PAWA (w/ 2D Semantic DocIDs) + Constrained Decoding	-	-	78.3	-	-	54.2	-	-	17.4

Table 5.3: Results on the development set of the scale variant MS MARCO V1 passage collections, reported in MRR@10. Best results per column and results within 0.1 of best are bolded. Note that FirstP here is equivalent to DaQ as MS MARCO input passages fit into the input window.

Constrained decoding only provides marginal value and generally is not worth the added complexity.

Continuing to answer **RQ1**, as corpus size scales:

RQ1 Scaling from 100K to 8.8M passages reveals drastic shifts in what techniques work. Synthetic queries transition from helpful to essential and become the most effective training strategy. In contrast, FirstP and DaQ representations lose their value on passage-based corpora. All document identifier schemes show large drops in effectiveness.

Next, to address **RQ2**:

RQ2 At large scale, our generative retrievers could not match state-of-the-art dual encoders. The best configuration on MSMarcoFULL falls significantly short of dense retrievers like GTR-base. This gap reflects the considerable challenge of encoding an entire 8.8M passage corpus within model parameters (220M for T5-Base) compared to dual encoders that can offload dense representations to external indices.

T5 Scale	Training	Params	Inference FLOPs	MRR@10
Base	D2Q Only + Atomic DocID	7.0B	0.9×10^{12}	24.2
Base	D2Q Only + Naive DocID	220M	1.4×10^{12}	13.3
Base	D2Q Only + PAWA (2D Sem.)	761M	6.8×10^{12}	17.3
Large	D2Q Only + Naive DocID	783M	3.5×10^{12}	21.4
Large	D2Q Only + PAWA (2D Sem.)	2.1B	1.1×10^{13}	19.8
XL	D2Q Only + Naive DocID	2.8B	9.3×10^{12}	26.7
XXL	D2Q Only + Naive DocID	11B	4.3×10^{13}	24.3

Table 5.4: The effect of scaling up model size for sequential DocID approaches in comparison to Atomic DocIDs for the MSMarcoFULL task.

5.3.3 Scaling Model Size

How much of the Atomic DocID setting’s effectiveness can be attributed to its additional model parameters? On MSMarcoFULL, decoding Atomic DocID tokens adds an overhead of 7B parameters to the otherwise 220M-parameter T5-Base model. We take the best configuration on MSMarcoFULL from Table 5.3 and scale model parameters of Naive DocID and Semantic DocID (PAWA) to similar sizes for comparison. We report results in Table 5.4.

Overall, we observe a general trend that as parameter count increases, retrieval effectiveness improves. Indeed, both Atomic DocIDs and PAWA Semantic DocIDs had the highest scores in Table 5.3, which we now attribute to their increased size. Notice that the difference here only comes out when scaling to MSMarcoFULL, where these parameter differences magnify significantly over corpora of a smaller scale. However, not all methods are equal. PAWA and 2D Semantic DocIDs [234] considerably increase decoding parameters with its extra decoding stack, yet yield no gain over naively scaling the Transformer with Naive DocIDs, underperforming by 4 points at around ~ 800 M parameters. This pattern continues to hold, scaling PAWA to 2.1B parameters. Thus, to conserve resources, we do not scale PAWA any further.

Scaling Transformers naively according to default T5 scales while using Naive DocIDs had the highest effectiveness on MSMarcoFULL at 26.7 MRR@10. Despite using only ~ 2.8 B parameters, this approach outperforms T5-Base with Atomic DocIDs, which uses 7B parameters while achieving only 24.2 MRR@10. However, while parameter count has practical implications regarding the resources required for training and inference (especially TPU/GPU memory), there are other trade-offs to consider, like FLOPs.

While Naive DocIDs perform well at T5-XL size, surprisingly, we find that scaling further to XXL (11B) does not improve effectiveness; in fact, it is detrimental to retrieval effectiveness (24.3 MRR@10 vs. XL’s 26.7) under the same experimental settings and hy-

Model / T5 Scale	Training	Params	MSMarcoFULL		TREC DL 19	TREC DL 20
			Recall@100	nDCG@10	nDCG@10	nDCG@10
<i>Baselines</i>						
BM25	–	–	65.8	22.8	–	–
BM25 w/ doc2query	–	–	81.9	33.8	–	–
GTR-Base [153]	–	110M	89.8	42.0	–	–
<i>Ours (T5 Scale)</i>						
Base	D2Q Only + Atomic DocID	7.0B	80.5	29.7	49.8	53.8
Large	D2Q Only + Naive DocID	783M	82.0	27.3	49.9	45.2
Large	D2Q Only + PAWA (2D Sem.)	2.1B	73.2	24.8	46.6	50.1
XL	D2Q Only + Naive DocID	2.8B	84.7	33.2	55.0	52.2
XXL	D2Q Only + Naive DocID	11B	81.9	30.6	52.0	49.0

Table 5.5: Additional evaluation of the top five model checkpoints from Table 5.4 on TREC DL 19 & 20. We decode using 100 beams here. The same metrics are reported over the original MSMarco validation set, with the addition of Recall@100.

perparameter settings, even though model training converges faster. This finding is counterintuitive to most generation tasks and the common conception of generative retrieval relying on model capacity for retrieval effectiveness.

To strengthen the evidence from our scaling results, we provide results on TREC DL 19 and 20 [41, 42] in Table 5.5. These additional densely judged evaluation sets, as opposed to the sparse judgments in the MS MARCO Passage development set, seek to address some of the issues with the original task like sparse judgments. We see that our original findings generally hold under these new metrics and believe that this reaffirms the soundness of our study.

In conclusion, to answer **RQ3**:

RQ3 Despite the extra decoding stack increasing parameters, PAWA with 2D Semantic DocIDs yields no gain over naive scaling with Naive DocIDs.

Additionally, in response to **RQ4**:

RQ4 Scaling model size on MSMarcoFULL reveals non-monotonic behavior. Effectiveness initially improves with larger models, then drops. This contradicts assumptions that generative retrieval effectiveness primarily depends on model capacity, showing that simply applying more compute yields diminishing and eventually negative returns.

5.4 Concluding Thoughts

This chapter presented the first large-scale empirical study on the scalability of generative retrieval. Our experiments, scaling up to 8.8 million passages and 11B parameter models, revealed several critical insights. We found that synthetic query generation is the single most important technique for effective retrieval, particularly as corpus size grows. Furthermore, we demonstrated that many proposed architectural complexities offer little benefit over naively scaling model parameters, and even then, simply increasing model size yields diminishing returns.

While generative retrieval showed promise on smaller corpora, our findings demonstrate that it struggles to compete with established dual-encoder architectures at a large scale. The paradigm faces significant, unresolved challenges in both effectiveness and efficiency that are not overcome by simply applying more compute. This conclusion, however, does not diminish the potential of large sequence-to-sequence models in information retrieval. It instead suggests that their strengths may be better applied to different stages of the retrieval pipeline. The next chapter confronts the Component Challenge from a different angle, moving back to the reranking stage. There, we will explore how the powerful zero-shot capabilities of LLMs can be distilled into highly effective, and crucially, open-source listwise rerankers, demonstrating a more viable path for integrating these models into modern information access systems.

Chapter 6

RankZephyr: An Open-Source Listwise Reranker

As established in [Chapter 1](#), modern information access systems are increasingly architected as complex, multi-stage pipelines where reranking models play a pivotal role in delivering precise results. The effectiveness of these pipelines, however, is often constrained by a growing reliance on proprietary, black-box models that limit reproducibility and hinder innovation. Zero-shot listwise rerankers, or “prompt-decoders”, represent the state of the art in this critical stage [\[207, 135\]](#), but large, closed-source LLMs like GPT₄ have dominated their development. While early open-source alternatives such as RankVicuna [\[175\]](#) made essential strides, a significant effectiveness gap remained, underscoring the need for a truly competitive, open alternative.

This chapter bookends the Component Challenge by introducing RankZephyr [\[176\]](#), an open-source 7B-parameter listwise reranker designed to provide a transparent, accessible, and highly effective building block for advanced information access systems. The following research questions guide this work:

- (RQ1) Can an open-source 7B LLM match or exceed the reranking effectiveness of larger proprietary models like GPT₄ in a zero-shot setting?
- (RQ2) What training strategies (e.g., choice of teacher model, hard negatives, window sizes) are most critical for building a robust listwise reranker?
- (RQ3) How sensitive are listwise rerankers to the initial ordering of documents?

- (RQ4) How does the choice of the first-stage retriever and the number of candidates affect downstream reranking effectiveness?
- (RQ5) How well does RankZephyr generalize to uncontaminated datasets?

Through comprehensive experiments, we demonstrate that RankZephyr achieves state-of-the-art effectiveness, is robust to different retrieval configurations, and generalizes well to novel data. We provide the research community with a powerful, transparent, and accessible reranking module, fostering reproducible research in modern information access systems.

6.1 Methodology

The core of RankZephyr is a strategic training methodology that enables a 7B-parameter model to rival much larger systems. The process involves a carefully designed prompt and a two-stage distillation curriculum.

6.1.1 Prompt Design

Effective LLM-based reranking requires a clear and structured prompt. We adapt the successful templates from Sun et al. [207] and Pradeep et al. [175] for RankZephyr. The reranking task is framed as follows: given a query q and candidate documents $\{D_1, \dots, D_n\}$, the model’s objective is to reorder the documents to optimize a metric that rewards highly relevant documents at the top of the list.

Our prompt, shown in Figure 6.1, provides the model with the query and documents, instructs it on the task, and specifies the output format. The same prompt is used for all listwise rerankers described in this section: RankZephyr, RankGPT_{3.5}, and RankGPT₄.

6.1.2 Training Stage 1: Distilling from GPT_{3.5}

The first training stage builds foundational ranking capabilities by distilling knowledge from RankGPT_{3.5}. Following the data preparation of Sun et al. [207], we use 100K queries from the MS MARCO v1 passage ranking training set. For each query, the top 20 passages retrieved by BM25 are reranked by RankGPT_{3.5} to create teacher rankings.

SYSTEM: You are RankLLM, an intelligent assistant that can rank passages based on their relevancy to the query.

USER: I will provide you with `{num}` passages, each indicated by a numerical identifier `[]`. Rank the passages based on their relevance to the search query: `{query}`.

[1] `{passage 1}`
...
`{num}`] `{passage {num}}`

Search Query: `{query}`.

Rank the `{num}` passages above based on their relevance to the search query. All the passages should be included and listed using identifiers, in descending order of relevance. The output format should be `[] > []`, e.g., `[4] > [2]`. Only respond with the ranking results, do not say any word or explain.

LLM: `[9] > [4] > [20] > ... > [13]`

Figure 6.1: The reranking prompt template and sample generation for RankZephyr.

To improve training data quality, we filter out malformed generations from the teacher model. We also augment the training data by shuffling the input order of the BM25 candidates. This presents the model with a more challenging reordering task and helps it learn to identify ideal orderings independent of the initial ranking.

We fine-tune the 7B parameter Zephyr- β model [220] for three epochs using the Axolotl library. Training incorporates a batch size of 64, a learning rate of 5×10^{-6} , bfloat16 precision, and noisy embeddings to improve instruction following [90]. This stage establishes strong baseline effectiveness but still lags behind GPT₄, motivating a second stage of refinement.

6.1.3 Training Stage 2: Distilling from GPT₄

The second stage refines the model using higher-quality supervision from RankGPT₄. Given the significant cost of generating data with GPT₄, we use a smaller, strategically selected dataset of 5K queries. The selection methods for these queries are as follows:

- **Random:** Queries are chosen randomly from the set in [Section 6.1.2](#).
- **Discriminative:** Queries are iteratively selected based on their distinctiveness. We do this by choosing a query from the remaining set whose ADA₂ embedding minimizes the maximum inner product score with the embeddings of the already selected queries, promoting diversity in the training samples.

Queries are selected using a combination of these two strategies.

A key improvement in this stage is substituting BM25 candidates with candidates retrieved by ADA₂, a more powerful zero-shot model [148]. This exposes our model to “harder” candidate sets during training, akin to using hard negatives in cross-encoder training as seen in Chapter 4, thereby improving its discriminative power. Additionally, to address a limitation in RankVicuna [175], which was trained on fixed-size lists of length 20, we train RankZephyr on random subsets of documents ($p \leq 20$). This variable-window training enables the model to handle varying candidate lists during inference and diversifies the training data. We warm-start this stage with the checkpoint from Stage 1, significantly reducing the computational cost.

6.1.4 Inference

During inference, RankZephyr can rerank lists of up to 20 candidates in a single pass. For larger sets (e.g., top 100), we employ a sliding window approach with a window size of 20 and a stride of 10 candidates, balancing effectiveness and efficiency. That is, for top 100 reranking, we first reorder candidates 81–100 in the list, then candidates 71–90 in the *new list*, and so on, until we finally reorder candidates 1–20 in the penultimate list.

We also support progressive reranking (RankZephyr _{ρ}), where the model performs multiple sequential passes over the *entire* list. This iterative refinement process, typically using three passes, often improves final ranking quality.

6.2 Experimental Setup

6.2.1 Datasets

We evaluate on several standard test collections:

- **MS MARCO v1:** The TREC 2019 and 2020 Deep Learning Tracks [41, 42], which are in-domain relative to our training data.
- **MS MARCO v2:** The TREC 2021 and 2022 Deep Learning Tracks [43, 44], which test generalization to a larger, different corpus, albeit with similar queries as DL19–20.

- **BEIR:** The NEWS and COVID collections from the BEIR benchmark [213], which provide out-of-domain evaluation scenarios.
- **NovelEval:** A dataset from Google Web Search containing queries and passages that postdate the model’s training cutoff, used to assess generalization to unseen data and mitigate data contamination concerns [207].

6.2.2 Baseline Systems

We compare RankZephyr against a diverse set of baseline systems spanning traditional, neural, and LLM-based approaches. For traditional retrieval, we include BM25 [191]. Neural retrieval methods include the sparse retriever, SPLADE++ EnsembleDistill [59], and the dense retriever, RepLLaMA [134]. For reranking, we evaluate supervised pointwise models like RankLLaMA-13B [134], as well as proprietary LLM-based listwise rerankers such as RankGPT_{3.5} and RankGPT₄ [207]. Finally, we include an open-source listwise LLM reranker, RankVicuna [175].

6.2.3 Evaluation Metrics and Procedures

We use normalized discounted cumulative gain at a rank cutoff of 10 (nDCG@10) as the primary evaluation metric. Results for proprietary models are averaged over multiple runs (six for GPT_{3.5}, three for GPT₄) to account for their non-deterministic nature. Unless specified, RankZephyr results are from the complete two-stage training process, using a single reranking pass. RankZephyr _{ρ} results correspond to the same model with three progressive passes.

6.3 Results and Analysis

6.3.1 Main Effectiveness Results

Table 6.1 compares different reranking pipelines using DL19 and DL20 data from the MS MARCO v1 passage ranking task, DL21 and DL22 data from the MS MARCO v2 passage ranking task, and NEWS and COVID from the BEIR collection. Rows (1–2) report baselines using only first-stage retrievers: BM25 [191] and SPLADE++ EnsembleDistil (ED) [59]. The first represents a traditional “bag-of-words” retrieval baseline, while

	Source		MSv1		MSv2		BEIR	
	Prev.	Top- k	DL19	DL20	DL21	DL22	NEWS	COVID
(1) BM25	None	$ C $	0.5058	0.4796	0.4458	0.2692	0.3952	0.5947
(2) SPLADE++ ED	None	$ C $	0.7308	0.7195	0.6846	0.5705	0.4152	0.7274
(3) RankLLaMA-13B	RepLLaMA	100	0.7567	0.7748	-	-	-	-
(4) RankGPT ₄	SPLADE++ ED	100	0.7464	0.7076	0.7721	0.7175	0.5327	0.8792
(5) RankGPT ₄ (PSC)	SPLADE++ ED	100	0.7601	0.7514	-	-	-	-
(6) RankVicuna	SPLADE++ ED	100	0.7459	0.7473	0.7011	0.5817	0.4750	0.8283
(7) RankZephyr	SPLADE++ ED	100	0.7816	0.8159	0.7598	0.6692	0.5060	0.8535
(8) RankZephyr _{ρ}	SPLADE++ ED	100	0.7855	0.8255	0.7688	0.6628	0.5107	0.8566

Table 6.1: nDCG@10 on the primary datasets. Each reranker uses the top 100 retrieved results of the previous stage as input.

the second is a supervised neural sparse method that has proven effective in a wide range of tasks. Row (3) corresponds to RankLLaMA-13B [134], a then state-of-the-art fine-tuned reranker for the MS MARCO v1 passage ranking task, reranking the top 100 candidates retrieved by RepLLaMA, a neural dense method, both of which were proposed by Ma et al. [134]. It is worth clarifying that despite the name similarities, RankLLaMA is a *pointwise* reranker.

The remaining rows report the effectiveness of using LLM prompt-decoders to reorder the top 100 candidate documents retrieved by SPLADE++ ED. These include a mix of proprietary models, rows (4–5), and open-source models, rows (6–8). Row (5) corresponds to the Permutation Self-Consistency method [209] employed to aggregate multiple RankGPT₄ results.

Focusing on the MS MARCO v1 passage ranking task, with the RankZephyr model, we establish state-of-the-art results:

- RankZephyr improves over the highly effective pointwise RankLLaMA-13B, despite not requiring human labels and having only 7B parameters, rows (7) vs. (3). This holds even when we rerank the RepLLaMA candidate set; see row (3c) in Table 6.4 for more details.
- RankZephyr improves over its proprietary RankGPT₄ teacher, rumored to be over two orders of magnitude larger, rows (7) vs. (4). The effectiveness boost with RankZephyr is considerably higher in cases where we are processing higher-quality candidate lists from RepLLaMA (more in Section 6.3.4) or SPLADE++ ED, even seeing relative gains as high as 15%. Our method also outshines PSC-aggregated RankGPT₄, row (5), which

further improves RankGPT₄ by marginalizing out different list orders in the prompt to aggregate a final ranked list.

These results demonstrate the superiority of our much smaller and open-source, RankZephyr model as an effective and robust reranker. We improve considerably upon RankVicuna, row (6), the prior state of the art for open-source prompt-decoders.

Additionally, we find that progressive reranking, shown in row (8), generally improves effectiveness compared to the single-pass counterpart, row (7), leading to our best results. This achieves the state of the art for DL19 and DL20, whether open-source or proprietary. With more refined multi-stage ranking pipelines [106] feeding even better candidates to RankZephyr _{ρ} , we would expect even higher scores. Note that our best system, RankZephyr _{ρ} , row (9), has Judged@10 rates of 0.942 and 0.983 for DL19 and DL20, respectively. This means that the nDCG@10 scores reported here represent a lower bound of what such a run would score if part of the judgment pool.

On the MS MARCO v2 passage ranking task and NEWS and COVID tasks from BEIR, whose corpora and queries were never exposed during training, we find that RankZephyr and RankZephyr _{ρ} remain very effective; here, progressive ranking also helps, rows (8) vs. (7). Both methods bring huge improvements over their first-stage retriever, SPLADE++ ED, row (2), and RankVicuna.

However, on these datasets, our models still underperform their RankGPT₄ teacher when reranking SPLADE++ ED output. Interestingly, though, when reranking lower-quality candidates such as BM25, our models exhibit higher effectiveness than RankGPT₄ (see discussion in [Section 6.3.4](#)). Nonetheless, this observation helps us understand the current limitations of our models and shows that more work is needed before we have models that are consistently better than RankGPT₄. We present two plausible explanations for this gap: First, RankZephyr is instruction fine-tuned only on data with a context size of 4096, as opposed to the 8192 context of RankGPT₄, which allows it to better attend to longer biomedical abstracts and news articles in BEIR. Second, we note 2–4 points lower Judged@10 scores for RankZephyr compared to RankGPT₄. Despite this, our model marks a considerable push towards state-of-the-art generalizable rerankers.

To answer **RQ1**:

RQ1 We find that an open-source 7B parameter model can match and often exceed the reranking effectiveness of much larger proprietary models like GPT₄. RankZephyr achieves state-of-the-art results on MS MARCO v1 benchmarks, demonstrating that strategic distillation and training can overcome the parameter count disadvantage.

	Teacher	Training Source	Prev. Disc.	DL19		DL20	
				BM25	ADA ₂	BM25	ADA ₂
(1) RankVicuna	RankGPT _{3.5}	BM25	×	0.6682	0.7374	0.6549	0.7210
(2) RankZephyr	RankGPT _{3.5}	BM25	×	0.7078	0.7161	0.6687	0.7008
(3a) RankZephyr	RankGPT ₄	BM25	×	0.7374	0.7645	0.6924	0.7503
(3b) RankZephyr	RankGPT ₄	BM25	✓	0.7275	0.7461	0.6851	0.7550
(4a) RankZephyr	RankGPT ₄	ADA ₂	×	0.7331	0.7627	0.7185	0.7762
(4b) RankZephyr	RankGPT ₄	ADA ₂	✓	0.7248	0.7513	0.7144	0.7779
(5) RankZephyr	RankGPT ₄	ADA ₂	∪	0.7364	0.7502	0.6941	0.7677

Table 6.2: nDCG@10 on DL19 and DL20 for different reranking methods, with RankGPT_{3.5} or RankGPT₄ as the teacher, and BM25 or ADA₂ as the training source. The method to select queries (Disc.) can be random (×), discriminative (✓), or both (∪). Each reranker uses the top 100 retrieved results of the previous stage as input.

6.3.2 Training Strategy Ablation

Table 6.2 investigates the choices made in the methodology and their impact on DL19 and DL20 effectiveness using BM25 and ADA₂.

Row (1) shows the RankVicuna baseline from Pradeep et al. [175], which leveraged RankGPT_{3.5} reorderings of the BM25 ranked lists of 100K MS MARCO v1 training queries. Row (2) examines the effect of moving from Vicuna to Zephyr, of comparable parameter count. Note that for DL19 and DL20, when reranking BM25 candidates, RankZephyr demonstrates higher effectiveness, but when reranking ADA₂ candidates, RankVicuna scores higher. RankZephyr scores higher on average and Zephyr is overall a more capable LLM, providing the rationale for our work to build on Zephyr.

Rows (3–5) show the effectiveness of the model corresponding to row (2) fine-tuned on RankGPT₄ data; rows (3–4) use 1K queries, and row (5) uses 2K queries. We see using ADA₂ orderings as input instead of BM25 generally results in better DL20 effectiveness while having comparable DL19 scores, rows (4*) vs. (3*). This is expected, as the top 20 ADA₂ orderings surpass those of BM25, providing RankZephyr with more sophisticated reorderings from the teacher.

Discriminative sampling generally leads to marginally reduced effectiveness, shown in rows (*b) vs. rows (*a). This observation indicates that more than covering tail queries, exposing the model to a general distribution of queries and teacher reorderings is essential.

To investigate whether combining various query selection strategies is beneficial, we amalgamated the query sets from rows (4a) and (4b) along with their respective input and

	Source	Prev.	Top- <i>k</i>	DL19		DL20	
				nDCG@10	MAP@100	nDCG@10	MAP@100
(1a) RankGPT _{3.5}	BM25		100	0.6855	0.3335	0.6202	0.3525
(1b) RankGPT _{3.5}	Shuf. BM25		100	0.6158±0.012	0.2810±0.007	0.5516±0.011	0.2929±0.007
(1c) RankGPT _{3.5}	SPLADE++ ED		100	0.7504	0.4731	0.7120	0.5011
(1d) RankGPT _{3.5}	Shuf. SPLADE++ ED		100	0.6028±0.007	0.3403±0.005	0.5479±0.013	0.3500±0.009
(2a) RankZephyr	BM25		100	0.7420	0.3683	0.7086	0.4151
(2b) RankZephyr	Shuf. BM25		100	0.7378±0.009	0.3421±0.008	0.7068±0.011	0.3910±0.007
(2c) RankZephyr	SPLADE++ ED		100	0.7816	0.4908	0.8159	0.5754
(2d) RankZephyr	Shuf. SPLADE++ ED		100	0.7670±0.011	0.4287±0.005	0.7910±0.004	0.5112±0.004

Table 6.3: nDCG@10 and MAP@100 on DL19 and DL20 for RankGPT_{3.5} and RankZephyr with shuffling on different first-stage retrieval models. For each, we perform reranking using the top 100 candidates from the previous stage on six shuffled orderings. We report average metrics with 99% confidence intervals.

teacher reorderings, shown in row (5). We observe no notable improvements in effectiveness (despite using more queries). Consequently, we decided to expand the method described in row (4a) — employing RankGPT₄ reorderings of ADA₂ input orderings for a randomly chosen training query set comprising 5K examples. This analysis confirms that careful design choices — selecting stronger teacher models and training with harder negatives — are critical for building robust rerankers, addressing RQ2. Unless explicitly stated otherwise, we refer to this setting as the “full” RankZephyr model.

In response to **RQ2**:

RQ2 We find that several training strategies are critical for building powerful listwise rerankers: (1) using RankGPT₄ as the teacher model in Stage 2; (2) training with harder negatives from ADA₂-retrieved candidates instead of BM25; (3) random query sampling proves more effective than discriminative sampling.

6.3.3 Robustness Analysis

The training methodology of RankZephyr includes sequences formed from altering the input order of documents and accordingly modifying the teacher generations. This process presents a more complex reordering challenge to the model, contributing to its robustness and effectiveness. Our investigation examines whether the effectiveness improvements observed in previous open-source LLMs persist in RankZephyr and how it compares with RankGPT_{3.5}. We present our experimental findings in Table 6.3.

An examination of rows (2a) and (2b), as well as (2c) and (2d), reveals that RankZephyr retains its effectiveness with shuffled inputs. A minor decrease in effectiveness is expected, possibly attributable to the model’s inability to simultaneously promote multiple relevant documents in cases where they are initially ranked lower, such as in ranks 81–100. Such occurrences are relatively rare, but can actually appear anywhere in the ranked lists.

Conversely, RankGPT_{3.5} demonstrates a marked decrease in effectiveness when fed shuffled inputs, as indicated by the contrast between rows (1b) and (1a), and (1d) and (1c). Notably, the effectiveness of RankGPT_{3.5} in reranking shuffled SPLADE++ ED is even lower than that of reranking shuffled BM25. This pronounced dependence on input order underscores a lack of robustness.

To answer **RQ3**:

RQ3 We find that RankGPT_{3.5} exhibits marked effectiveness degradation, while RankZephyr demonstrates strong robustness to input perturbations. This stems from our training data augmentation, exposing the model to varied input orderings.

Given that even good retrievers might produce sub-optimal rankings that behave like shuffled input, it is vital to build methods like RankZephyr or corrective strategies like RankZephyr _{ρ} to ensure both effectiveness and robustness.

6.3.4 First-Stage Retrieval Impact

This section delves into the influence of the quality and quantity of candidates generated by the first-stage retriever on the final output. To this end, our experiments employed four first-stage retrieval models, each with either the top 20 or top 100 results. The methods include: (1) BM25 [191], (2) SPLADE++ EnsembleDistil (ED) [59], (3) RepLLaMA [134], and (4) ADA₂ [148, 121]. Results are presented in Table 6.4.

Our findings clearly demonstrate that a better first-stage retrieval model leads to better overall reranking effectiveness. Specifically, stronger first-stage retrieval models such as SPLADE++ ED, rows (2*), and RepLLaMA, rows (3*), markedly improve downstream effectiveness. We expect to see this, as these methods yield better-quality candidate lists for reranking. However, a notable observation emerges when comparing the downstream effectiveness of reranking RepLLaMA vs. reranking SPLADE++ ED. Despite RepLLaMA exhibiting a higher MAP@100, reranking it does not translate to better results than reranking SPLADE++ ED, noticeable in DL19 and DL20 effectiveness, row (2c) vs. (3c). While

	Source	Prev.	Top- k	DL19		DL20	
				nDCG@10	MAP@100	nDCG@10	MAP@100
(1a) BM25	None		$ C $	0.5058	0.2476	0.4796	0.2685
(1b) RankZephyr	BM25		20	0.6489	0.3060	0.6196	0.3358
(1c) RankZephyr	BM25		100	0.7420	0.3683	0.7086	0.4151
(2a) SPLADE++ ED	None		$ C $	0.7308	0.4464	0.7197	0.4826
(2b) RankZephyr	SPLADE++ ED		20	0.7780	0.4691	0.7900	0.5389
(2c) RankZephyr	SPLADE++ ED		100	0.7816	0.4908	0.8159	0.5754
(3a) RepLLaMA	None		$ C $	0.7384	0.4706	0.7195	0.4927
(3b) RankZephyr	RepLLaMA		20	0.7848	0.4785	0.7799	0.5274
(3c) RankZephyr	RepLLaMA		100	0.7660	0.4887	0.7802	0.5545
(4a) ADA ₂	None		$ C $	0.7035	0.4151	0.6759	0.4587
(4b) RankZephyr	ADA ₂		20	0.7633	0.4428	0.7568	0.5024
(4c) RankZephyr	ADA ₂		100	0.7573	0.4632	0.7757	0.5342

Table 6.4: nDCG@10 and MAP@100 for RankZephyr with different first-stage retrieval models. For each, reranking is performed using the top 20 or 100 candidates.

this gap can be partially ascribed to a 2% reduction in the number of judged documents in the top 10 rankings of the RepLLaMA reranked runs, there are likely additional factors as well.

We observe diminishing returns when RankZephyr reranks more effective candidate lists. For instance, reranking the top 100 BM25 candidates with RankZephyr leads to a substantial 45%–55% improvement in effectiveness across various metrics. In contrast, the improvement is more restrained when RankZephyr reranks SPLADE++ ED candidates, ranging from 7%–20% for the same set of metrics. This trend is consistent with observations in other multi-stage ranking systems [172, 174, 173]. A comparison between reranking the top 20 versus the top 100 candidates reveals that processing a larger pool generally leads to higher MAP@100 and, in most cases, also improves nDCG@10.

These findings help answer **RQ4**:

RQ4 Both the choice of first-stage retriever and the number of candidates affect downstream reranking effectiveness. Stronger first-stage retrievers yield substantially better results. Additionally, reranking larger candidate pools improves effectiveness.

6.3.5 Generalization to Novel Data

Given that many widely used benchmark datasets today were created years ago, there is a possibility that LLMs have already memorized knowledge specific to those queries due to

	nDCG@1	NovelEval nDCG@5	nDCG@10
(1) BM25	0.3333	0.4596	0.5577
(2) monoT5 (3B)	0.8333	0.7838	0.8462
(3) RankGPT _{3.5}	0.7619	0.7415	0.7571
(4) RankGPT ₄	0.8571	0.8749	0.9045
(5) RankZephyr	0.9286	0.8615	0.8934

Table 6.5: nDCG@{1,5,10} on the NovelEval-2306 set [207] for different reranking methods with Google Search as the first stage. Each reranker uses the top 20 retrieved results of the previous stage as input.

contamination of the training (or worse, test) data. To address this possibility, Sun et al. [207] proposed NovelEval, a continuously updated retrieval test set, ensuring that queries and candidate passages have not been seen during training.

Table 6.5 presents our results on NovelEval-2306, a test set with 21 questions from four domains, curated in June 2023. Sun et al. [207] also provided the top 20 candidates from Google Search that have been manually judged on a scale from 0 to 2. Since we are reranking only the top 20 results, which is the size of a single window, we do not employ progressive reranking. Additionally, we report metrics in the order the authors presented them, focusing on nDCG@1 because of their emphasis on high precision. Row (1) shows the results of BM25; row (2), monoT5 (3B) [158]; rows (3–4), RankGPT_{3.5} and RankGPT₄, and row (5), RankZephyr. We find that in terms of the primary metric, RankZephyr outperforms RankGPT_{3.5} and RankGPT₄, row (5) vs. (3) and (4).

To answer **RQ5**:

RQ5 We find that RankZephyr generalizes effectively to uncontaminated datasets not seen during training. This suggests RankZephyr has learned versatile ranking principles rather than memorizing training data.

6.4 Concluding Thoughts

Chapters 3–6 address the Component Challenge laid out in Chapter 1. In culmination, here, we carefully designed a two-stage distillation methodology and have shown that a smaller, open model can not only match but often surpass the effectiveness of much larger proprietary systems, thereby breaking the community’s dependence on black-box APIs for this critical task.

However, a critical question remains: does this distillation approach imply that the open-source community is destined to remain dependent on proprietary teachers like RankGPT₄? While RankZephyr relies on synthetic data from such models, we view this as a bootstrapping phase rather than a permanent dependency. When we built RankZephyr, the gap between proprietary and open-weight models was large, but over time these gaps have largely shrunk. To ensure sustainable progress, the field should adhere to higher standards of data democratization [60].

Many follow-up works have spun out of RankZephyr, each building on its success in distinct ways. For instance, in LiT5 [208], we replace RankGPT₄ as a teacher and distill listwise ranking capabilities of RankZephyr into a family of smaller T5-based encoder-decoder models, enabling more efficient reranking (e.g., handling 100 passages in a single forward pass). We open-sourced RankLLM [200], a Python toolkit for reproducible IR research, including all the pointwise, pairwise, and listwise rerankers discussed in this thesis. Schlatt et al. [199] similarly distills RankZephyr behavior down to the pointwise monoELECTRA from Chapter 4. Similarly, FIRST [188] addresses the inference inefficiencies of listwise LLM rerankers by using only the first output token’s logits (combined with a specialized learning-to-rank loss) to produce the ranking, achieving roughly a 50% speedup in inference while maintaining effectiveness. We independently reproduced and improved on these results in a generalization study [32]. Another offshoot is Rank-K [244], a 32B-parameter listwise reranker that incorporates an explicit chain-of-thought reasoning step at inference time, yielding further gains in generalization.

These developments were directly inspired by RankZephyr and its open-source accessibility, with each work focusing on a different facet — model compactness, inference speed, and reasoning abilities — to push the boundaries of zero-shot reranking. In particular, the advent of RankZephyr lowered the barrier to exploring such distillation capabilities (e.g., LiT5 deliberately uses RankZephyr instead of GPT₄ for teacher supervision), underscoring how a strong open reranker can catalyze a new wave of research. Together, these efforts highlight RankZephyr’s impact on the community, steering researchers towards more efficient, transparent, and powerful listwise reranking models, thus contributing directly to the democratization of state-of-the-art information access systems.

However, as argued in Chapter 1, progress depends on the synergistic development of highly effective and open models, better benchmarks and scalable evaluation frameworks. We now turn our attention to the equally critical Benchmarking and Evaluation Challenges: developing benchmarks and scalable methodology for assessing the long-form generative outputs that components like RankZephyr help to create.

Chapter 7

Ragnarök: A Framework for Benchmarking Generative Information Access

As the previous chapters have established, the advancement of generative information access hinges on overcoming the Component, the Benchmarking, and the Evaluation Challenge. While [Chapter 6](#) addressed the first by providing an open and powerful reranking component, progress in information access is impossible without a standardized ecosystem for executing and comparing RAG systems. This gives rise to the Benchmarking Challenge: the lack of a unified, open-source framework that integrates state-of-the-art components and provides the necessary resources for rigorous, reproducible RAG evaluation. Without such a foundation, comparing different systems becomes an ad hoc and unreliable process, slowing the pace of innovation.

This chapter addresses the Benchmarking Challenge by introducing Ragnarök [\[179\]](#), a reusable, end-to-end RAG framework designed to standardize evaluation and provide strong, reproducible baselines. Ragnarök serves as the backbone for the TREC 2024 Retrieval-Augmented Generation Track, providing the community with a common platform for building, testing, and analyzing RAG systems. It provides not only a modular software architecture but also the critical data resources—deduplicated collections and diverse, non-factoid topic sets—necessary for meaningful benchmarking. The following research questions guide our work:

- (RQ1) How can we design a modular RAG framework that enables reproducible execution and easy component substitution for community-wide benchmarking?

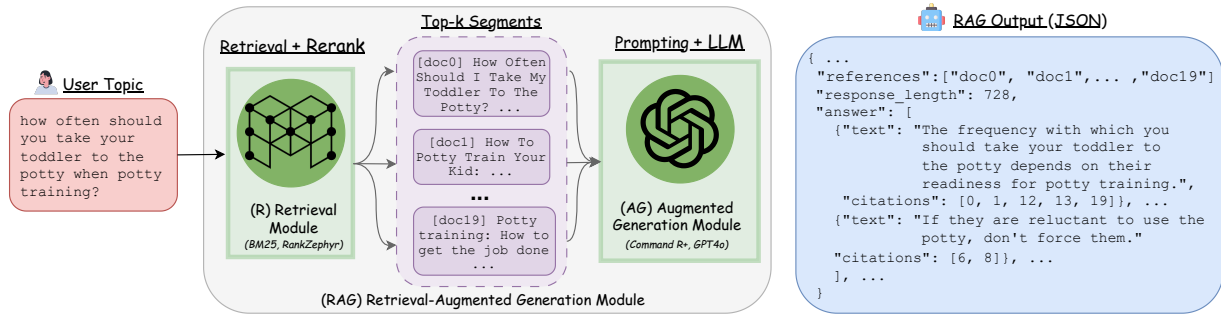


Figure 7.1: Schematic diagram of Ragnarök. Given a user topic (left), the process consists of two steps: retrieval + rerank (R), where the topic yields the top- k relevant segments from our document collection (e.g., potty training articles); and augmented generation (AG), where the retrieved segments with a suitable prompt template are fed to the LLM to generate the post-processed answer response (JSON) containing individual sentence-level citations.

- (RQ2) How can we best curate a large-scale document collection and diverse topic sets suitable for challenging RAG systems?
- (RQ3) What is the effectiveness of different retrieval pipelines on this new benchmark?
- (RQ4) How do different generation models compare in terms of answer characteristics and citation patterns?
- (RQ5) How can we enable community-driven evaluation of RAG systems through interactive comparison?

By answering these questions, this chapter details the creation of a comprehensive benchmarking ecosystem. We describe the Ragnarök framework, the curation of the MS MARCO V2.1 collection, the development of the TREC-RAGgy 2024 and TREC-Researchy 2024 topic sets, and the establishment of strong baseline systems so that the rest of the community can build. This work provides the essential infrastructure upon which the evaluation methodologies discussed in the next chapter build upon.

7.1 Our Framework

Ragnarök is an open-source, reproducible, and reusable framework implementing an end-to-end RAG pipeline, comprising two modules applied sequentially: retrieval (R) and aug-

mented generation (AG). Through Ragnarök, we provide several baselines to all participants in TREC RAG. An overview of the framework is provided in Figure 7.1. We first describe both modules and expand on the I/O specifications in our framework.

Retrieval Module. This module retrieves the relevant segments for a user topic as the input. It supports (i) first-stage lexical retrieval models such as BM25 [191] and (ii) reranking models such as RankZephyr [176]. The retrieval system searches for relevant segments in the document collection. It retrieves the top-100 segments, further reranked by the reranker model to filter out the top-20 relevant segments for the next stage.

Augmented Generation Module. This module takes in the user topic and the top-20 retrieved segments (from the retrieval module) as the input and a prompting strategy to the LLM to generate the answer response with in-context citations for the topic. The answer response is divided into individual sentences, each sentence within the answer contains text, and is grounded on retrieved documents provided as references to the LLM (if possible). We support most state-of-the-art proprietary models (GPT_{4o}, Gemini 2.5, etc.) through OpenRouter,¹ and open-source models (Command R+, LLaMA-3.1-70B-Instruct, etc.) through vLLM [104].²

7.1.1 RAG Input/Output Definitions

RAG Input. The input specifications are straightforward as the user can formulate any question they wish to ask, provide the user topic as input, and call Ragnarök.

RAG Output. The user receives a JSON output in response to their topic from Ragnarök. The first key in the output JSON schema, **references**, provides a list of segment IDs that are referenced in the answer. These segments are selected from among the top-20 results returned by the retrieval module. Next, **answer**, provides the LLM-generated RAG answer to the user topic, presented as a top-to-bottom list of sentence-level texts with corresponding segment citations. All citations are zero-based indexed, indicating the exact position of the segment ID from the **references** list. Finally, **response_length**, provides the total count of the whitespace-separated words present in the output RAG answer.

To answer RQ1:

¹<https://openrouter.ai>

²<https://github.com/vllm-project/vllm>

Collection	Version V2	Version V2.1 (Ours)
MS MARCO Document	11,959,635	10,960,555
MS MARCO Segment	124,131,414	113,520,750

Table 7.1: Comparison of document and segment counts between versions V2 and V2.1 (our version after removing near-duplicates) of the MS MARCO collection.

RQ1 We design Ragnarök as a two-stage modular pipeline with standardized REST APIs enabling seamless component substitution. The retrieval module supports multiple first-stage retrievers (BM25, dense encoders) and rerankers (monoT5-3B, RankZephyr), while the augmented generation module integrates diverse proprietary and open-source LLMs. Standardized I/O specifications — JSON inputs for topics and structured JSON outputs with references, sentence-level citations, and response metadata — ensure consistency across configurations, enabling reproducible community-wide evaluation.

7.2 Document Collection

The MS MARCO V2 document collection, previously used in the TREC-DL tracks, contains a substantial overlap of near-duplicates (documents with sufficiently similar text information) [44, 45]. When left intact, these near-duplicates degrade the downstream retrieval accuracy and reduce the diversity of the collected documents, potentially impacting the effectiveness of RAG systems.

Documents in the existing collection tend to be verbose, containing extensive information about individual topics. Chunking, which breaks down a long, verbose document into smaller, compact representations, is a key challenge, as the retrieved chunk representations correlate with the RAG answer quality [126].

MS MARCO V2.1 Document Collection. We conduct a deduplication strategy in the MS MARCO V2 document collection to avoid near-duplicates in two stages. In the first stage, we establish an equivalence class of the documents using Locality Sensitive Hashing (LSH) with MinHash [21] and 9-gram shingles. Next, we select a representative document for each equivalence class for our refined MS MARCO V2.1 document collection, reducing the duplicates in the original MS MARCO V2 document collection by 8.35% as shown in Table 7.1.

TREC-RAGgy 2024				TREC-Researchy 2024				
Topic	Category	%	First Word	%	Intrinsic Attributes	%	First Word	%
Aggregation		24.2	What	37.5	Knowledge-Intensive	79.8	How	41.0
Simple w/ cond.		23.3	How	27.5	Multi-Faceted	75.7	Why	25.5
Set		20.8	Why	3.3	Reasoning-Intensive	75.5	What	15.0
Simple		10.0	Is	2.5	Subjective	48.5	Is	5.2
Comparison		6.7	When	1.7	Assumptive	25.7	Should	2.2

Table 7.2: TREC-RAGgy 2024 and TREC-Researchy 2024 topic distribution. The table shows the top-5 categories in topic classification for TREC-RAGgy 2024 (each topic classified into a single category), intrinsic attributes for TREC-Researchy 2024, and the first word in all topics.

MS MARCO V2.1 Segment Collection. We segment the MS MARCO V2.1 document collection into overlapping segments (or chunks) and develop the MS MARCO V2.1 segment collection, with more than 113 million text segments (Table 7.1). We utilize a sliding window technique to generate the segments, by fixing the sliding window size of 10 sentences and a stride of 5 sentences to create each segment, roughly on average, between 500 and 1000 characters long. To easily map each segment back to the document, every segment contains the document ID within the segment ID. Further, two new fields: `start_char` and `end_char`, which indicate the start (inclusive) and the end position character (exclusive) of where the segment begins and ends in the mapped MS MARCO V2.1 document collection, respectively.

7.3 Topic Collection

Topics, i.e., user queries, are crucial for robust evaluation of RAG systems. Traditionally, popular retrieval and traditional QA benchmarks primarily consist of factoid queries, where answers are typically found within a single sentence or paragraph. However, these topics lack complexity, leading to short answers that can be easily memorized by LLMs. For instance, MS MARCO [13] surprisingly contains up to 55% factoid queries [16, 194]. To avoid short-form answers in RAG, we utilize two collections containing non-factoid topics covering diverse information and requiring long-form answers. We describe these collections below:

TREC-RAGgy 2024. We develop TREC-RAGgy 2024, a collection with topics filtered from TREC Deep Learning 2021-2023 tracks [43, 44, 45], based on topic category and generated-answer classification. We classify each available topic into seven categories and filter out a subset of topics that either have a long-form answer or require information aggregation across multiple sources of information. Out of the 210 original topics available, we filter and include 120 topics (57.1%) in the TREC-RAGgy 2024 collection. From Table 7.2, we observe 24.2% of the topics included are “aggregation”, indicating RAG systems should aggregate information from multiple segments to generate an accurate long-form answer. Additionally, 65% of the topics start with “what” or “how” questions.

The TREC-RAGgy 2024 collection includes mapped document-level relevance judgments from previous TREC Deep Learning tracks. These relevance judgments, associated initially with the full documents, have been mapped to the new deduplicated corpus, enabling effective evaluation of the retrieval systems. However, since most of our work operates at the segment level, prior to the evaluation, we perform a MaxP step — leveraging the score of the most relevant segment as the representative score of the document.

TREC-Researchy 2024. Researchy Questions, introduced in Rosset et al. [194], contains 102K non-factoid topics with long-form answers. These topics were curated from Bing Search logs and evaluated by GPT₄ on a scale of 0–10 based on eight intrinsic attributes, such as subjectivity and multifacetedness. Notably, unlike TREC-RAGgy 2024, these queries lack relevance judgments. To curate a smaller development subset for a faster evaluation of RAG systems, we employ a sampler designed to maximize diversity based on the eight intrinsic attributes. This is achieved by iteratively selecting the query with the highest l_1 norm in the intrinsic attribute space (of all eight dimensions) relative to the already-sampled set. We refer to the resultant topic set as TREC-Researchy 2024. From Table 7.2, about 80% of the topics are Knowledge-Intensive, and about 76% are Multi-Faceted, highlighting the need for effective RAG systems. Additionally, 66.5% of topics start with “how” or “why”, emphasizing explanatory questions. These distributions suggest that TREC-Researchy 2024 prioritizes complex and multi-dimensional topics.

For the TREC RAG test topics, we released a new and fresh scrape of topics close to the submission period. This approach compiled a fresh and newer set of topics, similar to Rosset et al. [194], thereby minimizing the risk of data leakage and ensuring a fair evaluation with existing commercially available LLMs.

In response to **RQ2**:

RQ2 We curate resources through a multi-faceted approach: (i) deduplicating and segmenting a large-scale document collection to produce MS MARCO V2.1; (ii) filtering

traditional factoid benchmarks to create non-factoid topic sets requiring information aggregation; and (iii) sampling diverse topics from search logs to maximize complexity and multi-facetedness. By releasing fresh topic scrapes and emphasizing long-form topics, we create challenging evaluation conditions that mitigate test collection overfitting while better reflecting real-world information needs.

7.4 TREC RAG Baselines

7.4.1 Retrieval

Our retrieval module integrates both first-stage retrievers and rerankers. For traditional dual encoders, we employ BM25 and BM25 + Rocchio, available in Anserini [245] and retrieve the top 3000 segments for a given topic. BM25 has proven effective as a first-stage retriever due to its capability to capture lexical overlap, with Rocchio introducing relevance feedback to further refine the search results based on initial BM25 retrieval.

We further extend our retrieval evaluation by incorporating GTE-L, a dense dual encoder model from Alibaba-NLP [113]. GTE-L is their larger variant comprising 434M parameters, with embeddings of dimensionality 1024, and support for a maximum sequence length of 8196 tokens. This model is particularly suited for long-context retrieval tasks, enabling it to effectively capture relationships across larger text segments. GTE-L is a strong baseline demonstrating state-of-the-art effectiveness in the MTEB [147] benchmark, when it was introduced.

Additionally, we evaluate two other dense dual encoders from Snowflake: ArcticEmbed-M and ArcticEmbed-L [142]. ArcticEmbed-M is a 137M parameter model, generating embeddings with a dimensionality of 768, while ArcticEmbed-L is a larger model with 335M parameters and an embedding dimensionality of 1024. Both models have shown state-of-the-art performance in retrieval tasks, particularly on the BEIR benchmark [213], where they excel in diverse retrieval settings. ArcticEmbed-M and ArcticEmbed-L provide efficient and scalable solutions for semantic retrieval, complementing our first-stage retrieval suite.

For each retrieval method (BM25, BM25 + Rocchio, GTE-L, ArcticEmbed-M, and ArcticEmbed-L), we retrieve the top 3000 segments. To leverage the strengths of multiple retrieval strategies, we employ Reciprocal Rank Fusion (RRF) as a hybrid approach. The RRF technique fuses the retrieval results from BM25 + Rocchio, GTE-L, ArcticEmbed-M, and ArcticEmbed-L, with the aim of producing a better candidate set for reranking, as

hybrid approaches have been shown to improve retrieval effectiveness in multiple TREC tracks [106].

Following retrieval, we explore neural reranking models for first-stage reranking. We leverage monoT5-3B, the pointwise reranker with 3B parameters from Chapter 3, that demonstrates strong effectiveness in reranking by assigning relevance scores to individual document-query pairs [158, 172]. Note that we fuse these results with the results from the prior stage.

Next, we incorporate RankZephyr from Chapter 6, a highly effective listwise reranker, which takes as input a query and a list of passages and outputs a reordered list based on relevance. RankZephyr is particularly effective in scenarios where contextual relationships between documents are useful for ideal reranking. In our pipeline, we use RankZephyr to rerank the top 100 resultant candidates from monoT5-3B, ensuring a more precise final ranking. Note that we fuse the RankZephyr results with the results from the prior stage.

Additionally, we evaluate RankZephyr _{ρ} , which adopts a progressive reranking strategy. RankZephyr _{ρ} iteratively refines the ranking of candidate documents over three passes, progressively improving the precision of the final ranked list. This iterative refinement has been shown to enhance retrieval effectiveness, particularly in cases where high-precision ranking is required.

Both monoT5-3B and RankZephyr, and also other rerankers such as LRL [135], RankGPT [207], and LiT5 [208], are available through the `rank_llm` package, which we leverage as part of our reranking pipeline in Ragnarök. While budget constraints prevent us from testing every reranking model, support for this functionality continues through Ragnarök. Some of these retrieved results in the case of TREC-RAGgy 2024 can be evaluated after running the MaxP operation to get the representative score for the document. Finally, the top-20 reranked segments are passed on to the next stage, i.e., augmented generation.

Results. Table 7.3 presents the nDCG@10, MAP@100, and recall@100 for the primary retrieval baselines under consideration. The traditional lexical retrievers BM25 (1a) and BM25+Rocchio (1b) demonstrate comparable scores with only marginal differences. Dense dual encoders (1c, 1d, 1e) exhibit superior effectiveness compared to traditional lexical methods (1a, 1b). Among the dense models, the ArcticEmbed variants (1d, 1e) achieve slightly higher effectiveness metrics than GTE-L (1c). Notably, the application of Reciprocal Rank Fusion across these models (1f) yields substantial improvements compared to any individual retriever (1a–e).

Model	nDCG@10	MAP@100	Recall@100
<i>Lexical & Dual Encoders</i>			
(1a) BM25	0.4227	0.1561	0.2807
(1b) BM25+Rocchio	0.4188	0.1818	0.3141
(1c) GTE-L	0.5682	0.2162	0.3512
(1d) ArcticEmbed-M	0.5749	0.2349	0.3692
(1e) ArcticEmbed-L	0.5776	0.2277	0.3623
(1f) RRF(1b,1c,1d,1e)	0.6064	0.2592	0.3990
<i>Rerankers</i>			
(2a) RRF(1f, monoT5-3B)	0.6175	0.2708	0.4208
(2b) RRF(2a, RankZephyr)	0.6357	0.2770	0.4208
(2c) RRF(2a, RankZephyr _{ρ})	0.6317	0.2771	0.4208

Table 7.3: Results on the Document Ranking Task of the TREC-RAGgy 2024.

The integration of the pointwise reranker monoT5-3B (2a) further enhances retrieval effectiveness across all metrics. Subsequent application of listwise reranking models (2b, 2c) yields additional effectiveness gains. However, the progressive reranking approach does not demonstrate meaningful improvements in high-precision metrics for this particular collection, as evidenced between the multiple and single-pass RankZephyr variants (2c vs. 2b).

In summary, to answer **RQ3**:

RQ3 We find that dual encoders substantially outperform lexical methods, while hybrid first-stage retrieval approaches achieve the best results. Additionally, the multi-stage ranking pipeline, first with monoT5 and then with RankZephyr further improves the scores.

7.4.2 Augmented Generation

Our generation module is designed to support a wide range of both open-source and proprietary models, enabling flexible integration for various use cases. At the core of our open-source support is vLLM [104], an optimized framework for efficient LLM inference. vLLM excels in handling resource-intensive models by leveraging tensor parallelism and dynamic batching, ensuring scalability and performance across distributed systems. This

```
System: This is a chat between a user and an artificial intelligence
assistant. The assistant gives helpful, detailed, and polite answers to the
user's questions based on the context. The assistant should also indicate
when the answer cannot be found in the context.
```

```
INSTRUCTION: Please give a complete answer to the question. Cite each
context document that supports your answer within brackets [] using the
IEEE format.
```

```
QUESTION: {query}
```

```
CONTEXTS:
```

```
[1] {Passage title}: {Passage text}
```

```
[2] {Passage title}: {Passage text}
```

```
...
```

```
[20] {Passage title}: {Passage text}
```

```
INSTRUCTION: Please give a complete answer to the question. Cite each
context document that supports your answer within brackets [] using the
IEEE format.
```

Figure 7.2: ChatQA prompt template [130] used for RAG generation with in-text citations with LLaMA3.1-70B and GPT_{4o} in our Ragnarök framework.

makes it well-suited for open-source LLMs, where efficient resource management is critical. In addition, vLLM provides a wide support for text-based LLMs³, including custom models fine-tuned with LoRA [84].

Our focus is on three popular LLMs: (i) GPT_{4o} is the latest GPT version from OpenAI [160]; (ii) Command R+ is Cohere’s open-source instruction-following LLM developed for complex RAG pipelines [38]; (iii) LLaMA-3.1-70B-Instruct is Meta’s flagship open-source instruction-tuned generative model optimized for multilingual dialogue use cases [52]. Given that Command R+ cites in a span level, we map the citations to their parent sentences. For GPT_{4o} and LLaMA-3.1-70B, we follow the ChatQA prompt template [130] and cite relevant segments within the text (in-line) using the IEEE format. The prompt template is shown in Figure 7.2.

Additionally, Ragnarök incorporates a variety of refined ChatQA prompts specifically tailored for biomedical RAG tasks, particularly within the context of the TREC 2024

³https://docs.vllm.ai/en/latest/models/supported_models.html

Biomedical Generative REtrieval (BioGen) track. These refinements enhance the system’s ability to address domain-specific queries effectively in the expected manner and steer the model with expected word counts and sentence structures (each sentence is an independent assertion). Furthermore, the foundational ChatQA template was repurposed for the TREC 2024 Neural Cross-Language Information Retrieval (NeuCLIR) track’s report generation task. This adaptability underscores Ragnarök’s versatility in prompt management, allowing users to modify prompts with minimal intervention, typically requiring only a single addition within the prompt template file. This streamlined approach ensures that users can efficiently adapt our framework across various domains.

7.4.3 RAG-Bench Evaluation

Evaluating different RAG answers is challenging as multiple factors within the output response are crucial for effectiveness evaluation. To combat this, recent works rely on an LLM-as-a-judge setup [259], where strong LLM assessors judge the RAG-generated output in a pairwise evaluation style (side-by-side) in a head-on tournament. In our work, we briefly overview our baseline techniques using human evaluators.

This provides us with the answer to **RQ4**:

RQ4 The Command R+ baseline outputs shorter answers and cites more relevant segments, whereas the GPT_{4o} baseline outputs longer and more detailed answers and cites fewer segments. Therefore, for topics in both TREC-RAGgy 2024 and TREC-Researchy 2024, GPT_{4o} intuitively is the better choice for RAG answer generation. We found that the LLaMA3.1-70B model produces long and detailed answers, but sometimes generates a separate references section, from which it cites.

7.5 Ragnarök System Arena

Heavily inspired by the success of Chatbot Arena [34, 259], a crowdsourcing benchmark WebUI featuring anonymous battles, we extend the concept to multi-stage configurable RAG pipelines with Ragnarök. In the arena, users interact with two unblinded/blinded RAG systems simultaneously, issuing the same topic to both. The participants evaluate and select the pipeline that delivers their most preferred response, with the identities of the modules in the end-to-end pipeline revealed after the voting process in the blinded case. We leverage Gradio [1] to build the WebUI for Ragnarök. Each step of the pipeline uses REST

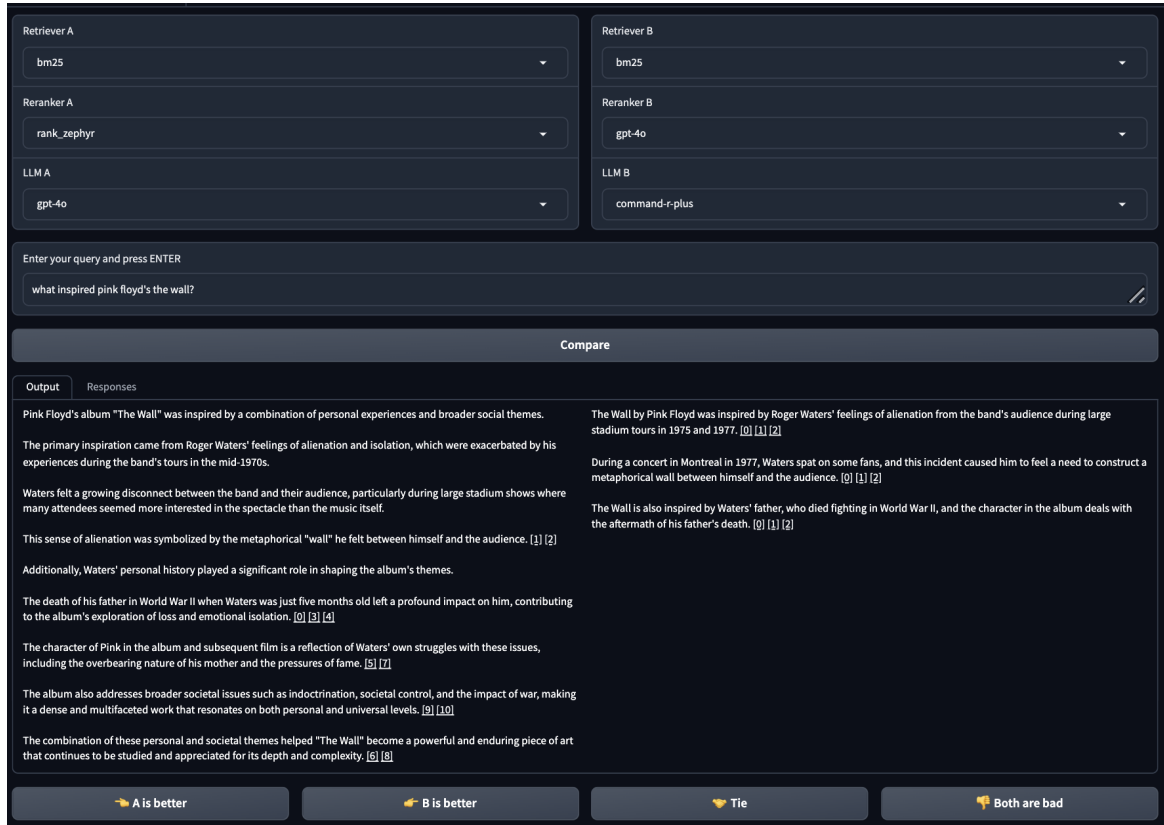


Figure 7.3: WebUI showcasing the Ragnarök System Arena and the user query, “what inspired pink floyd’s the wall?”, with answers from two pipelines side-by-side comparing GPT_{4o} (left) and Command R+ (right).

APIs for intercommunication, enabling easy module switching within the pipeline. This modular design simplifies the integration of different retrieval and LLM configurations, enhancing scalability and maintainability.

Figure 7.3 illustrates an example topic “what inspired pink floyd’s the wall?” processed by two different pipelines: Pipeline A, comprising BM25 → RankZephyr → GPT_{4o} (left), and Pipeline B, comprising BM25 → RankGPT_{4o} → Command R+ (right) in the unblinded tab. The outputs generated by each pipeline are compared, allowing users to discern which system provided a more satisfactory answer. Note that when the user hovers the mouse over a citation, they can preview the cited segment. We also provide a similar blinded pairwise evaluation interface along with the ability to view the responses in the JSON form (a tab) in the WebUI for Ragnarök.

After reading both outputs, the user has to pick one of the four choices for preference scoring: (i) A is better, (ii) B is better, (iii) Tie, or (iv) Both are bad. These results from preference scoring are organized into three separate Elo leaderboards, leveraging an SQLite database to track module effectiveness. These leaderboards evaluate R, AG, and RAG pipelines, providing a comprehensive view of their relative contributions.

To summarize, we answer **RQ5**:

RQ5 We develop the Ragnarök System Arena, a Gradio-based WebUI that allows users to interact with two RAG pipelines simultaneously in blinded or unblinded modes. Users issue the same topic to both systems, compare generated responses side-by-side with hoverable citation previews, and select their preference. These preference scores feed into three separate Elo leaderboards tracked via an SQLite database, evaluating LLMs (AG-only), retrievers (R), and full RAG pipelines, enabling transparent community-driven assessment of component effectiveness

7.6 Concluding Thoughts

This chapter has detailed the development of Ragnarök, a comprehensive, open-source ecosystem designed to address the Benchmarking Challenge outlined in [Chapter 1](#). By creating a standardized framework, curating high-quality data resources, and establishing strong performance baselines, we have provided the essential infrastructure for the rigorous and reproducible evaluation of RAG systems. Ragnarök furnishes the community with the tools to not only build and test complex generative information access systems but also to understand their behavior through a common, shared platform.

The creation of the MS MARCO V2.1 collection, along with the TREC-RAGgy 2024 and TREC-Researchy 2024 topic sets, represents a critical contribution to the field, offering a robust alternative to existing benchmarks that may suffer from contamination or a lack of complexity. The baselines established using state-of-the-art retrievers, rerankers, and generators serve as a valuable foundation for future research. As a result of this, we have successfully run two years of TREC RAG tracks, as well as made several state-of-the-art submissions to other tracks with an easy-to-spin pipeline.

Having established this, the narrative of this thesis now switches to the equally critical task of *evaluating* model responses. With the infrastructure from Ragnarök in place, we can now turn our full attention to the Evaluation Challenge: how do we reliably and at scale measure the quality of long-form, generated answers? The next chapter introduces the

AutoNuggetizer framework, a direct attempt to solve this problem by automating nugget-based recall evaluation, building directly upon the benchmarks and systems described here.

Chapter 8

AutoNuggetizer: A Framework for Evaluating Generative Information Access

The shift from document retrieval to generative information access, as discussed in [Chapter 1](#), presents a profound Evaluation Challenge. Traditional IR metrics, designed for ranked lists of documents, are ill-equipped to directly assess long, multi-faceted answers generated by RAG systems. While human evaluation remains the gold standard, it is prohibitively expensive and slow, creating a significant bottleneck that hampers research progress. The urgent need for reliable and scalable evaluation methodologies is one of the most pressing issues facing the field today.

This chapter attempts to address the Evaluation Challenge by introducing the AutoNuggetizer framework [\[180\]](#), which refactors a classic and well-studied evaluation methodology for the modern era of LLMs. Drawing inspiration from the nugget-based evaluation of the TREC Question Answering (QA) track [\[229\]](#), AutoNuggetizer automates the assessment of information nugget recall in RAG answers. AutoNuggetizer was validated at scale in the TREC 2024 RAG Track [\[179\]](#), demonstrating that LLMs can be effectively grounded in well-established evaluation principles to provide reliable assessments. The following research questions guide our investigation:

- (RQ1) How does a fully automated nugget-based evaluation framework, where an LLM performs both nugget creation and assignment, compare to a fully manual or semi-manual framework?

Query: how taylor swift’s age affects her relationships

Answer: Taylor Swift’s age has played a significant role in her relationships, often influencing public perception and the dynamics of her romances. Throughout her dating history, Swift has been involved with partners of varying ages, demonstrating that she does not discriminate based on age. [. . .] In summary, Taylor Swift’s age has influenced her relationships in various ways, from public scrutiny over age differences to the emotional experiences that have fueled her songwriting. Her willingness to date partners of different ages, both younger and older, demonstrates her openness in her love life, despite the challenges and criticisms that come with it.

Table 8.1: Sample answer from the TREC 2024 RAG Track for topic 2024-79081. Note that here we have purposely omitted citations in the answer since this thesis does not cover support evaluation. The answer has been elided for length considerations.

- (RQ2) What is the effectiveness where only the nugget assignment step is automated, offering a potential balance between cost and quality?
- (RQ3) What are the qualitative differences between human and LLM-based nugget assignment, and what do these differences reveal about the capabilities and biases of LLM judges?

By answering these questions, this work provides an in situ-validated, scalable framework for RAG evaluation that offers flexible trade-offs between automation and human effort, establishing a foundation to hill-climb on.

It is important to emphasize that this work focuses exclusively on evaluating the recall of information nuggets within RAG answers. There are, of course, other important aspects to the Evaluation Challenge, such as the assessment of citation support and the fluency of answers. These are important to the overall evaluation picture, but beyond the scope of this report.

8.1 The TREC 2024 RAG Track

The validation of the AutoNuggetizer framework was conducted within the context of the TREC 2024 RAG Track, a large-scale, community-wide evaluation campaign designed to assess the state of the art in RAG. The track, conducted with NIST, provided a public, standardized setting to test our methodology against extensive human judgments.

The track leveraged the MS MARCO V2.1 Segment collection and featured 301 fresh queries gathered from the Bing search logs, which we introduced in [Chapter 7 \[179\]](#). These queries were designed to be long-form and multi-faceted, requiring systems to synthesize information from multiple documents. A sample query and response can be found in [Table 8.1](#).

We defined three primary tasks for the TREC 2024 RAG Track:

- **Retrieval (R):** Return a ranked list of relevant passages from the MS MARCO V2.1 corpus for each query.
- **Answer Generation (AG):** Generate a comprehensive and well-cited answer using a fixed set of 100 provided passages. This task isolates the generation capability of the LLMs.
- **Retrieval-Augmented Generation (RAG):** Generate a comprehensive and well-cited answer using passages retrieved freely from the entire corpus.

The R task is not the focus of this report; interested readers are directed to Upadhyay et al. [\[221\]](#). However, it is worth noting that, for retrieval evaluation, documents were judged on a scale from 0 to 3, based on how relevant they are to the query. This was also conducted with varying levels of automation using UMBRELA [\[222\]](#).

8.2 The AutoNuggetizer Framework

The AutoNuggetizer framework operationalizes the classic nugget-based evaluation methodology by leveraging LLMs to automate the core tasks of nugget creation and assignment. The framework is designed to be flexible, allowing researchers to choose the level of automation that best suits their needs.

8.2.1 Nugget Creation

Automatic Nugget Creation

The AutoNuggetizer framework begins by extracting a list of nuggets, or atomic information units, from a set of input documents. This nugget creation process, dubbed “nuggetization”, characterizes the information that should be contained in a high-quality answer to the user query.

SYSTEM: You are NuggetizeLLM, an intelligent assistant that can update a list of atomic nuggets to best provide all the information required for the query.

USER: Update the list of atomic nuggets of information (1-12 words), if needed, so they best provide the information required for the query. Leverage only the initial list of nuggets (if exists) and the provided context (this is an iterative process). Return only the final list of all nuggets in a Pythonic list format (even if no updates). Make sure there is no redundant information. Ensure the updated nugget list has at most 30 nuggets (can be less), keeping only the most vital ones. Order them in decreasing order of importance. Prefer nuggets that provide more interesting information.

Search Query: {query}

Context:

[1] {seg₁}
[2] {seg₂}
...
[10] {seg₁₀}

Search Query: {query}

Initial Nugget List: {n_{i-1}}

Initial Nugget List Length: {len(n_{i-1})}

Only update the list of atomic nuggets (if needed, else return as is). Do not explain. Always answer in short nuggets (not questions). List in the form ["a", "b", ...] and a and b are strings with no mention of ".

Updated Nugget List:

LLM: [n_{i,1}, ...]

Figure 8.1: Prompt for the iterative nuggetization at turn i .

We use GPT_{4o} as our LLM of choice for all experiments. This process is run over all documents that are judged to be at least “related” to the query (grade ≥ 1). Note that, depending on the actual evaluation condition, these relevance judgments (qrels) are either provided by NIST assessors (manual) or by UMBRELA [222] (automatic). Examples of these differences are shown in Table 8.2, top vs. middle.

Using the prompt in Figure 8.1, the LLM iteratively updates a list of nuggets that collectively represent key facts required to fully answer the query, conditioned on the provided passages. The first iteration for each query begins with an empty list. This iterative approach allows for the generation of a rich set of nuggets, capturing both explicit and implicit information requirements derived from the query. Informed by previous implementations of nugget evaluations, we limited generation to at most 30 nuggets.

Once we have generated a set of nuggets for a query, the next step is to assess the importance of each nugget. We again use the LLM to assign an importance label of either

Automatic nugget creation (AutoNuggets) using UMBRELA qrels
Swift’s naivety was taken advantage of by older partners (vital)
Age gaps have strained Swift’s relationships (vital)
Swift dated John Mayer at 19, he was 32 (vital)
Swift dated Jake Gyllenhaal at 20, he was 29 (vital)
Swift’s relationships often end in songs (okay)
Automatic nugget creation (AutoNuggets) using NIST qrels
Swift’s age gaps in relationships are significant (vital)
Swift’s maturity level impacted her relationships (vital)
Swift’s relationships often involve commitment issues (vital)
Swift’s relationships often involve learning experiences (vital)
Swift’s relationships often end in songs (okay)
Post-edited nuggets (AutoNuggets+Edits) using NIST qrels
Swift’s maturity level impacted her relationships (vital)
Swift’s relationships often involve personal growth (vital)
Swift’s relationships often involve prioritizing friends and family (vital)
Swift’s heartbreaks influenced her music (vital)
Swift’s relationships often involve learning experiences (okay)

Table 8.2: Comparison of five sample nuggets for the query “how Taylor Swift’s age affects her relationships” across three conditions: AutoNuggets using UMBRELA qrels, AutoNuggets using NIST qrels, and AutoNuggets+Edits (i.e., human post-edited nuggets).

“vital” or “okay” using the prompt shown in [Figure 8.2](#). At this point, we sort the nuggets in descending order of importance and select the first 20 nuggets. The resulting nugget set we dub AutoNuggets.

SYSTEM: You are NuggetizeScoreLLM, an intelligent assistant that can label a list of atomic nuggets based on their importance for a given search query.

USER: Based on the query, label each of the $\{\text{len}(n_f)\}$ nuggets either a vital or okay based on the following criteria. Vital nuggets represent concepts that must be present in a “good” answer; on the other hand, okay nuggets contribute worthwhile information about the target but are not essential. Return the list of labels in a Pythonic list format (type: `List[str]`). The list should be in the same order as the input nuggets. Make sure to provide a label for each nugget.

Search Query: $\{\text{query}\}$

Nugget List: $\{n_f\}$

Only return the list of labels (`List[str]`). Do not explain.

Labels:

LLM: `["vital", "okay", ...]`

Figure 8.2: Prompt for determining the importance of nuggets. At each turn, at most 10 nuggets are passed to the LLM.

Semi-Manual Nugget Creation

In the condition that we denote as AutoNuggets+Edits, NIST assessors post-edit nuggets that have been proposed by AutoNuggetizer as a “rough draft”. Here, we start with the set of documents that NIST assessors have manually judged to be at least “related” to the query (grade ≥ 1), which serves as the input to AutoNuggetizer to create nuggets automatically (per above).

The automatically generated nuggets that we provide to NIST assessors are prepared in a slightly different way: for each query, we created a set of 30 nuggets without any indication of importance. These nuggets are then edited by NIST assessors, who may add, eliminate, or combine nuggets. The NIST assessors perform this task by concurrently considering the list of relevant documents for that topic. Based on our logs, this process takes roughly an hour per topic on average, which suggests that the NIST assessors are not merely “rubber-stamping” automatically generated nuggets, but are actually carefully deliberating over how they are formulated. An example of this process is shown at the bottom of [Table 8.2](#).

Manual Nugget Creation

In the condition that we denote as ManualNuggets, NIST assessors examine the documents that have been manually judged to be at least “related” to the query (grade ≥ 1) to create

SYSTEM: You are NuggetizeAssignerLLM, an intelligent assistant that can label a list of atomic nuggets based on if they are captured by a given passage.

USER: Based on the query and passage, label each of the $\{\text{len}(n_f)\}$ nuggets either as support, partial_support, or not_support using the following criteria. A nugget that is fully captured in the passage should be labeled as support. A nugget that is partially captured in the passage should be labeled as partial_support. If the nugget is not captured at all, label it as not_support. Return the list of labels in a Pythonic list format (type: List[str]). The list should be in the same order as the input nuggets. Make sure to provide a label for each nugget.

Search Query: {query}

Passage: {p}

Nugget List: {n_f}

Only return the list of labels (List[str]). Do not explain.

Labels:

LLM: ["support", "not_support", "partial_support", ...]

Figure 8.3: Prompt for nugget assignment. At each turn, at most 10 nuggets are passed to the LLM.

a set of at most 20 nuggets with their importance. This followed the same procedure as the TREC 2003 QA Track. Manual nugget creation is known to be a time-consuming and labor-intensive task. Each topic required substantial assessor effort to analyze the relevant passages thoroughly and to synthesize a concise set of nuggets. This took roughly 2.5 hours per topic, based on our measurements.

8.2.2 Nugget Assignment

Automatic Nugget Assignment

The final component of our AutoNuggetizer framework, AutoAssign, automatically assigns nuggets to systems' answers. We adopt a listwise approach to nugget assignment, where the LLM is used to analyze an answer and determine if each nugget is fully supported (**support**), partially supported (**partial_support**), or not supported (**not_support**) by the answer. Again, we employ GPT_{4o} through the Azure endpoint, with the prompt shown in [Figure 8.3](#). We iteratively prompt the model with at most 10 nuggets to evaluate the assignment at each step.

Nuggets	Assignment
Automatic nuggets (AutoNuggets) using UMBRELA qrels	
Swift’s naivety was taken advantage of by older partners (vital)	Not Support
Age gaps have strained Swift’s relationships (vital)	Support
Swift dated John Mayer at 19, he was 32 (vital)	Not Support
Swift dated Jake Gyllenhaal at 20, he was 29 (vital)	Not Support
Swift’s relationships often end in songs (okay)	Support
Post-edited nuggets (AutoNuggets+Edits) using NIST qrels	
Swift’s maturity level impacted her relationships (vital)	Partial Support
Swift’s relationships often involve personal growth (vital)	Not Support
Swift’s relationships often involve prioritizing friends and family (vital)	Not Support
Swift’s heartbreaks influenced her music (vital)	Support
Swift’s relationships often involve learning experiences (okay)	Not Support

Table 8.3: Comparison of nugget assignment across different nugget creation approaches.

Manual Nugget Assignment

In the manual nugget assignment process, we leveraged the expertise of the original NIST assessor involved in creating the list of nuggets (i.e., the same assessor performed nugget creation *and* nugget assignment). The assessor examines each answer, assigning each nugget one of the three labels (same as with automatic assignment). It is worth clarifying that assessors in this step are not shown any LLM output, so this is a fully manual process.

An example is shown in [Table 8.3](#): on the top, we show fully automatic assignment (AutoAssign) of automatically created nuggets (AutoNuggets) using UMBRELA relevance judgments (qrels); on the bottom, we show manual assignment (ManualAssign) of post-edited nuggets (AutoNuggets+Edits) using NIST relevance judgments. The answer being evaluated is the one shown in [Table 8.1](#).

8.2.3 Evaluation Strategies

Within our general AutoNuggetizer framework, we can instantiate a number of variants combining different design choices for nugget creation (Section 8.2.1) and nugget assignment (Section 8.2.2). In this work, we considered the following combinations:

1. Automatic nugget creation with manual post-editing and manual nugget assignment (AutoNuggets+Edits / ManualAssign). Nuggets are initially generated automatically from passages judged relevant by NIST assessors and then refined by human assessors. Nugget assignment is performed manually by NIST assessors. This represents semi-manual nugget creation with manual assignment.
2. Manual nugget creation and assignment (ManualNuggets / ManualAssign). NIST assessors manually create nuggets from relevant passages and manually assign them to system answers. This is a fully manual evaluation baseline, mirroring the original TREC QA approach.
3. Fully automatic nugget creation and assignment (AutoNuggets / AutoAssign). Nuggets are automatically generated from documents assessed as relevant by UMBRELA, and nugget assignment is performed automatically using AutoAssign. This represents a fully automated, end-to-end evaluation.
4. Semi-manual nugget creation with automatic nugget assignment. In this condition, we use the nuggets from conditions (1) and (2), which involve human assessors in nugget creation (AutoNuggets+Edits and ManualNuggets, respectively). However, we replace manual nugget assignment with automatic nugget assignment using AutoAssign, i.e., AutoNuggets+Edits / AutoAssign and ManualNuggets / AutoAssign, respectively. This condition isolates the impact of automating only the nugget assignment step.

8.2.4 Scoring

By now, we have already completed nugget creation and nugget assignment. For each query, we have a list of nuggets, and for each system answer, we have a record of which nuggets it contains, in terms of a three-way judgment: **support**, **partial_support**, and **not_support** (either manual or automatic nugget assignment).

The final step is to compute the score for a system’s answer to a query q . The score of a run is simply the mean of the scores for each query. We compute the following scores per query:

Condition		# topics	avg nuggets	avg nugget length	% nuggets vital okay		% assigned NS PS S		
(1a)	AutoNuggets+Edits / ManualAssign	36	14.1	7.0	61.1	38.9	57.9	6.6	35.6
(1b)	ManualNuggets / ManualAssign	20	13.7	8.5	59.0	41.0	62.9	3.5	33.6
(2a)	AutoNuggets / AutoAssign (1a subset)	36	18.2	6.8	66.2	33.8	50.9	22.0	27.0
(2b)	AutoNuggets / AutoAssign (1b subset)	20	18.7	7.2	70.3	29.7	47.1	24.9	28.0
(3a)	AutoNuggets+Edits / AutoAssign	36	14.1	7.0	61.1	38.9	54.1	18.7	27.2
(3b)	ManualNuggets / AutoAssign	20	13.7	8.5	59.0	41.0	57.8	17.6	24.6
(4)	AutoNuggets / AutoAssign (Overall)	301	18.7	6.8	72.5	27.5	50.9	23.6	25.5

Table 8.4: Descriptive statistics for nugget creation and assignment under different conditions.

All Strict (A_{strict}) We calculate a score based on all nuggets in an answer, but with strict nugget matching. For a given nugget i :

$$p_i = \begin{cases} 1 & \text{if assignment} = \text{support} \\ 0 & \text{otherwise} \end{cases} \quad (8.1)$$

The “All Strict” score is then calculated as:

$$A_{\text{strict}} = \frac{\sum_i p_i}{N_{\text{nuggets}}}$$

Vital Strict (V_{strict}) This score, used as the primary evaluation metric, applies the same strict matching criterion as p_i above, but only over the vital nuggets, p_i^v . The score is calculated as:

$$V_{\text{strict}} = \frac{\sum_i p_i^v}{N_{\text{vital nuggets}}}$$

We emphasize that the scores presented here focus exclusively on evaluating the recall of nuggets in RAG answers. This analysis does not encompass other crucial aspects of RAG evaluation, such as the assessment of citation support or fluency, which are essential but outside the scope of this report.

8.3 Results

8.3.1 Descriptive Statistics

We begin with some general statistics for the TREC 2024 RAG Track, NIST received 93 runs from 20 groups for the RAG task and 53 runs from 11 groups for the AG task.

Given resource constraints, the NIST annotators were able to evaluate only the two highest priority submissions from each group across the RAG and AG tasks, which translates into 31 runs from 18 groups for RAG and 14 runs from 9 groups for AG.

To be clear, our analyses are over 56 topics that have been entirely judged across the 31 RAG runs and 14 AG runs discussed above. Of these, 36 topics were under the AutoNuggets+Edits / ManualAssign condition and 20 topics were under the ManualNuggets / ManualAssign condition. Descriptive statistics for nugget creation and assignment are presented in [Table 8.4](#), broken down by the various experimental conditions. Naturally, since condition (3) in [Section 8.2.3](#), corresponding to row (4) in the table, is fully automated, we were able to evaluate all topics, whereas for the other conditions we were constrained by available human resources.

Examining the nugget characteristics, we observe that conditions employing AutoNuggets, rows (2a), (2b), and (4), tend to generate a higher average number of nuggets per topic (around 18) compared to conditions using AutoNuggets+Edits or ManualNuggets (around 14). Interestingly, the average nugget length remains relatively consistent across all conditions, hovering around 7–8 tokens, suggesting a stable granularity in nugget extraction regardless of the automation level in nugget creation. Furthermore, humans tend to be more conservative in assigning the label of “vital” importance, likely due to a stricter evaluation of what constitutes essential information within the context of a topic.

Focusing on nugget assignment, a clear distinction emerges between manual and automatic assignment. Conditions employing manual assignment, (1a) and (1b), exhibit a substantially lower percentage of nuggets categorized as “Partial Support”. In contrast, the automatic assignment conditions, especially the directly comparable ones, (3a) and (3b), see higher percentages. This shift suggests that automatic assignment tends to distribute some human-deemed “Support” assignments to “Partial Support”, potentially indicating a different interpretation or application of support criteria compared to manual assignment. However, this behavior could simply be the result of the current prompt that we are using.

8.3.2 Fully Automatic Nugget Evaluation

To address **RQ1**, we examine the correlation between our fully automatic nugget evaluation methodology and variants that require manual intervention to various degrees, as detailed in conditions (1) and (2) in [Section 8.2.3](#). We reiterate that the AutoNuggets / AutoAssign condition is fully automatic, requiring no manual intervention (specifically, nuggetization is based on automatically generated relevance labels).

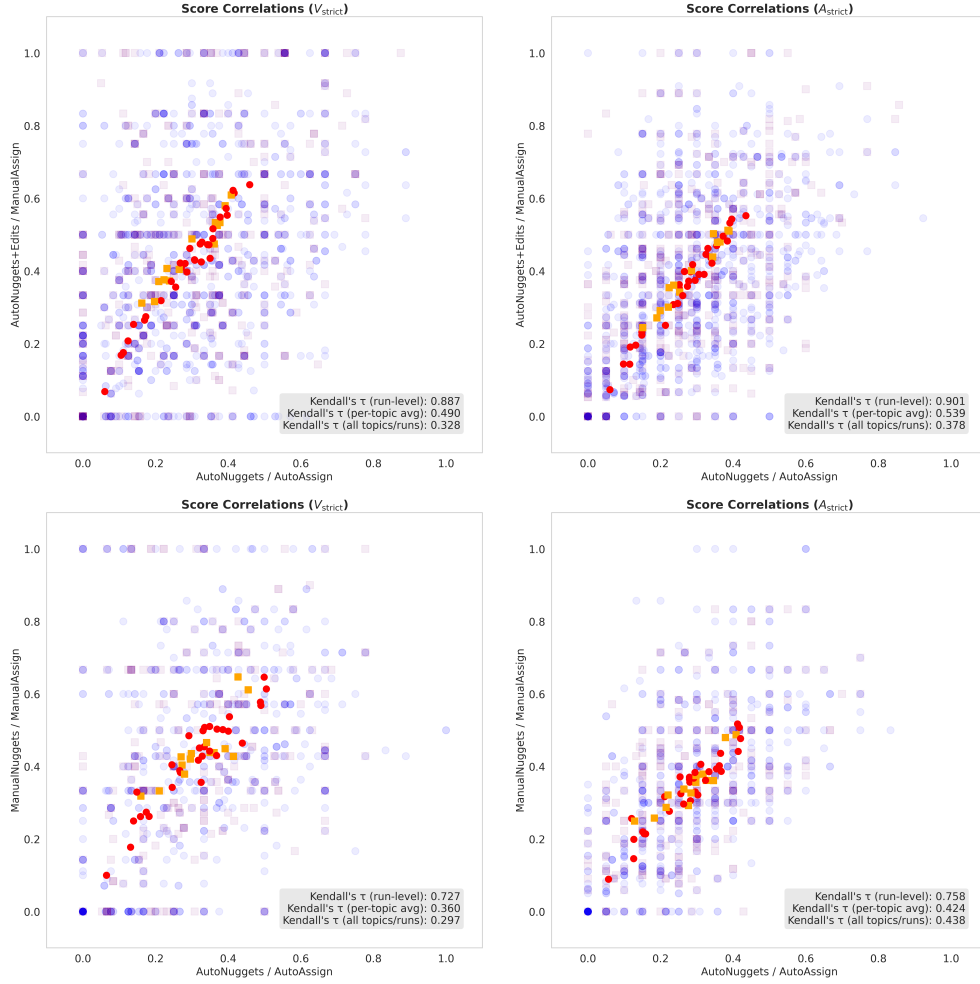


Figure 8.4: Scatter plots between manual vs. automatic V_{strict} and A_{strict} scores for AG and RAG runs. The x axes show scores from AutoNuggets / AutoAssign and the y axes show scores from different manual conditions. Red circles (RAG runs) and orange squares (AG runs) represent run-level scores. Blue circles (RAG runs) and purple squares (AG runs) show all topic/run combinations. The bottom-right box reports Kendall's τ correlations at the run level (red circles/orange squares), over all topic/run combinations (blue circles/purple squares), average of per-topic correlations.

Here, we adopt the meta-evaluation methodology common in the information retrieval literature dating back several decades [266, 227, 23, 24, 198, 79]. We evaluate runs both with our fully automatic nugget evaluation methodology and with the other variants involving

humans. Rank correlations (in terms of Kendall’s τ) between the system orderings induced by the different approaches capture the quality of our automatic metric. Our goal is to obtain high rank correlations against human-based metrics.

Figure 8.4 presents scatter plots of scores derived from the various experimental conditions. In the left column, we show V_{strict} and in the right column, A_{strict} . In all these plots, the x -axes show the AutoNuggets / AutoAssign score. The top row shows the comparison against the AutoNuggets+Edits / ManualAssign condition, where nuggets are automatically generated and post-edited by humans, while assignment is manual. The bottom row shows the comparison against the ManualNuggets / ManualAssign condition, where both nugget creation and assignment are fully manual. Red circles and orange squares represent run-level scores for RAG and AG tasks, respectively. Blue circles (RAG runs) and purple squares (AG runs) represent all topic/run combinations for the same.

We compute Kendall’s τ in three different ways: (1) run-level correlations (i.e., we first compute run scores by averaging topic scores, and then compute correlations); (2) average of per-topic correlations (per-topic avg), where we compute correlations per topic, and then average those; (3) correlation across all topic/run combinations (all topics/runs), where each topic/run combination is considered an independent observation.

For the AutoNuggets+Edits / ManualAssign condition (top row), we observe a strong run-level Kendall’s τ correlation of 0.887 for V_{strict} and 0.901 for A_{strict} . In IR meta-evaluations, these would be sufficient to “validate” a metric. For the ManualNuggets / ManualAssign condition (bottom row), the run-level Kendall’s τ correlations drop to 0.727 for V_{strict} and 0.758 for A_{strict} . The correlations are slightly lower, but still good. This can be partly explained by the much smaller topic count of 20: experimenting with a 20 topic subset of AutoNuggets+Edits results in correlations of 0.826 for V_{strict} and 0.838 for A_{strict} . This drop could perhaps be explained by fully manual nuggets “looking different” due to the assessors being anchored by the automatically generated nuggets (in the post-editing condition). Nevertheless, results suggest good agreement between automatic and manual metrics at the run level.

However, examining the scatter of blue circles and purple squares representing individual topic/run combinations in all the scatter plots, we observe a greater dispersion compared to the run-level scores. The Kendall’s τ correlation across all topic/run combinations is much lower, ranging from 0.297 to 0.438. The average per-topic Kendall’s τ correlations fall to between 0.360 and 0.539. This indicates that while run-level rankings are well-preserved by our automatic evaluation, there is more variability at the topic level.

In summary, to answer **RQ1**:

RQ1 We find that scores from our fully automatic (end-to-end) nugget evaluation framework show strong correlations with manual nugget evaluations at the run level. This suggests that our approach can potentially serve as a good surrogate for manual evaluations in assessing RAG systems.

We observe stronger correlations with AutoNuggets+Edits than ManualNuggets, but Kendall’s τ remains relatively high even with fully manual nuggets. We emphasize that while we achieve good *run-level* correlations, *per-topic* correlations are relatively low, suggesting that our evaluation framework is inadequate for fine-grained debugging of individual answers.

8.3.3 Automatic Nugget Assignment

To investigate **RQ2**, we analyze whether automating nugget assignment in isolation, an expensive annotation step, leads to stronger agreement with manual evaluations compared to fully automating the entire framework. That is, we hold the nuggetization approach constant, but vary the assignment method.

Figure 8.5 compares scores with automatic vs. manual nugget assignment. The top row shows the AutoNuggets+Edits / AutoAssign condition against AutoNuggets+Edits / ManualAssign, isolating the effect of automating assignment when nuggets are post-edited by humans. The bottom row compares ManualNuggets / AutoAssign against ManualNuggets / ManualAssign, isolating the effect of automating assignment when nuggets are manually created.

Comparing the top row of Figure 8.5 to the top row of Figure 8.4, we observe that automating only nugget assignment (AutoNuggets+Edits / AutoAssign vs. AutoNuggets+Edits / ManualAssign) continues to show strong agreement at the run level. Additionally, it results in a higher per-topic average and all topics/runs Kendall’s τ correlations for both V_{strict} and A_{strict} . Specifically, for V_{strict} , the per-topic average Kendall’s τ increases from 0.490 in the fully automatic (AutoNuggets / AutoAssign in Figure 8.4) to 0.662 when only nugget assignment is automated (AutoNuggets+Edits / AutoAssign in Figure 8.5). Similarly, the all topics/runs Kendall’s τ improves from 0.328 to 0.599. For A_{strict} , we observe comparable improvements. These metrics are arguably more sensitive to the nuances of topic-level evaluation, as they capture agreement across individual queries rather than just at the aggregate run level. The substantial increase in correlations suggests that automating nugget assignment while maintaining human-curated nuggets is perhaps a way forward to evaluate a large number of system responses, given that the assignment step occupies a lot of the assessment budget.

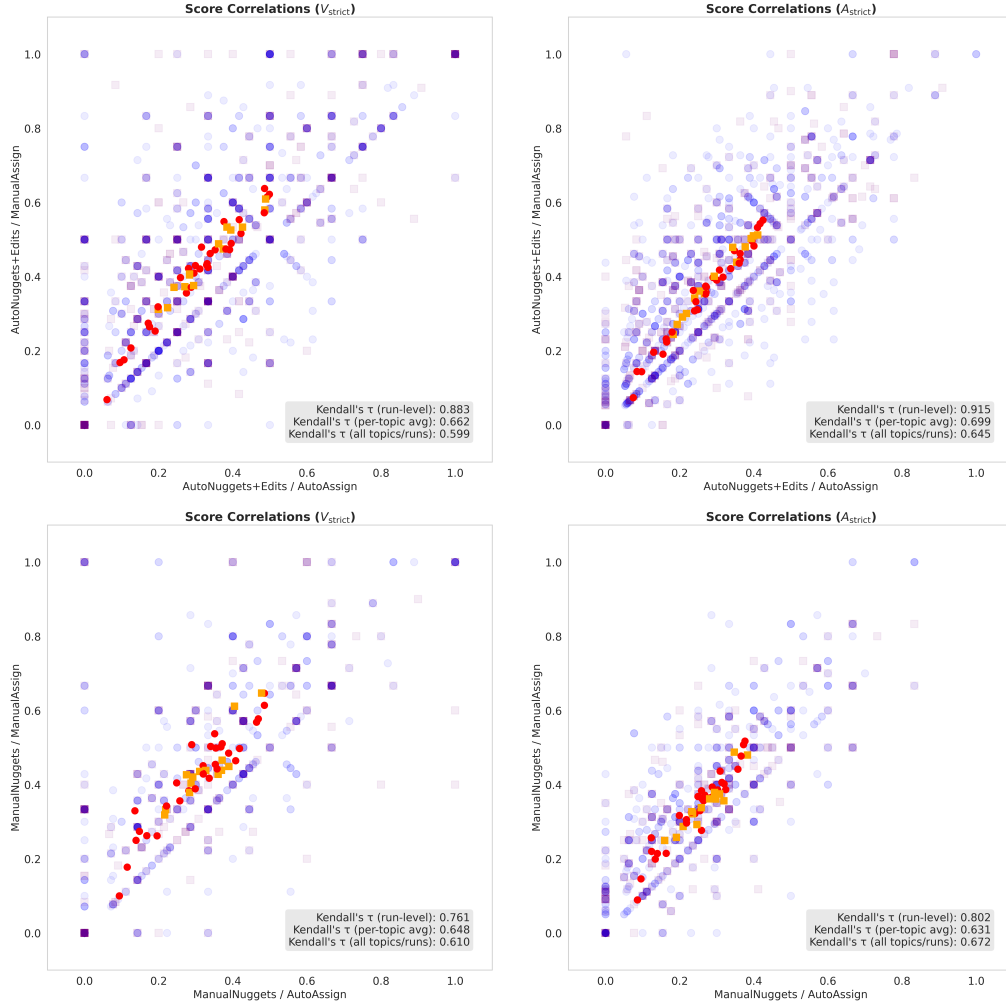


Figure 8.5: Scatter plots showing correlations designed to answer RQ2, isolating the effects of nugget assignment. Overall organization is identical to Figure 8.4.

When we examine the bottom row of Figure 8.5, where nuggets are manually created, we see a similar pattern of improvement. Comparing ManualNuggets / AutoAssign to ManualNuggets / ManualAssign, we observe increases in per-topic average and all topics/runs Kendall's τ correlations for both V_{strict} and A_{strict} . This further reinforces the finding that automating nugget assignment, regardless of whether nuggets are automatically generated and post-edited or manually created from scratch, leads to a better alignment with manual evaluations, particularly at the topic level.

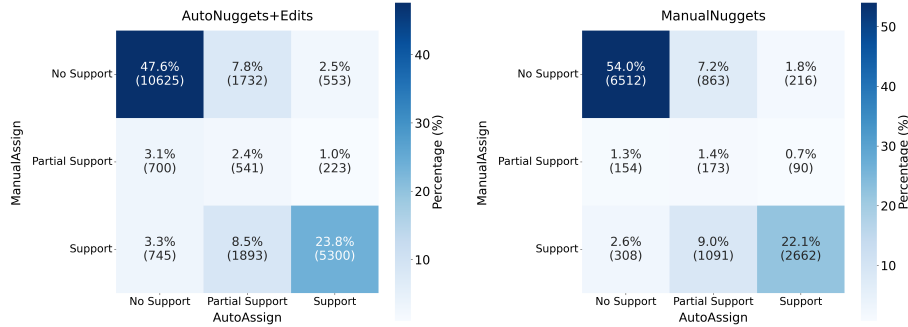


Figure 8.6: Confusion matrices showing agreement between automatic and manual nugget assignment strategies.

To answer **RQ2**:

RQ2 We find that automating only nugget assignment leads to stronger agreement with manual evaluations, compared to a fully automated evaluation where nuggets are constructed automatically.

Our framework is not an “all or nothing” proposition. Instead, there is a vast design space that supports different trade-offs between effort and quality at the run- and topic-level.

8.3.4 Human vs. LLM Assignment

To address **RQ3**, we investigate quantitative differences between human and LLM nugget assignments. Figure 8.6 presents confusion matrices that quantify agreements (and disagreements) in nugget assignment labels between automatic (AutoAssign) and manual (ManualAssign) approaches. The left matrix compares AutoAssign against ManualAssign when using AutoNuggets+Edits nuggets, and the right matrix compares them when using ManualNuggets. Each matrix shows the percentages and counts of nugget assignments falling into different categories.

Examining the diagonal elements of both confusion matrices, we observe substantial agreement between automatic and manual nugget assignment, particularly for the “Support” and “No Support” labels. For AutoNuggets+Edits nuggets (left matrix), 47.6% of the response–nugget pairs resulted in both AutoAssign and ManualAssign labeling a “No Support” and 23.8% a “Support”. For ManualNuggets (right matrix), we see 54.0% of the distribution with “No Support” and 22.1% with “Support”. Partial support appears to be

a source of disagreement in both conditions, as evidenced by only a small portion of the distribution contained in the central element of the matrices.

Off-diagonal elements reveal disagreements. In both matrices, we observe a tendency for AutoAssign to assign “Partial Support” more frequently than ManualAssign. In general, we see a slightly larger mass of the distribution in the lower triangle part of both matrices compared to the upper triangle, indicating that LLM assessors are a bit stricter compared to NIST assessors.

The use of LLMs to provide “draft” nuggets in the prior nuggetization step (AutoNuggets+Edits vs. ManualNuggets) does not appear to drastically alter the alignment across labeling in nugget assignment, as the confusion matrices for both nugget types exhibit similar patterns of agreement and disagreement.

To answer **RQ3**:

RQ3 Our analyses suggest that LLM assessors appear to be more strict than NIST assessors in nugget assignment. Additionally, the use of LLMs to provide “draft” nuggets in the nugget creation step does not appear to noticeably increase alignment with human nugget assignment.

Two points of caution here, though: These findings are specific to the current implementation. For example, it is entirely possible that a different LLM or a different prompt might elicit different behavior. Another gap in our current understanding is the relationship between inter-annotator agreement and LLM–human differences. This analysis only compares *one* set of nugget assignments between an assessor and a particular LLM implementation. We do not know the variations exhibited by different human assessors and different LLMs and how they compare, which would be an interesting question to tackle in the future.

8.4 Concluding Thoughts

We presented the AutoNuggetizer framework, a scalable and validated approach for evaluating the information nugget recall of generative information access systems. By refactoring the classic nugget-based methodology with modern LLMs, we have demonstrated a path toward resolving the Evaluation Challenge outlined in [Chapter 1](#).

AutoNuggetizer provides the research community with a validated and scalable tool to more effectively measure progress in information nugget recall in RAG systems. However,

as noted in our results, the methodology currently exhibits higher variance at the per-topic level compared to run-level correlations. Addressing this limitation is a priority for future work. Potential next steps include moving from single-judge assessments to an ensemble of LLM judges to smooth out stochasticity, and refining prompts with few-shot examples to better calibrate systems with human assessors.

Despite these current limitations, this is especially important today, when there is a significant push towards Knowledge-Grounded or Deep Research Agents. This contribution, paired with the development of open components like RankZephyr and frameworks like Ragnarök for benchmarking TREC 2024 Retrieval-Augmented Generation Track, creates a synergistic foundation for building and rigorously assessing the next generation of information access systems.

Chapter 9

Conclusions and Future Work

In this thesis, we advanced the state of open, robust information access systems by developing interoperable retrieval components and the evaluation toolkits required to measure them rigorously. We addressed three intertwined challenges introduced in [Chapter 1](#): providing effective rerankers, constructing reliable RAG benchmarks, and scaling evaluation practices that keep pace with rapid system progress. Collectively, they demonstrate that we can deliver transparent, reproducible systems that rival proprietary alternatives while driving community progress.

Our contributions are summarized as follows:

- **Pushing the Limits of Supervised Reranking (Chapter 3–4).** We introduced the influential Expando-Mono-Duo T5 pattern, and, through a systematic study of training pointwise rerankers, introduced monoELECTRA.
- **Understanding Generative Retrieval at Scale (Chapter 5).** We offered the first comprehensive analysis of generative retrieval across corpora containing up to 8.8 million passages. The results surfaced scalability bottlenecks and highlighted where sequence-to-sequence retrieval remains practical.
- **Delivering an Open-Source Listwise Reranker (Chapter 6).** We released RankZephyr, a 7B-parameter listwise reranker distilled from larger proprietary systems through a two-stage process. RankZephyr provides the community with a transparent, high-utility component that achieves competitive effectiveness while remaining lightweight enough for research and production use.

- **Standardizing RAG Benchmarking (Chapter 7).** We built Ragnarök, an end-to-end RAG benchmarking ecosystem featuring curated MS MARCO V2.1 resources, diverse query formulations, and turn-key pipelines. This framework was critical to the TREC 2024 Retrieval-Augmented Generation Track and demonstrated how shared infrastructure can accelerate large-scale evaluation.
- **Scaling Nugget-Based Evaluation with LLMs (Chapter 8).** We developed AutoNuggetizer, which operationalizes nugget-based evaluation with modern language models and validated it against human annotations in the TREC 2024 RAG Track. AutoNuggetizer closes the loop between system development and assessment by making fine-grained recall measurement practical and reproducible.

Together, these contributions can be combined to deliver reliable retrieval systems for modern information needs. They collectively respond to the Component, Benchmarking, and Evaluation Challenges by demonstrating that we can construct trustworthy information access pipelines end-to-end.

Building on these findings, we see several promising directions for expanding the reach and impact of this thesis:

- **Deep Research Evaluation for TREC 2025 and 2026.** The TREC 2024 RAG Track established multifaceted evaluation practices, including relevance [221], support assessment [217], and nugget recall via AutoNuggetizer [180]. Moving forward, we must stress-test our automated evaluation pipelines for the “deep research” topics curated for the TREC 2025 and 2026 RAG Tracks, ensuring that complex information needs remain measurable and reproducible.
- **Listwise Rerankers that Reason over Evidence.** Extending RankZephyr to account for redundancy, complementarity, and contradictions within candidate lists is a natural next step. Future training regimes should encourage structured reasoning across documents so that downstream users, whether humans or LLMs, receive candidate sets optimized for end-task utility.
- **Generalizing Nugget-Based Evaluation.** Democratizing RAG evaluation requires a robust, open-source release of AutoNuggetizer, aligning the methodology with the Component Challenge and mitigating reproducibility risks. Beyond web search, we must expand its coverage to high-stakes domains such as biomedical retrieval and conversational settings to explain human preference in side-by-side responses [201].

- **The Distinction Challenge: From Good to Great.** Perhaps the most urgent frontier is resolution. AutoNuggetizer easily separates poor systems from good ones, but doing so for good systems from the very best is still difficult. Future work must develop higher-resolution evaluation frameworks capable of fine-grained system differentiation. This includes improved per-topic effectiveness predictions to identify exactly when and why a system fails, rather than relying solely on aggregate metrics that mask failure modes.

Advancing in these directions will broaden the utility of our open components, strengthen community benchmarking resources, and unlock richer evaluation signals for next-generation information access systems.

References

- [1] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Y. Zou. Gradio: Hassle-free sharing and testing of ML models in the wild. *arXiv:1906.02569*, 2019.
- [2] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In *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 3490–3496, November 2019.
- [3] Marwah Alaofi, Negar Arabzadeh, Charles LA Clarke, and Mark Sanderson. Generative information retrieval evaluation. In *Information Access in the Era of Generative AI*, pages 135–159. Springer, 2024.
- [4] Mohammad Aliannejadi, Zahra Abbasiantaeb, Shubham Chatterjee, Jeffery Dalton, and Leif Azzopardi. TREC ikat 2023: The interactive knowledge assistance track overview. *arXiv:2401.01330*, 2024.
- [5] Samuel Joseph Amouyal, Ohad Rubin, Ori Yoran, Tomer Wolfson, Jonathan Herzig, and Jonathan Berant. QAMPARI: : An open-domain question answering benchmark for questions with many answers from multiple paragraphs. *arXiv:2205.12665*, 2022.
- [6] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha

- Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. Gemini: A family of highly capable multimodal models. *arXiv:2312.11805*, 2023.
- [7] Anthropic. System card: Claude Sonnet 4.5, September 2025. Accessed 2025-10-10.
- [8] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Saniket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. ExT5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*, 2022.
- [9] Nima Asadi and Jimmy Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 997–1000, 2013.
- [10] Akari Asai, Jungo Kasai, Jonathan Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. XOR QA: Cross-lingual open-retrieval question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 547–564. Association for Computational Linguistics, June 2021.
- [11] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Yu Bowen, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xing Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv:2309.16609*, 2023.
- [12] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv:2502.13923*, 2025.

- [13] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated machine reading comprehension dataset. *arXiv:1611.09268*, 2016.
- [14] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994.
- [15] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. Autoregressive search engines: Generating substrings as document identifiers. *arXiv:2204.10628*, 2022.
- [16] Valeria Bolotova, Vladislav Blinov, Falk Scholer, W. Bruce Croft, and Mark Sanderson. A non-factoid question-answering taxonomy. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*, pages 1196–1207, 2022.
- [17] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. InPars: Unsupervised dataset generation for information retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*, pages 2387–2392, 2022.
- [18] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, et al. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, pages 2206–2240, 2022.
- [19] Leonid Boytsov, Preksha Patel, Vivek Sourabh, Riddhi Nisar, Sayani Kundu, Ramya Ramanathan, and Eric Nyberg. InPars-Light: Cost-effective unsupervised training of efficient rankers. *arXiv:2301.02998*, 2023.
- [20] Lola Le Breton, Quentin Fournier, Mariam El Mezouar, John X. Morris, and Sarath Chandar. NeoBERT: A Next-Generation BERT. *arXiv:2502.19587*, 2025.
- [21] Andrei Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of SEQUENCES 1997, June 11-13, 1997, Proceedings*, pages 21–29. IEEE, 1997.
- [22] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell,

- Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
- [23] Chris Buckley and Ellen M. Voorhees. Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000)*, pages 33–40, 2000.
- [24] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 25–32, 2004.
- [25] James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1994)*, pages 302–310, 1994.
- [26] B. Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, Zhaohui Zheng, and Jon Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*, pages 411–420, 2010.
- [27] Harrison Chase. LangChain, 2022.
- [28] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)*, pages 17754–17762, 2024.
- [29] Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J. Shane Culpepper. Efficient cost-aware cascade ranking in multi-stage retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 445–454, 2017.
- [30] Xiaoyang Chen, Yanjiang Liu, Ben He, Le Sun, and Yingfei Sun. Understanding differential search index for text retrieval. *arXiv:2305.02073*, 2023.

- [31] Xuanang Chen, Jian Luo, Ben He, Le Sun, and Yingfei Sun. Towards robust dense retrieval via local ranking alignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, pages 1980–1986, 2022.
- [32] Zijian Chen, Ronak Pradeep, and Jimmy Lin. Accelerating listwise reranking: Reproducing and enhancing first. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 3165–3172. Association for Computing Machinery, 2025. ISBN 9798400715921.
- [33] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality, March 2023.
- [34] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. *arXiv:2403.04132*, 2024.
- [35] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259*, 2014.
- [36] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways, 2022.
- [37] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELEC-

- TRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.
- [38] Cohere. Introducing Command R+: A scalable LLM built for business, 2024.
 - [39] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv:2507.06261*, 2025.
 - [40] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*, pages 758–759, 2009.
 - [41] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the TREC 2019 Deep Learning Track. In *Proceedings of the Twenty-Eighth Text REtrieval Conference Proceedings (TREC 2019)*, 2019.
 - [42] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the TREC 2020 Deep Learning Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference Proceedings (TREC 2020)*, 2020.
 - [43] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. Overview of the TREC 2021 Deep Learning Track. In *Proceedings of the Thirtieth Text REtrieval Conference (TREC 2021)*, May 2021.
 - [44] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. Overview of the TREC 2022 Deep Learning Track. In *Proceedings of the Thirty-First Text REtrieval Conference (TREC 2022)*, 2022.
 - [45] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Hossein A. Rahmani, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. Overview of the TREC 2023 Deep Learning Track. In *Text REtrieval Conference (TREC 2023)*. TREC, February 2024.
 - [46] Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv:1910.10687*, 2019.
 - [47] Zhuyun Dai and Jamie Callan. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 985–988, 2019.

- [48] Zhuyun Dai, Vincent Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv:2209.11755*, 2022.
- [49] Hoa Trang Dang and Jimmy Lin. Different structures for evaluating answers to complex questions: pyramids won’t topple, and neither will human assessors. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pages 768–775, 2007.
- [50] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [51] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. TREC Complex Answer Retrieval overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC 2018)*, 2018.
- [52] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, et al. The Llama 3 herd of models. *arXiv:2407.21783*, 2024.
- [53] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, 2024.
- [54] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv:1805.04833*, 2018.
- [55] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3558–3567, 2019.
- [56] Naghmeh Farzi and Laura Dietz. Pencils down! Automatic rubric-based evaluation of retrieve/generate systems. In *Proceedings of the 2024 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR ’24*, pages 175–184, 2024.

- [57] Yanlin Feng, Simone Papicchio, and Sajjadur Rahman. CypherBench: Towards precise retrieval over full-scale modern knowledge graphs in the LLM era. *arXiv:2412.18702*, 2024.
- [58] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv:2109.10086*, 2021.
- [59] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. From distillation to hard negative sampling: Making sparse neural IR models more effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*, pages 2353–2359, 2022.
- [60] Ophir Frieder. On Democratizing Data: Diminishing Disparity and Increasing Scientific Productivity. *Harvard Data Science Review*, (Special Issue 4), April 2 2024.
- [61] Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. GPTScore: Evaluate as you desire. *arXiv:2302.04166*, 2023.
- [62] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv:2108.05540*, 2021.
- [63] Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, November 2021.
- [64] Luyu Gao, Zhuyun Dai, and Jamie Callan. Understanding BERT rankers under distillation. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval, ICTIR ’20*, page 149–152, 2020.
- [65] Luyu Gao, Zhuyun Dai, Zhen Fan, and Jamie Callan. Complementing lexical retrieval with semantic residual embedding. *arXiv:2004.13969*, 2020.
- [66] Luyu Gao, Zhuyun Dai, and Jamie Callan. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042, June 2021.
- [67] Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *Proceedings of the 43rd European Conference on Information Retrieval (ECIR 2021)*, 2021.

- [68] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, 2023.
- [69] Mingqi Gao, Jie Ruan, Renliang Sun, Xunjian Yin, Shiping Yang, and Xiaojun Wan. Human-like summarization evaluation with ChatGPT. *arXiv:2304.02554*, 2023.
- [70] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, 2023.
- [71] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. ChatGPT outperforms crowdworkers for text-annotation tasks. *Proceedings of the National Academy of Science*, 120(30), 2023.
- [72] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. End-to-end retrieval in continuous space. *arXiv:1811.08008*, 2018.
- [73] Yvette Graham. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, September 2015.
- [74] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16*, page 55–64, 2016.
- [75] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, 2010.
- [76] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, pages 3929–3938, 2020.
- [77] Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jenyuan Wang, Lan Liu, William Yang Wang, Bonan Min, and Vittorio Castelli. RAG-QA arena: Evaluating domain robustness for long-form retrieval augmented question answering. In

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 4354–4374, November 2024.

- [78] Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. Learning-to-rank with BERT in TF-Ranking. *arXiv:2004.08476*, 2020.
- [79] Donna Harman. *Information Retrieval Evaluation*. Morgan & Claypool Publishers, 2011.
- [80] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1993)*, pages 56–68, 1993.
- [81] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [82] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. Interpretable & time-budget-constrained contextualization for re-ranking. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, pages 513–520, 2020.
- [83] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 113–122, 2021.
- [84] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, April 25-29, 2022*, 2022.
- [85] Wenbo Hu, Jia-Chen Gu, Zi-Yi Dou, Mohsen Fayyaz, Pan Lu, Kai-Wei Chang, and Nanyun Peng. MRAG-Bench: Vision-centric evaluation for retrieval-augmented multimodal models. *arXiv:2410.08182*, 2024.
- [86] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, 2017.

- [87] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018)*, pages 279–287, 2018.
- [88] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, 2021.
- [89] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv:2112.09118*, 2021.
- [90] Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. NEFTune: Noisy embeddings improve instruction finetuning. *arXiv:2310.05914*, 2023.
- [91] Xingyu Ji, Aditya Parameswaran, and Madelon Hulsebos. TARGET: Benchmarking table retrieval for generative tasks. In *Proceedings of the NeurIPS 2024 Third Table Representation Learning Workshop*, 2024.
- [92] Kelvin Jiang, Ronak Pradeep, and Jimmy Lin. Exploring listwise evidence reasoning with T5 for fact verification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 402–410, 2021.
- [93] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv:1705.03551*, 2017.
- [94] Ehsan Kamalloo, Aref Jafari, Xinyu Zhang, Nandan Thakur, and Jimmy Lin. HA-GRID: A Human-LLM Collaborative Dataset for Generative Information-Seeking with Attribution. *arXiv:2307.16883*, 2023.
- [95] 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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, November 2020.

- [96] Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. *arXiv:2006.10369*, 2020.
- [97] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, 2020.
- [98] Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48. Association for Computing Machinery, 2020.
- [99] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.
- [100] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, 2020.
- [101] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. *arXiv:2409.12941*, 2024.
- [102] Sayali Kulkarni, Sheide Chammas, Wan Zhu, Fei Sha, and Eugene Ie. AQUaMuSe: Automatically generating datasets for query-based multi-document summarization. *arXiv:2010.12694*, 2020.
- [103] 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 Le, and Slav Petrov. Natural Questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019.
- [104] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

- [105] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv:1909.11942*, 2019.
- [106] Carlos Lassance, Ronak Pradeep, and Jimmy Lin. Naverloo @ TREC Deep Learning and NeuCLIR 2023: As easy as zero, one, two, three — cascading dual encoders, mono, duo, and listo for ad-hoc retrieval. In *Proceedings of the Thirty-Second Text REtrieval Conference (TREC 2023)*, 2023.
- [107] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, July 2019.
- [108] Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. A modern perspective on query likelihood with deep generative retrieval models. *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, 2021.
- [109] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, July 2020.
- [110] 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 33*, pages 9459–9474, 2020.
- [111] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. PARADE: Passage representation aggregation for document reranking. *arXiv:2008.09093*, 2020.
- [112] Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers, 2011.
- [113] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv:2308.03281*, 2023.

- [114] Jimmy Lin. Is searching full text more effective than searching abstracts? *BMC Bioinformatics*, 10:46, 2009.
- [115] Jimmy Lin and Dina Demner-Fushman. Automatically evaluating answers to definition questions. In *Proceedings of the 2005 Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, pages 931–938, 2005.
- [116] Jimmy Lin and Dina Demner-Fushman. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9(5):565–587, 2006.
- [117] Jimmy Lin and Dina Demner-Fushman. Will pyramids built of nuggets topple over? In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 383–390, 2006.
- [118] Jimmy Lin and Pengyi Zhang. Deconstructing nuggets: the stability and reliability of complex question answering evaluation. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 327–334, 2007.
- [119] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, 2021.
- [120] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Morgan & Claypool Publishers, 2021.
- [121] Jimmy Lin, Ronak Pradeep, Tommaso Teofili, and Jasper Xian. Vector search with OpenAI embeddings: Lucene is all you need. *arXiv:2308.14963*, 2023.
- [122] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv:2010.11386*, 2020.
- [123] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, 2021.

- [124] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3214–3252. Association for Computational Linguistics, 2022.
- [125] Jerry Liu. LlamaIndex, 2022.
- [126] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12: 157–173, 02 2024.
- [127] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2017)*, pages 1557–1565, 2017.
- [128] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
- [129] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*, 2019.
- [130] Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. ChatQA: Building GPT-4 level conversational QA models. *arXiv:2401.10225*, 2024.
- [131] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. B-PROP: Bootstrapped pre-training with representative words prediction for ad-hoc retrieval. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, 2021.
- [132] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. PROP: Pre-training with representative words prediction for ad-hoc retrieval. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021.
- [133] Xueguang Ma, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. Document expansions and learned sparse lexical representations for MS MARCO V1 and V2. In

Proceedings of the 45th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022), pages 3187–3197, 2022.

- [134] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning LLaMA for multi-stage text retrieval. *arXiv:2310.08319*, 2023.
- [135] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. Zero-shot listwise document reranking with a large language model. *arXiv:2305.02156*, 2023.
- [136] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1101–1104, 2019.
- [137] Sean MacAvaney, Arman Cohan, and Nazli Goharian. SLEDGE: A simple yet effective zero-shot baseline for coronavirus scientific knowledge search. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4171–4179, 2020.
- [138] Joel Mackenzie, Shane Culpepper, Roi Blanco, Matt Crane, Charles Clarke, and Jimmy Lin. Query driven algorithm selection in early stage retrieval. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM 2018)*, pages 396–404, 2018.
- [139] Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. ExpertQA: Expert-curated questions and attributed answers. *arXiv:2309.07852*, 2023.
- [140] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 437–444, 2006.
- [141] James Mayfield, Eugene Yang, Dawn Lawrie, Sean MacAvaney, Paul McNamee, Douglas W. Oard, Luca Soldaini, Ian Soboroff, Orion Weller, Efsun Kayi, Kate Sanders, Marc Mason, and Noah Hibbler. On the evaluation of machine-generated reports. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2024)*, pages 1904–1915, 2024.
- [142] Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. Arctic-Embed: Scalable, efficient, and accurate text embedding models. *arXiv:2405.05374*, 2024.

- [143] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. Rethinking search: Making domain experts out of dilettantes. *SIGIR Forum*, 55(1), jul 2021.
- [144] Sewon Min, Kalpesh Krishna, Xinxu Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FactScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, 2023.
- [145] Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126, 2019.
- [146] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*, pages 1291–1299, 2017.
- [147] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023)*, pages 2006–2029, 2023.
- [148] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pre-training. *arXiv:2201.10005*, 2022.
- [149] Ani Nenkova and Rebecca Passonneau. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, 2004.
- [150] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, August 2019.
- [151] Xuan-Phi Nguyen, Shrey Pandit, Senthil Purushwalkam, Austin Xu, Hailin Chen, Yifei Ming, Zixuan Ke, Silvio Savarese, Caiming Xiong, and Shafiq Joty. SFR-RAG: Towards contextually faithful LLMs. *arXiv:2409.09916*, 2024.

- [152] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. Sentence-T5: Scaling up sentence encoder from pre-trained text-to-text transfer transformer. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874. Association for Computational Linguistics, May 2022.
- [153] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855. Association for Computational Linguistics, December 2022.
- [154] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv:1901.04085*, 2019.
- [155] Rodrigo Nogueira and Jimmy Lin. From doc2query to docTTTTTquery, 2019.
- [156] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *arXiv:1910.14424*, 2019.
- [157] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv:1904.08375*, 2019.
- [158] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, November 2020.
- [159] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, and Matthew Lease. Neural information retrieval: At the end of the early years. *Information Retrieval*, 21(2–3):111–182, 2018.
- [160] OpenAI. Hello GPT-4o, 2024.
- [161] OpenAI. GPT-5 system card, August 2025. Accessed 2025-10-10.
- [162] 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 E. Miller, Maddie Simens, Amanda Askell,

- Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *arXiv:2203.02155*, 2022.
- [163] Arnold Overwijk, Chenyan Xiong, and Jamie Callan. ClueWeb22: 10 billion web documents with rich information. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*, pages 3360–3362, 2022.
 - [164] Paul Owoicho, Jeff Dalton, Mohammad Aliannejadi, Leif Azzopardi, Johanne R. Trippas, and Svtlana Vakulenko. TREC CAsT 2022: Going beyond user ask and system retrieve with initiative and response generation. In *Proceedings of the Thirty-First Text REtrieval Conference (TREC 2022)*, 2022.
 - [165] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, July 2002.
 - [166] Andrew Parry, Sean MacAvaney, and Debasis Ganguly. Top-down partitioning for efficient list-wise ranking. In *The Third Workshop on Reaching Efficiency in Neural Information Retrieval*, 2024.
 - [167] Virgil Pavlu, Shahzad Rajput, Peter B. Golbus, and Javed A. Aslam. IR system evaluation using nugget-based test collections. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM 2012)*, pages 393–402, 2012.
 - [168] Jan Pedersen. Query understanding at Bing. In *Industry Track Keynote at the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010)*, 2010.
 - [169] Ronak Pradeep, Xueguang Ma, Xinyu Zhang, Hang Cui, Ruizhou Xu, Rodrigo Nogueira, and Jimmy Lin. H₂oloo at TREC 2020: When all you got is a hammer... Deep Learning, Health Misinformation, and Precision Medicine. In *Proceedings of the Twenty-Ninth Text REtrieval Conference (TREC 2020)*, 2020.
 - [170] Ronak Pradeep, Xueguang Ma, Rodrigo Nogueira, and Jimmy Lin. Vera: Prediction techniques for reducing harmful misinformation in consumer health search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2066–2070, 2021. ISBN 9781450380379.

- [171] Ronak Pradeep, Xueguang Ma, Rodrigo Nogueira, and Jimmy Lin. Scientific claim verification with VerT5erini. In *Proceedings of the 12th International Workshop on Health Text Mining and Information Analysis*, pages 94–103, 2021.
- [172] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The Expando-Mono-Duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv:2101.05667*, 2021.
- [173] Ronak Pradeep, Yilin Li, Yuetong Wang, and Jimmy Lin. Neural query synthesis and domain-specific ranking templates for multi-stage clinical trial matching. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*, pages 2325–2330, 2022.
- [174] Ronak Pradeep, Yuqi Liu, Xinyu Zhang, Yilin Li, Andrew Yates, and Jimmy Lin. Squeezing water from a stone: A bag of tricks for further improving cross-encoder effectiveness for reranking. In *Proceedings of the 44th European Conference on Information Retrieval (ECIR 2022), Part I*, pages 655–670, 2022.
- [175] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. RankVicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv:2309.15088*, 2023.
- [176] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. RankZephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv:2312.02724*, 2023.
- [177] Ronak Pradeep, Daniel Lee, Ali Mousavi, Jeffrey Pound, Yisi Sang, Jimmy Lin, Ihab Ilyas, Saloni Potdar, Mostafa Arefiyan, and Yunyao Li. ConvKGYarn: Spinning configurable and scalable conversational knowledge graph QA datasets with large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1176–1206, November 2024.
- [178] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, and Jimmy Lin. Initial nugget evaluation results for the TREC 2024 RAG Track with the AutoNuggetizer framework. *arXiv:2411.09607*, 2024.
- [179] Ronak Pradeep, Nandan Thakur, Sahel Sharifymoghaddam, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, and Jimmy Lin. Ragnarök: A reusable RAG framework and baselines for TREC 2024 Retrieval-Augmented Generation Track. In *Proceedings of the 47th European Conference on Information Retrieval (ECIR 2025), Part I*, pages 132–148, 2025.

- [180] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, and Jimmy Lin. The Great Nugget Recall: Automating fact extraction and RAG evaluation with large language models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 180–190. Association for Computing Machinery, 2025.
- [181] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. *arXiv:2306.17563*, 2023.
- [182] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, 2021.
- [183] Zackary Rackauckas, Arthur Câmara, and Jakub Zavrel. Evaluating RAG-Fusion with RAGElo: an automated Elo-based framework. *arXiv:2406.14783*, 2024.
- [184] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training, 2018.
- [185] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.
- [186] Colin Raffel, Noam 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. *Journal of Machine Learning Research*, 21(140):1–67, 2020.
- [187] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. 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. *arXiv:2305.05065*, 2023.
- [188] Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. FIRST: Faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8642–8652. Association for Computational Linguistics, November 2024.

- [189] Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *arXiv:2203.17189*, 2022.
- [190] Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen Voorhees, Lucy Lu Wang, and William R. Hersh. TREC-COVID: Rationale and structure of an information retrieval shared task for COVID-19. *Journal of the American Medical Informatics Association*, 27(9):1431–1436, 2020.
- [191] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
- [192] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, pages 109–126, 1994.
- [193] Sara Rosenthal, Avirup Sil, Radu Florian, and Salim Roukos. CLAPNQ: cohesive long-form answers from passages in natural questions for RAG systems. *arXiv:2404.02103*, 2024.
- [194] Corby Rosset, Ho-Lam Chung, Guanghui Qin, Ethan C. Chau, Zhuo Feng, Ahmed Awadallah, Jennifer Neville, and Nikhil Rao. Researchy questions: A dataset of multi-perspective, compositional questions for LLM web agents. *arXiv:2402.17896*, 2024.
- [195] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation, 1985.
- [196] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. ARES: An automated evaluation framework for retrieval-augmented generation systems. 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 338–354, 2024.

- [197] Alireza Salemi and Hamed Zamani. Evaluating retrieval quality in retrieval-augmented generation. *arXiv:2404.13781*, 2024.
- [198] Mark Sanderson and Justin Zobel. Information retrieval system evaluation: Effort, sensitivity, and reliability. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005)*, pages 162–169, 2005.
- [199] Ferdinand Schlatt, Maik Fröbe, Harrison Scells, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Benno Stein, Martin Potthast, and Matthias Hagen. Rank-DistILLM: Closing the Effectiveness Gap Between Cross-Encoders and LLMs for Passage Re-ranking. In *Advances in Information Retrieval: 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6–10, 2025, Proceedings, Part III*, page 323–334. Springer-Verlag, 2025.
- [200] Sahel Sharifmoghaddam, Ronak Pradeep, Andre Slavescu, Ryan Nguyen, Andrew Xu, Zijian Chen, Yilin Zhang, Yidi Chen, Jasper Xian, and Jimmy Lin. RankLLM: A python package for reranking with LLMs. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’25*, page 3681–3690. Association for Computing Machinery, 2025.
- [201] Sahel Sharifmoghaddam, Shivani Upadhyay, Nandan Thakur, Ronak Pradeep, and Jimmy Lin. Chatbot Arena meets nuggets: Towards explanations and diagnostics in the evaluation of LLM responses. *arXiv:2504.20006*, 2025.
- [202] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, June 2018.
- [203] Shangeetha Sivasothy, Scott Barnett, Stefanus Kurniawan, Zafaryab Rasool, and Rajesh Vasa. RAGProbe: An automated approach for evaluating RAG applications. *arXiv:2409.19019*, 2024.
- [204] Luca Soldaini and Alessandro Moschitti. The cascade transformer: an application for efficient answer sentence selection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5697–5708, 2020.
- [205] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. ASQA: factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Em-*

- pirical Methods in Natural Language Processing (EMNLP 2022)*, pages 8273–8288, 2022.
- [206] 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. *arXiv:2304.04171*, 2023.
 - [207] Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? Investigating large language models as re-ranking agent. *arXiv:2304.09542*, 2023.
 - [208] Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. Scaling Down, LiTting Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. *arXiv:2312.16098*, 2023.
 - [209] Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *arXiv:2310.07712*, 2023.
 - [210] Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. Ul2: Unifying language learning paradigms. *arXiv:2205.05131*, 2022.
 - [211] 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. *arXiv:2202.06991*, 2022.
 - [212] Qwen Team. Qwen3 technical report. *arXiv:2505.09388*, 2025.
 - [213] 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.
 - [214] Nandan Thakur, Luiz Bonifacio, Crystina Zhang, Odunayo Ogundepo, Ehsan Kammalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Boxing Chen, Mehdi Rezagholizadeh, and Jimmy Lin. Knowing When You Don’t Know: A multilingual relevance assessment dataset for robust retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12508–12526, November 2024.

- [215] Nandan Thakur, Suleman Kazi, Ge Luo, Jimmy Lin, and Amin Ahmad. MIRAGE-bench: Automatic multilingual benchmark arena for retrieval-augmented generation systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 274–298, April 2025.
- [216] Nandan Thakur, Jimmy Lin, Sam Havens, Michael Carbin, Omar Khattab, and Andrew Drozdov. FreshStack: Building realistic benchmarks for evaluating retrieval on technical documents. *arXiv:2504.13128*, 2025.
- [217] Nandan Thakur, Ronak Pradeep, Shivani Upadhyay, Daniel Campos, Nick Craswell, Ian Soboroff, Hoa Trang Dang, and Jimmy Lin. Assessing support for the TREC 2024 RAG Track: A large-scale comparative study of LLM and human evaluations. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’25, page 2759–2763. Association for Computing Machinery, 2025.
- [218] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv:2302.13971*, 2023.
- [219] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*, 2023.
- [220] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourier, Nathan

- Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. Zephyr: Direct distillation of LM alignment. *arXiv:2310.16944*, 2023.
- [221] Shivani Upadhyay, Ronak Pradeep, Nandan Thakur, Daniel Campos, Nick Craswell, Ian Soboroff, Hoa Trang Dang, and Jimmy Lin. A large-scale study of relevance assessments with large language models: An initial look. *arXiv:2411.08275*, 2024.
- [222] Shivani Upadhyay, Ronak Pradeep, Nandan Thakur, Nick Craswell, and Jimmy Lin. UMBRELA: UMBrela is the (open-source reproduction of the) Bing RElevance Assessor. *arXiv:2406.06519*, 2024.
- [223] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [224] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.
- [225] Ellen Voorhees. The philosophy of information retrieval evaluation. In *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, Lecture Notes in Computer Science Volume 2406*, pages 355–370, 2002.
- [226] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. TREC-COVID: Constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54(1): 1–12, 2020.
- [227] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 315–323, 1998.
- [228] Ellen M. Voorhees. Overview of the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, 2003.
- [229] Ellen M. Voorhees. Evaluating answers to definition questions. In *Companion Volume of the Proceedings of HLT-NAACL 2003 — Short Papers*, pages 109–111, 2003.
- [230] Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on*

Research and Development in Information Retrieval (SIGIR 2011), pages 105–114, 2011.

- [231] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. CORD-19: The COVID-19 Open Research Dataset. *arXiv:2004.10706*, 2020.
- [232] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective works best for zero-shot generalization? In *International Conference on Machine Learning*, pages 22964–22984. PMLR, 2022.
- [233] Yang Wang, Alberto Garcia Hernandez, Roman Kyslyi, and Nicholas Kersting. Evaluating quality of answers for retrieval-augmented generation: A strong LLM is all you need. *arXiv:2406.18064*, 2024.
- [234] Yujing Wang, Ying Hou, Hong Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. A neural corpus indexer for document retrieval. *arXiv:2206.02743*, 2022.
- [235] Zora Zhiruo Wang, Akari Asai, Xinyan Velocity Yu, Frank F. Xu, Yiqing Xie, Graham Neubig, and Daniel Fried. CodeRAG-Bench: Can retrieval augment code generation? *arXiv:2406.14497*, 2025.
- [236] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv:2412.13663*, 2024.
- [237] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently. *arXiv:2303.03846*, 2023.
- [238] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison,

- Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace’s Transformers: State-of-the-art natural language processing. *arXiv:1910.03771*, 2019.
- [239] Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. Early exiting BERT for efficient document ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 83–88, 2020.
- [240] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’17, page 55–64, 2017.
- [241] 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 *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*, 2021.
- [242] Fangyuan Xu, Yixiao Song, Mohit Iyyer, and Eunsol Choi. A critical evaluation of evaluations for long-form question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3225–3245, 2023.
- [243] Zhixiang Eddie Xu, Kilian Q. Weinberger, and Olivier Chapelle. The greedy miser: Learning under test-time budgets. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, 2012.
- [244] Eugene Yang, Andrew Yates, Kathryn Ricci, Orion Weller, Vivek Chari, Benjamin Van Durme, and Dawn Lawrie. Rank-K: Test-time reasoning for listwise reranking. *arXiv:2505.14432*, 2025.
- [245] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, 2017.
- [246] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 10(4):Article 16, 2018.

- [247] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. Critically examining the “neural hype”: weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 1129–1132, 2019.
- [248] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, 2019.
- [249] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. CRAG – comprehensive RAG benchmark. *arXiv:2406.04744*, 2024.
- [250] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.
- [251] Andrew Yates, Siddhant Arora, Xinyu Zhang, Wei Yang, Kevin Martin Jose, and Jimmy Lin. Capreolus: A toolkit for end-to-end neural ad hoc retrieval. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 861–864, 2020.
- [252] Andrew Yates, Kevin Martin Jose, Xinyu Zhang, and Jimmy Lin. Flexible IR pipelines with Capreolus. In *Proceedings of the 29th International Conference on Information and Knowledge Management (CIKM 2020)*, 2020.
- [253] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min 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 (SIGIR 2021)*, page 1503–1512, 2021.
- [254] Biao Zhang, Behrooz Ghorbani, Ankur Bapna, Yong Cheng, Xavier Garcia, Jonathan Shen, and Orhan Firat. Examining scaling and transfer of language model architectures for machine translation. In *International Conference on Machine Learning*, pages 26176–26192. PMLR, 2022.

- [255] Edwin Zhang, Nikhil Gupta, Raphael Tang, Xiao Han, Ronak Pradeep, Kuang Lu, Yue Zhang, Rodrigo Nogueira, Kyunghyun Cho, Hui Fang, and Jimmy Lin. Covidex: Neural ranking models and keyword search infrastructure for the COVID-19 Open Research Dataset. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 31–41, 2020.
- [256] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, pages 2021–2032, 2020.
- [257] Xinyu Zhang, Andrew Yates, and Jimmy Lin. Comparing score aggregation approaches for document retrieval with pretrained transformers. In *Proceedings of the 43rd European Conference on Information Retrieval (ECIR 2021), Part II*, pages 150–163. Springer International Publishing, 2021.
- [258] Yidan Zhang, Ting Zhang, Dong Chen, Yujing Wang, Qi Chen, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, Fan Yang, Mao Yang, Qingmin Liao, and Baining Guo. IRGen: Generative modeling for image retrieval. *arXiv:2303.10126*, 2023.
- [259] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.
- [260] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv:2208.09257*, 2022.
- [261] 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. *arXiv:2210.10634*, 2022.
- [262] Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. *arXiv:2310.14122*, 2023.
- [263] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv:2206.10128*, 2022.

- [264] Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A set-wise approach for effective and highly efficient zero-shot ranking with large language models. *arXiv:2310.09497*, 2023.
- [265] Noah Ziemis, Wenhao Yu, Zhihan Zhang, and Meng Jiang. Large language models are built-in autoregressive search engines. *arXiv:2305.09612*, 2023.
- [266] Justin Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 307–314, 1998.