GRITHopper: Decomposition-Free Multi-Hop Dense Retrieval

Anonymous ACL submission

Abstract

Decomposition-based multi-hop retrieval methods rely on many autoregressive steps to break down complex queries, which breaks end-to-end differentiability and is computationally expensive. Decomposition-free methods tackle this, but current decomposition-free approaches struggle with longer multi-hop problems and generalization to out-of-distribution data. To address these challenges, we introduce **GRITHopper-7B**[1], a novel multi-hop dense retrieval model that achieves state-of-the-art performance on both in-distribution and out-of-distribution benchmarks. GRITHopper combines generative and representational instruction tuning by integrating causal language modeling with dense retrieval training. Through controlled studies, we find that incorporating additional context after the retrieval process, referred to as *post-retrieval language modeling*, enhances dense retrieval performance. By including elements such as final answers during training, the model learns to better contextualize and retrieve relevant information. GRITHopper-7B offers a robust, scalable, and generalizable solution for multi-hop dense retrieval, and we release it to the community for future research and applications requiring multi-hop reasoning and retrieval capabilities.

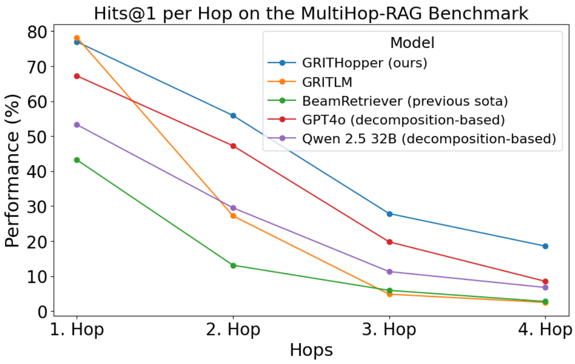

Figure 1: Out-of-distribution Multi-Hop Retrieval Performance on the MultiHop-RAG Benchmark (Tang and Yang, 2024). GRITHopper substantially outperforms previous state-of-the-art multi-hop retrieval models on out-of-distribution Benchmarks on deep hops.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in reasoning (Huang and Chang, 2023), reflection, and decomposition, making them indispensable tools for a wide range of natural language processing tasks. Their generative abilities have been successfully leveraged to solve open-domain multi-hop problems, where complex questions are broken into smaller sub-questions to retrieve supporting evidence and reflect on them (Asai et al., 2024; Shao et al., 2023; Guan et al., 2024) in a step-by-step manner. However, such decomposition-based approaches require multiple autoregressive steps and discrete intermediate outputs, which breaks the end-to-end differentiability of the retrieval pipeline and increases computational overhead.

Decomposition-free approaches, such as Multi-Hop Dense Retrieval (MDR) (Xiong et al., 2021), and cross-encoder-based methods like Beam Retriever (Zhang et al., 2024a), enable end-to-end differentiability by not requiring discrete decompositions, but both suffer from significant limitations. MDR offers an efficient and scalable dense retrieval framework by concatenating the query with passages and encoding them into a single vector representation in one model call per iteration. However, it struggles with more complex datasets like MuSiQue (Trivedi et al., 2022), more hops than 2, and performs poorly on out-of-distribution benchmarks. On the other hand, Beam Retriever achieves state-of-the-art in-distribution performance by leveraging cross-encoder architectures. Unlike bi-encoders, which independently

[1] Anonymous/GritHopper

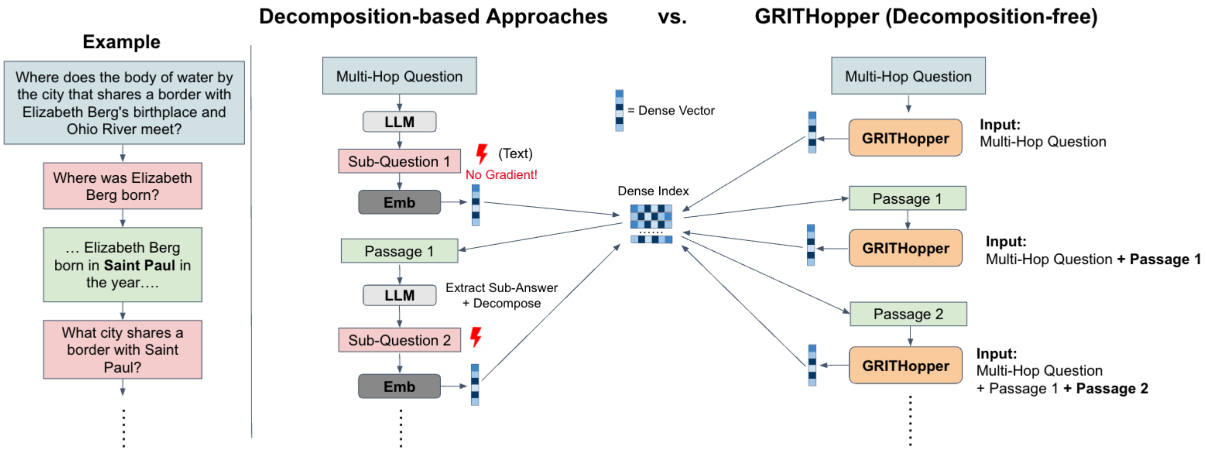

Figure 2: Comparison of decomposition-based approaches like (Guan et al., 2024; Shao et al., 2023) to our encoder-only approach with GRITHopper. While decomposition-based approaches require many auto-regressive steps to decompose questions, extract answers, and a different model for retrieval, our encoder-only approach only requires a single forward pass per hop to compute the next dense vector. Example is from (Trivedi et al., 2022).

encode questions and passages to compute similarity, cross-encoders process both as a single sequence, resulting in linear scaling with respect to the number of passages. This makes them only suited as a retriever for a few hundred passages but not open book retrieval. Despite its strengths, it shares MDR's generalization issues while introducing scalability challenges due to its computational overhead, making it impractical for large-scale open retrieval tasks. These limitations underscore the need for a scalable and generalizable multi-hop retrieval framework that can perform well on both in-distribution and out-of-distribution benchmarks in open-domain retrieval scenarios.

To address these challenges, we introduce GRITHopper-7B, the first decoder-based end-to-end multi-hop dense retrieval model trained on an unprecedented scale of multi-hop datasets spanning both question-answering and fact-checking tasks. GRITHopper-7B achieves state-of-the-art performance across out-of-distribution benchmarks (see Figure 1) while preserving the simplicity and scalability of encoder-only paradigms like MDR (see Figure 2). The foundation of GRITHopper lies in GRITLM (Muennighoff et al., 2025), a Mistral-7B-based model that integrates causal language modeling with dense retrieval training. GRITLM's design sparked a critical debate in the field: *Does joint optimization of generative and retrieval tasks enhance dense embedding quality?* While GRITLM initially demonstrated state-of-the-art results in retrieval while achieving strong performance in generation, subsequent studies (Lee et al., 2025) show that contrastive-only approaches, using the same Mistral-7B backbone, outperform GRITLM on key benchmarks such as BEIR (Thakur et al., 2021) and MTEB (Muennighoff et al., 2023).

This raises fundamental questions about the utility of generative objectives in retrieval and sets the stage for a deeper exploration of their role. Building upon a shared data foundation for both the retrieval and generation objective, we incrementally add information to the generative component without altering the embedding component. This strategy allows us to assess whether incorporating external information (beyond the retrieval chain) into the generative training can improve dense retrieval performance. We refer to this approach as *post-retrieval language modeling*, where we include elements such as final answers and judge the retrieved paragraphs after the retrieval chain. Through this controlled experimental setup, we systematically explore how post-retrieval language modeling influences dense embedding quality, offering new insights into their roles in enhancing multi-hop retrieval performance. Our experiments create a novel ReAct style (Yao et al., 2023) end-to-end multi-hop dense retrieval that can conduct neural search via bi-directional attention and control itself (stop the search, answer, or rerank) via causal language modeling.

The following research questions guide our study:

RQ1: How do decomposition-free approaches compare to decomposition-based approaches?

RQ2: How does GRITHopper generalize on the out-of-distribution benchmarks compared to existing methods?
RQ3: What is the effect of combining generative and embedding training in multi-hop dense retrieval compared to embedding-only training?
RQ4: If generative training improves dense retrieval performance, does post-retrieval language modeling during training further enhance it?

2 Related Work

2.1 Multi-Hop Retrieval and Reasoning

Multi-hop question answering requires models to retrieve and integrate information from multiple documents to answer complex queries (Trivedi et al., 2022; Ho et al., 2020). Decomposition-based methods address this by breaking down complex questions into simpler sub-questions. Wolfson et al. (2020) introduced the Break It Down (Break) method, which decomposes questions into a sequence of simpler queries. Other methods extended decompositions with extensive reasoning (Shao et al., 2023; Khot et al., 2023; Yao et al., 2023). However, these methods require multiple autoregressive steps and generate intermediate outputs, leading to increased computational overhead and disrupting end-to-end differentiability. Decomposition-free approaches have been proposed to overcome these limitations.

2.2 Decomposition-Free Multi-Hop Retrieval

Multi-Hop Dense Retrieval (MDR) (Xiong et al., 2021) introduced an approach where the query is concatenated with previously retrieved passages, and the combined text is encoded into a single vector representation using a bi-encoder architecture. Other works have extended MDR, such as BeamDR by adding beam search and Ma et al. (2024) by extending MDR multi-hop problems longer than 2 hops. While MDR allows for efficient and scalable retrieval but has limitations in handling complex multi-hop queries that require more hops than 2 and generalizing to unseen datasets.

Multi-Hop cross-encoder models (Asai et al., 2020), like the BeamRetriever (Zhang et al., 2024a), achieve state-of-the-art performance on in-distribution datasets by modeling the retrieval process by encoding the question with each paragraph together. Despite their effectiveness, these models face scalability issues due to high computational costs, making them less practical for large-scale open-domain retrieval tasks. Furthermore, we will show that these methods suffer from overfitting and fail to generalize on out-of-distribution benchmarks.

2.3 Causal Language Modeling and Reward Modeling

While Causal language modeling (CLM) is primarily used for generation tasks (Radford et al., 2019), recent research has combined it with dense retrieval, specifically GRITLM Muennighoff et al. (2025), integrating causal language modeling with contrastive learning by simply adding the next token and contrastive loss. While the method trained on two distinct datasets for retrieval and generation, it leaves much room for exploration on how these two losses work together.

In language models, reward modeling can guide the generation process towards more accurate or contextually appropriate responses. Zelikman et al. (2022) and Huang and Chang (2023) explored how self-taught reasoning and reflection can improve reasoning capabilities in language models, which could be beneficial for retrieval tasks that require complex reasoning. To distinguish positive from negative passages, we adopt the approach from (Zhang et al., 2024b) that has shown that language models can simulate reward learning through simple next-token prediction. This comes especially handy for GRITLM's joint generative and embedding objective.

3 Problem Statement & Evaluation

3.1 Problem Definition

In the context of multi-hop retrieval, given a fixed corpus of paragraphs P and a multi-hop-question q, the task is to identify a sequence of paragraphs $[p_1, p_2..., p_n]$ where $p_i \in P$, that collectively answer q (Trivedi et al., 2022; Ho et al., 2020). Decomposition-free methods (Xiong et al., 2021; Zhang et al., 2024a) concatenate the multi-hop question together with previously retrieved paragraphs $[q, p_1, p_2, .., p_n]$ on the word level and feed them as a single string into an Encoder model E to retrieve the next paragraph as:

$$E(q, p_1, p_2, \ldots, p_n) \rightarrow E(p_{n+1}) \qquad (1)$$

where all candidate passages $p_{n+1} \in P$ are pre-computed offline. Apart from question answering, we also adapt fact-checking retrieval as

$[claim, p_1, p_2, .., p_n]$ where paragraphs can either be supporting or refuting paragraphs.

3.2 Datasets

We use a range of datasets to evaluate our approach. We train all models on MuSiQue (Trivedi et al., 2022), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Explainable Fever (Ma et al., 2024), and HoVer (Jiang et al., 2020). These datasets encompass question-answering and fact-checking tasks with varying levels of complexity and hop depths. For out-of-distribution evaluation, we use the MultiHopRAG Benchmark (Tang and Yang, 2024) and MoreHopQA (Schnitzler et al., 2024).

3.3 Evaluation

To demonstrate the performance of all approaches at different hop depths, we calculate Hits@k at each hop. This metric considers a hop successful if the relevant passage is retrieved within the top-k results. Importantly, the evaluation only continues to the next hop if the previous hop was successful. This allows us to analyze the performance across varying hop depths, highlighting the ability of models to retrieve relevant passages in a sequential multi-hop setup. If not explicitly mentioned, we evaluate only the retrieval performance. In our end-to-end evaluation, we also measure the performance of the model to decide when to stop retrieval.

4 Methods

Our central objective is to understand how integrating causal language modeling (CLM) with dense embedding training impacts multi-hop retrieval **(RQ3)**, and whether adding post-retrieval signals (e.g., final answers, judging hard negatives) can further improve performance **(RQ4)**. Unlike prior work, (Muennighoff et al., 2025), which combined generative and embedding training on *different* datasets, we investigate their interplay under a unified, controlled setup. This allows us to isolate the influence of the generative objective on embedding quality. Previous research in language model pretraining has shown that combining masked language modeling (MLM) with embedding training on the *same* dataset often improves downstream representations (Devlin et al., 2019; Wu et al., 2020).

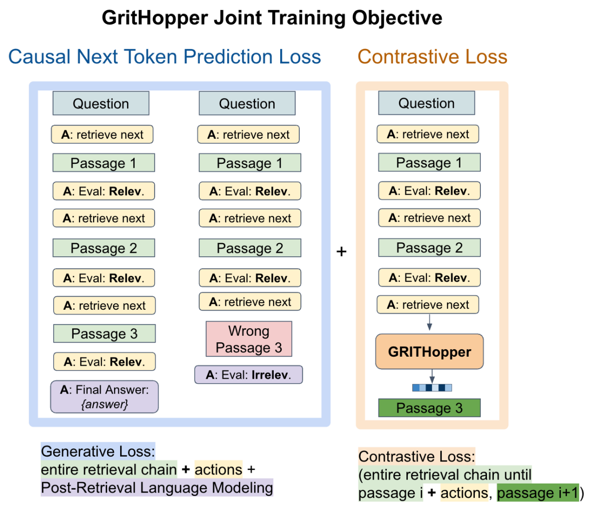

Figure 3: Highlighting the joint training objective (generative and contrastive) of GRITHopper. Both objectives consume the exact same tokens, except for the post-retrieval added information to the generative loss in purple. Note that if the model is used like MDR without a stopping condition, we keep one forward pass per hop to generate the embedding, as all action tokens are only prompt tokens (not output tokens). Only if we want to use the framework end-to-end by controlling when to stop/conduct reranking do we have to do one/two additional causal forward passes.

4.1 A Shared Dataset for a Controlled Setup

To critically evaluate how CLM and embedding objectives affect each other, we start from a *shared dataset*, where both objectives consume identical tokens. Concretely, consider a multi-hop question q and the sequence of previously retrieved paragraphs $[p_1, p_2, \ldots, p_n]$. The embedding model learns to represent $[q, p_1, \ldots, p_n]$ so that it can retrieve the next relevant paragraph p_{n+1}, while the generative model predicts the next tokens on the same sequence in a causal manner. This controlled baseline ensures that any retrieval improvement upon adding the generative loss cannot be attributed to extraneous factors like domain shifts or additional training data. Instead, it must arise from the generative objective itself, addressing **RQ3**: does integrating CLM with embedding training, under controlled conditions, enhance retrieval?

Starting from this shared dataset, we then incrementally enrich the generative model's input with post-retrieval information while keeping the embedding input fixed. This step-by-step strategy ensures that each addition's impact on retrieval is transparent and attributable solely to the newly introduced elements, addressing **RQ4**.

1. Adding Final Answers: We append the final answer ans to the retrieval chain:

$$[q, p_1, p_2, \ldots, p_n, ans].$$

The embedding objective gets the exact same tokens $[q, p_1, \ldots, p_n]$ as the generative objective, while the generative objective now additionally predicts the ans.

2. Adding Hard Negatives: We further augment the generative training by introducing an irrelevant passage p_{ir}, marked as such:

$$[q, p_1, p_2, \ldots, p_{n-1}, p_{ir}, \text{Irrelevant}].$$

The model learns to label the irrelevant paragraph causally via next-token prediction (Zhang et al., 2024b). If retrieval benefits from this, it indicates that contrasting positive and negative evidence in a generative framework helps refine the embedding space. This incremental approach, starting from a pure shared dataset and progressively adding final answers and negatives, provides a precise experimental lens. We directly measure how each augmentation in the generative domain influences the embedding model's retrieval capabilities.

4.2 ReAct-Style Instruction Tuning for End-to-End Multi-Hop Retrieval

To incorporate these different actions to represent the entire multi-hop retrieval as a coherent textual sequence, we adapt the ReAct framework (Yao et al., 2023). Each retrieval hop, document evaluation, and final answer production is expressed as a short instruction or "action" phrase (see Figure 3).

All these actions are represented as textual strings and integrated into the same sequences used by both the embedding and generative objectives. Their exact formatting for all multi-hop datasets (see §3.2) is described in Algorithm 1 in Appendix D. Because these augmented sequences include both the retrieval chain (i.e., $[q, p_1, \ldots]$) and the action strings, we maintain the shared data dataset principle for both embedding and generative training. This ReAct adaptation allows us to combine everything, final answers, negative passages, and retrieval steps, into a single, end-to-end system. Crucially, this framework allows the model to:

- Decide if a retrieved document is relevant or not. (`Eval` in Figure 3)
- Stop the search early if it encounters an irrelevant paragraph. (after (`Eval: Irrelev.`) in Figure 3)
- Continue retrieving until all necessary information is gathered (`retrieve next` in Figure 3)
- Finally, produce the answer. (`Final Answer:` in Figure 3)

In other words, the ReAct-style instruction tuning not only aligns with our controlled experimental design but also yields a system capable of autonomously handling the retrieval pipeline end-to-end. The model can determine how many steps to take and when to stop while providing a realistic and comprehensive testbed for studying the interplay of CLM and embedding objectives in multi-hop retrieval.

5 Experimental Setup

We train GRITHopper in two different setups. First, we explore our ablations by fine-tuning one dataset, MuSiQue (Trivedi et al., 2022). MuSiQue offers decomposition steps with which we can ensure highly qualitative hard negatives and is the most difficult multi-hop question answering dataset in our dataset collection, according to Trivedi et al. (2022). Furthermore, we train our core ablations on a large collection of multi-hop datasets (described in §3.2) on two seeds. We explore in Appendix C how we adapt each dataset in detail and describe the hard negative mining in Appendix B.1.

5.1 Training

GRITHopper-7B is trained on 8 $\times$ A100-80GB GPUs with a contrastive batch size of 2048 using GradCache (Luyu Gao, 2021) and a 256 batch size for the generative loss, like GRITLM in a Fully Sharded Data Parallel (FSDP) setup. We train all models for 5 epochs and select the best checkpoint via dense retrieval performance in the distractor setting.

5.2 Baselines

Our baselines can be split into decomposition-free approaches and decomposition-based approaches. Starting with decomposition-free approaches, we chose GRITLM as our first baseline with the prompting formats we utilize for GRITHopper. GRITLM has also been trained on multi-hop question answering on HotpotQA and several Fever datasets (Thorne et al., 2018) for single-step retrieval. Secondly, we train BeamRetriever (beam size 1), the current state-of-the-art method for multi-hop retrieval and MDR, on MuSiQue as well as our entire dataset collection (see §3.2). However, MDR has only been trained on a fixed number of 2 hops. Therefore, we remove any additional hops after the second hop in our experiments. For MDR, we choose RoBerta-Large (Liu et al., 2019), and for BeamRetriever and Deberta-v3-Base (He et al., 2023), we find that these models perform best among Large and XL variations with the corresponding architectures. For more details on how we explored different base models for these architectures, see appendix A. Besides decomposition-free methods like GRITHopper, BeamRetriever, and MDR, we add an additional baseline using decompositions. For this, we employ a simple one-step-at-a-time decomposition (like (Guan et al., 2024) but with only one try for a fair comparison) method using Qwen 2.5 32B (and GPT4o on two datasets) for decomposing the multi-hop problem into a single sub-question with 4 few-shot samples. In the second step, we use GRITLM to embed the sub-query and retrieve candidates. If a supporting paragraph is retrieved within the top-k range, we continue by asking Qwen/GPT4o to extract the answer and use the previously solved sub-questions to decompose the next sub-query. We provide the prompt templates and GPT4o generation outputs in the appendix A.3.

6 Experiments and Discussion

In this section, we first investigate GRITHopper's ablations in detail, as these represent the core baselines for the first decoder-based multi-hop dense retrieval model. We then compare GRITHopper to existing methods in an open retrieval setting, including decomposition-free BERT-based models (MDR, BeamRetriever), general instruction-tuned retrieval models (GRITLM), and decomposition-based approaches with GPT-4o and Qwen. Subsequently, we focus specifically on decomposition-based methods (RQ1) and analyze GRITHopper's out-of-distribution generalization capabilities (RQ2), highlighting its robustness over previous state-of-the-art approaches. We discuss inference compute in Appendix E and training time in Appendix F.

6.1 Evaluating Generative Objectives and Post-Retrieval Information (RQ3, RQ4)

As GritHopper is the first decoder-based decomposition-free Multi-Hop Dense retriever, we extensively ablate our training objective to motivate our auxiliary training signals, including

1. only contrastive learning (like MDR, just GRITLM fine-tuned on additional multi-hop datasets)
2. contrastive + causal language modeling with no post-retrieval information (same data for causal + contrastive)
3. contrastive + causal language modeling with final answers
4. contrastive + causal language modeling with final answers and causal negative

to address our research questions, RQ3 & RQ4.

We first conduct a series of controlled experiments on the MuSiQue dataset under the distractor setting (see Table 1) and then move to training on all datasets in Table 2 on open retrieval averaged across two seeds. This scenario allows us to

Model	Average Hits@1
Dense Retrieval	
GRITHopper (Answers & Reward)	**82.32**
GRITHopper (Answers)	82.08
GRITHopper (no post lm)	80.78
GRITHopper (Contrastive Only)	78.02
Cross Encoder	
BeamRetriever Large (all datasets)	**85.10**
BeamRetriever (all datasets)	81.78
BeamRetriever (MuSiQue Only)	80.98
GRITHopper ReRank*	59.04
End-to-End Retrieval	
GRITHopper end-to-end*	**75.00**
BeamRetriever end-to-end	38.21

Table 1: MuSiQue distractor-setting dense retrieval performance. All GRITHopper models are trained only on the MuSiQue dataset. * Uses GRITHopper (Answers & Reward). `No post lm` stands for causal modeling only on the retrieval chain

Dataset	Avg. Hits@1 for GRITHopper with:		
	Ans + Rew	Ans	No Post
In Distribution			
ExFever	87.10	**91.81**	89.69
MuSiQue	**76.16**	75.95	75.22
Hover	93.34	**94.29**	94.36
Zero-Shot Benchmarks			
MoreHopQA	**96.14**	95.80	94.68
MultiHopBench	51.74	**54.03**	51.13

Table 2: GRITHopper trained on all datasets in open retrieval performance. Results are averaged over two seeds. **Ans** includes the final answer in the generative samples. **Rew** includes reward modeling to distinguish negatives from positives, while **No Post** does not include post-retrieval language modeling.

isolate and compare the effects of different generative strategies (with and without final answers) and reward modeling before deploying the chosen configurations in the more challenging open retrieval environment.

On MuSiQue, our best-performing GRITHopper variant uses both final answers and reward modeling, achieving a Hits@1 score of 82.32. Even without reward modeling, adding the final answer results in a still-impressive Hits@1 score of 82.08. Compared to a purely contrastive approach (like MDR) without generative signals (78.02), these findings demonstrate that causal language modeling on the same dataset (80.78) improves performance (RQ3). Building on that, the inclusion of final answers (part of RQ4) substantially improves retrieval accuracy (82.08) and is essential for outperforming BeamRetriever in distribution on MuSiQue (81.78). The final answer during training provides a clearer retrieval target, guiding the model to select more relevant passages at each hop. However, when scaling these ablations to all datasets (Table 2), reward modeling, while effective in the distractor setting, led to overfitting in open retrieval. Specifically, the GRITHopper observing negatives causally during training (cross-encoder training) caused a 7.32% drop in Average Hits@1 when transitioning from the distractor setting to open retrieval on MuSiQue, compared to a milder 5.09% drop for its counterpart (only with Answers), averaged over two seeds. This is even more extreme with BeamRetriever, which excels under conditions closely matching its training distribution (distractor setting in Table 1) but struggles to generalize on the same dataset in open retrieval (Table 3). Here, the DeBerta Large version, while achieving the strongest results under distractors (see Table 1), performs worse than the base variant in open retrieval; we explore this further in Appendix A.1. These findings suggest that learning difficult negatives causally can improve discrimination on difficult distractors but hinder broader generalization in dense retrieval. By contrast, GradCache's large in-batch negatives provide a more robust discriminative learning signal while having a slight disadvantage of "hand-crafted" distractor discrimination. Thus, while both generative training and final answers prove beneficial (answering RQ3 and partially RQ4 affirmatively), reward modeling offers only limited gains and at a considerable cost to generalization. Furthermore, we compare the end-to-end performance of the models to stop after the correct amount of hops; BeamRetriever can do so by comparing the scores from the current and the previous hop; if it decreases, it stops (see (Zhang et al., 2024a) Appendix C). However, we find that these scores are biased to decrease after the first hop, often leading to premature stopping. GRITHopper seems to be more robust in this scenario (see Table 1). However, we find a slight misalignment in the causal and dense retrieval performance, which we explore in Appendix B.2.

6.2 Comparison to Existing Methods on Open Retrieval

Table 3 summarizes the performance of various models on both in-distribution and out-of-distribution benchmarks across different hop depths. We compare GRITHopper to GRITLM, BeamRetriever, MDR, and Qwen 32B / GPT4o + GRITLM with decompositions.

Across all evaluated tasks, GRITHopper consistently outperforms all other techniques, including the state-of-the-art model Beam-Retriever, while being significantly more efficient, as we explore in appendix E. For example, on the most difficult dataset, the out-of-distribution MultiHopRAG benchmark, GRITHopper, achieves a significant improvement in Hits@1 at deeper hops. GRITLM, a previous generative-retrieval hybrid model, performs well for the first hop but struggles with deeper hops. BeamRetriever, despite demonstrating strong performance in in-distribution tasks, exhibits a substantial performance drop when tested on the out-of-distribution MultiHopRAG benchmark, highlighting its tendency to overfit on datasets it was trained on. Similarly, while GRITLM is strong in certain scenarios, it cannot match GRITHopper's robustness across multiple datasets and more complex multi-hop problems. In contrast, GRITHopper maintains strong retrieval quality even when encountering unseen data (RQ2). MDR degrades in the scenario the most.

6.3 Decomposition-Based Approaches (RQ1)

We now turn our focus to decomposition-based methods. The Qwen 32B + GRITLM decomposition approach breaks a complex multi-hop query into sub-questions. While this can simplify the reasoning steps, it introduces a notable trade-off in retrieval specificity. As shown in Table 3, the decomposition-based approach demonstrates a

Model	Hits@1					Hits@5					Hits@10				
	1	2	3	4	*Avg*	1	2	3	4	*Avg*	1	2	3	4	*Avg*
MuSiQue															
GRITHopper (ours)	94.25	76.13	55.45	32.10	*76.42*	99.59	96.32	85.92	57.04	*93.18*	99.79	98.59	91.07	69.63	*95.85*
GRITLM	91.15	57.51	22.32	5.43	*60.51*	99.50	91.31	65.49	35.56	*86.18*	99.96	96.61	83.26	51.85	*92.61*
MDR	81.75	45.18	-	-	*63.47*	94.37	71.04	-	-	*82.71*	96.73	78.82	-	-	*87.77*
Beam Retriever	88.75	60.70	30.73	12.84	*62.80*	95.45	85.40	65.84	41.48	*82.85*	97.02	90.44	77.25	51.60	*88.07*
Qwen 2.5 32B + GRITLM decomposition	82.62	45.72	13.91	1.48	*51.06*	95.45	76.25	36.05	13.09	*72.19*	96.69	82.91	46.61	17.78	*77.39*
GPT4o + GRITLM decomposition	81.96	48.53	13.39	1.98	*51.81*	95.82	79.19	33.39	9.63	*72.74*	97.35	85.35	42.23	14.81	*77.58*
Explainable Fever															
GRITHopper (ours)	96.88	92.20	85.38	-	*93.02*	99.79	99.29	98.72	-	*99.40*	99.94	99.53	99.13	-	*99.63*
GRITLM	91.13	54.88	17.28	-	*63.83*	99.47	82.89	41.89	-	*82.99*	99.79	88.47	51.98	-	*87.12*
MDR	92.93	77.16	-	-	*85.13*	99.08	94.11	-	-	*96.62*	99.44	95.97	-	-	*97.72*
Qwen 32B + GRITLM decomposition	63.24	29.88	11.93	-	*40.90*	83.74	55.14	31.87	-	*63.27*	88.96	63.61	40.14	-	*70.34*
HoVer															
GRITHopper (ours)	95.86	91.56	91.69	92.31	*93.88*	99.79	99.61	99.43	100.00	*99.69*	99.95	99.68	99.71	100.00	*99.83*
GRITLM	95.81	88.09	83.95	88.46	*91.81*	99.89	99.53	98.28	96.15	*99.57*	99.89	99.76	98.85	100.00	*99.74*
MDR	84.77	65.69	-	-	*77.10*	96.60	89.51	-	-	*93.75*	97.98	92.51	-	-	*95.78*
Beam Retriever	98.04	88.96	85.96	76.92	*93.42*	99.47	97.56	97.71	100.00	*98.61*	99.73	97.79	97.71	100.00	*98.84*
Qwen 32B + GRITLM decomposition	75.38	61.44	50.43	46.15	*67.69*	82.23	74.84	68.19	69.23	*78.09*	84.24	78.15	72.21	73.08	*80.78*
Zero-Shot Multi-Hop RAG Benchmark															
GRITHopper (ours)	76.98	55.92	27.89	18.59	*55.87*	98.63	89.22	60.97	51.76	*84.80*	99.78	94.90	71.43	64.32	*90.17*
GRITLM	78.23	27.23	4.85	2.51	*40.19*	98.49	75.21	33.76	16.33	*71.98*	99.87	91.04	59.86	36.93	*84.75*
MDR	19.56	2.22	-	-	*10.89*	41.60	9.36	-	-	*25.48*	50.55	15.12	-	-	*32.84*
Beam Retriever	43.24	13.13	5.95	2.76	*22.22*	60.09	28.47	19.56	14.07	*37.52*	68.56	37.03	27.89	19.85	*45.83*
Qwen 32B + GRITLM decomposition	53.30	29.53	11.31	6.78	*33.33*	79.56	60.27	36.05	28.89	*60.68*	86.74	71.00	50.09	42.96	*70.96*
GPT4o + GRITLM decomposition	67.23	47.27	19.81	8.54	*46.83*	91.18	79.51	49.91	29.15	*74.82*	96.41	88.12	64.80	47.74	*84.04*
Zero-Shot MoreHopQA															
GRITHopper (ours)	96.96	93.92	-	-	*95.44*	99.91	99.19	-	-	*99.55*	100.00	99.73	-	-	*99.87*
GRITLM	98.75	95.53	-	-	*97.14*	100.00	98.84	-	-	*99.42*	100.00	99.73	-	-	*99.87*
MDR	88.73	75.58	-	-	*82.16*	98.30	90.79	-	-	*94.54*	99.46	93.47	-	-	*96.47*
Beam Retriever	97.85	93.02	-	-	*95.44*	99.82	98.21	-	-	*99.02*	100.00	98.39	-	-	*99.19*
Qwen 32B + GRITLM decomposition	96.24	55.19	-	-	*75.72*	99.55	65.38	-	-	*82.47*	100.00	68.78	-	-	*84.39*

Table 3: Open Retrieval comparison on different hop depths. We compare our best GRITHopper (with Answers but no reward modeling) to BeamRetriever, GRITLM, MDR, and a decomposition-based approach.

larger gap between Hits@1 and Hits@5 compared to other methods. Specifically, the average gap from Hits@1 to Hits@5 for the decomposition approach is 13.95, which is significantly higher than GRITHopper's 7.44, BeamRetriever's 6.57, and GRITLM's 8.45.

This substantial gap suggests that generated subqueries often underspecify the necessary context, causing initial retrieval inaccuracies. While relevant passages appear among the top-k retrieved documents, the first-ranked results are more likely to be off-target. By contrast, GRITHopper's end-to-end differentiability preserves the full complexity of the query, yielding more specific embeddings that ensure relevant passages appear at the top, reducing the need for multiple autoregressive steps.

7 Conclusion

We introduced **GRITHopper-7B**, a novel multi-hop dense retrieval model that achieves state-of-the-art performance across both in-domain and out-of-distribution datasets. By training on extensive multi-hop datasets in question-answering and fact-checking, GRITHopper-7B outperforms previous decomposition-based methods while maintaining the efficiency of dense encoders. Our study demonstrated that decomposition-free approaches like GRITHopper surpass decomposition-based methods in multi-hop retrieval tasks due to better query specificity and reduced computational overhead. GRITHopper generalizes exceptionally well on out-of-distribution benchmarks, confirming its robustness across diverse datasets. We found that integrating causal language modeling with embedding training substantially enhances dense retrieval performance compared to embedding-only training. Additionally, incorporating post-retrieval language modeling by including final answers further improves the model's ability to retrieve relevant passages, while causal negatives lead to stronger distractor but worse open retrieval performance. We have demonstrated how its generative training enables GRITHopper for end-to-end retrieval, outperforming previous state-of-the-art methods. We release GRITHopper-7B to the community as a resource for future research in natural language processing tasks requiring complex reasoning and retrieval capabilities.

8 Limitations

Despite its state-of-the-art performance, **GRITHopper-7B** has several limitations:

- **Scalability Challenges for Large Corpora:** While GRITHopper efficiently handles open-domain multi-hop retrieval, the reliance on pre-computed dense embeddings limits its scalability for extremely large corpora. The computational cost of creating and maintaining dense representations for frequent updates remains for a 7B model significant.

- **Dependency on High-Quality Hard Negatives:** GRITHopper relies on effective hard negative mining to train contrastive objectives. This dependency may limit its applicability in domains or datasets lacking high-quality distractor annotations or the ability to mine suitable negatives. This is something we especially observe in reward learning, where there are substantial performance drops on datasets where we lack information on answers and sub-questions (like Fact-Checking) to determine which makes a passage irrelevant or relevant.

- **Computational Overhead for Training:** The integration of both embedding and generative objectives requires substantial GPU resources (e.g. 8 $\times$ A100-80GB GPUs). This makes GRITHopper less accessible for research groups with limited computational resources.

- **Sensitivity to Dataset Characteristics:** GRITHopper performs exceptionally well on multi-hop tasks with well-defined retrieval chains (e.g., MuSiQue, HoVer). However, its performance on tasks with noisier or less structured retrieval chains (e.g., conversational QA) remains untested, highlighting potential brittleness to dataset variability.

- **Multi-Hop Dense Retrieval Model** Since, in contrast to GRITLM, we do not train on (retrieval-independent) instruction datasets in parallel, we do not expect that the model will perform well on generation on other tasks. Thus, our model is intended only for decomposition-free multi-hop dense retrieval.

- **Absence of Directly Comparable Baselines for Decoder-based Multi-Hop Dense Retrieval:**

  A central challenge in evaluating our approach arises from the lack of directly comparable baselines. Previous models either (a) employed decoder-based architectures but focused solely on single-hop retrieval for maximum 2-hop problems using only the question and no further context (e.g., GRITLM, (Muennighoff et al., 2025, p. 48)), or (b) addressed multi-hop retrieval problems but exclusively utilized BERT-based architectures (e.g., MDR, BeamRetriever). As GRITHopper represents the first decoder-based model explicitly designed for decomposition-free multi-hop dense retrieval tasks, direct comparisons to prior work are inherently constrained. To address this, we (a) fine-tuned GRITLM on multi-hop datasets to establish a relevant decoder-based baseline and (b) from there conducted comprehensive ablation studies to clearly quantify and isolate the effects of each component within GRITHopper's design. Furthermore, (c) we increased the size of encoder models (e.g. DeBERTaXL) for previous decomposition-free multi-hop retrieval models, which resulted in overfitting and diminished performance (see Appendix A.1).

- **Limited Exploration of End-to-End Retrieval Dynamics:** While GRITHopper enables end-to-end retrieval with generative outputs, its ability to reliably optimize retrieval dynamics is not yet at the optimum. For e.g., the best stopping performance is achieved at 75%, but since we focus on selecting the best dense retriever, the stopping performance is at 71.22%. This choice ensures the best generalization in embedding performance, which typically differs from the optimal generative performance. Future work should explore whether scaling the dataset further can help close this gap between causal language modeling and dense retrieval.

9 Ethics

The development and deployment of **GRITHopper-7B** raise two key ethical considerations. First, the model's reliance on large-scale datasets introduces the risk of propagating biases present in the training data (Prakash and Lee, 2023; Schramowski et al., 2022), potentially leading to skewed retrieval outcomes or amplification of misinformation. Additionally, the open-domain nature of the retrieval task heightens the risk of retrieving sensitive or harmful content, which could pose challenges in privacy and content moderation. Second, GRITHopper's decomposition-free approach reduces interpretability compared to methods that produce intermediate outputs, making it harder to explain and trust its decisions in high-stakes scenarios.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*.

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

Gabriel de Souza P. Moreira, Radek Osmulski, Mengyao Xu, Ronay Ak, Benedikt Schifferer, and Even Oldridge. 2024. Nv-retriever: Improving text embedding models with effective hard-negative mining. *arXiv:2407.15831*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Jian Guan, Wei Wu, zujie wen, Peng Xu, Hongning Wang, and Minlie Huang. 2024. AMOR: A recipe for building adaptable modular knowledge agents through process feedback. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. 2020. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3441–3460, Online. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. NV-embed: Improved techniques for training LLMs as generalist embedding models. In *The Thirteenth International Conference on Learning Representations*, Singapore.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*.

Jiawei Han Jamie Callan Luyu Gao, Yunyi Zhang. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP*.

Huanhuan Ma, Weizhi Xu, Yifan Wei, Liuji Chen, Liang Wang, Qiang Liu, Shu Wu, and Liang Wang. 2024. EX-FEVER: A dataset for multi-hop explainable fact verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9340–9353, Bangkok, Thailand. Association for Computational Linguistics.

Niklas Muennighoff, Hongjin SU, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Nirmalendu Prakash and Roy Ka-Wei Lee. 2023. Layered bias: Interpreting bias in pretrained large language models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 284–295, Singapore. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Julian Schnitzler, Xanh Ho, Jiahao Huang, Florian Boudin, Saku Sugawara, and Akiko Aizawa. 2024. Morehopqa: More than multi-hop reasoning. *arXiv:2406.13397*.

Patrick Schramowski, Cigdem Turan, Nico Andersen, Constantin A. Rothkopf, and Kristian Kersting. 2022. Large pre-trained language models contain human-like biases of what is right and wrong to do. *Nature Machine Intelligence*, 4(3):258–268.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Yixuan Tang and Yi Yang. 2024. Multihop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. 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)*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

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

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. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929, Online. Association for Computational Linguistics.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Liu Yong, and Shen Huang. 2024a. End-to-end beam retrieval for multi-hop question answering. 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 1718–1731, Mexico City, Mexico. Association for Computational Linguistics.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024b. Generative verifiers: Reward modeling as next-token prediction. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

A Baselines

A.1 Beam Retriever

The Beam Retriever (Zhang et al., 2024a) employs a cross-encoder architecture and relies on beam search to determine the number of steps required for retrieving multi-hop evidence. Unlike methods that have a predetermined number of computations, the Beam Retriever dynamically expands or shrinks the retrieval process, which is why the authors train with a Batch Size of 1. Because large-scale parallelization on GPUs requires a uniform number of computations, this variability makes batching and distributed training for the model infeasible. Attempting to scale the Beam Retriever beyond DeBERTa-Base results in both performance degradation in open-retrieval and over-fitting on the distractor setting while facing dramatically increased training times. We tested ModerdBert Large (Warner et al., 2024), DeBerta Large, DeBerta XL and the DeBerta base variant of the original paper. As highlighted in Table 4, we find that larger models, while achieving substantial performance improvements in the distractor setting, drop in performance in open retrieval on the same dataset. Showcasing the overfitting tendency to only train on distractors.

Model	MuSiQue	
	Distractor	Open Retrieval
DeBERTa Base	81.78	**62.80**
DeBERTa Large	**85.10**	61.90
DeBERTa XL	72.36	58.24
ModernBert Large	74.53	60.06

Table 4: BeamRetriever Performance on MuSiQue Distractor vs. MuSiQue Open Retrieval

A.2 MDR

Multi-Hop Dense Retrieval (MDR) (Xiong et al., 2021) is natively designed for exactly two-hop retrieval. Efforts to extend MDR to more than two hops by adapting the loss function, as suggested by Ma et al. (2024), led to instabilities in our experiments, including scenarios where the model's embeddings collapse. Since MDR's loss is computed at the sample level, adapting it for varying hop lengths becomes non-trivial. These complexities, combined with the need to maintain large batch sizes for good generalization, hindered scaling to larger models or additional hops.

We train MDR on 8 $\times$ A100-80GB GPUs and find that batch size must decrease as model size grows. For instance, we can use a batch size of 16×8 for base models, 8×8 for roberta/deberta large ones, and only 2×8 for the largest variant (DeBerta XL). This reduction in batch size likely impacts the model's generalization capabilities. Table 5 in the main paper shows that even scaling MDR to RoBERTa-Large yields only minor improvements, and attempts to go beyond this configuration or handle more than two hops fail due to the aforementioned instabilities. To remain fair to the original authors, we report MDR results that remain as close as possible to their original setup. Bringing MDR up to today's standards would likely involve adopting modern embedding objectives with techniques like gradient caching and instruction-tuned LLM backbones approaches we have integrated in our ablations with GRITHopper, where combining generative and embedding training yields superior performance compared to contrastive-only baselines (like MDR).

A.3 Decompostion based approach

As discussed in Section 5.2, our decomposition-based baseline uses a step-by-step query decomposition approach. Each complex multi-hop question is decomposed into simpler sub-questions, and at each step we retrieve supporting paragraphs and extract the relevant answer.

We employ four prompt templates for decomposition:

1. **First-Hop Sub-Question Generation:** Generates the initial sub-question from the original multi-hop question.
2. **Second-Hop (Next) Sub-Question Generation:** Generates the next sub-question given the original question and the previously answered sub-questions.
3. **Third-Hop (Next) Sub-Question Generation:** Similar to second-hop but for the third hop.
4. **Fourth-Hop (Next) Sub-Question Generation:** Similar to above, for the fourth hop.

Finally, we have an **Answer Extraction Prompt**, used after retrieving paragraphs, to extract the answer snippet.

Note on Evaluation Fairness: We evaluate retrieval performance at each hop by checking if the correct evidence appears within the top-k retrieved paragraphs. This evaluation is independent of the sub-questions order. Thus, regardless of how a model decomposes the problem, the evaluation remains fair and consistent across all methods.

Transparency of GPT4o experiments We provide the code for our GPT4o experiments and

Model	Hits@1					Hits@5					Hits@10				
	1	2	3	4	*Avg*	1	2	3	4	*Avg*	1	2	3	4	*Avg*
Comparison to other models on MuSiQue															
GRITHopper (ours)	93.09	74.93	55.19	32.10	*75.48*	99.75	95.86	86.44	58.02	*93.22*	99.88	97.77	93.05	71.36	*96.03*
GRITLM	91.15	57.51	22.32	5.43	*60.51*	99.50	91.31	65.49	35.56	*86.18*	99.96	96.61	83.26	51.85	*92.61*
Beam Retriever	88.75	60.70	30.73	12.84	*62.80*	95.45	85.40	65.84	41.48	*82.85*	97.02	90.44	77.25	51.60	*88.07*
Qwen 32B + GRITLM decomposition	82.62	45.72	13.91	1.48	*51.06*	95.45	76.25	36.05	13.09	*72.19*	96.69	82.91	46.61	17.78	*77.39*
MDR on MuSiQue															
DeBerta Base	62.43	20.60	-	-	*41.52*	79.98	40.67	-	-	*60.32*	85.52	49.28	-	-	*67.40*
Deberta Large	74.35	32.06	-	-	*53.21*	85.97	52.25	-	-	*69.11*	89.78	59.95	-	-	*74.87*
XL DeBerta	87.05	48.37	-	-	*67.71*	96.07	75.42	-	-	*85.75*	97.60	82.75	-	-	*90.17*
Roberta Large	86.06	50.19	-	-	***68.12***	95.32	76.71	-	-	*86.02*	96.40	82.42	-	-	*89.41*
MDR on all Datasets															
Roberta Large	81.75	45.18	-	-	*63.47*	94.37	71.04	-	-	*82.71*	96.73	78.82	-	-	*87.77*

Table 5: MDR ablations on different backbone architecturs

Prompt B.1: Decomposition of next Sub-Question

You are given a multi-hop question and the answers to previous sub-questions. Given this information, break down the multi-hop question into the next smaller sub-question that can be answered by retrieving information via a search engine.
`(Few-shot Examples: Multi-hop question + previous answers)`
Input:

```
Multi-hop Question: {multi_hop_question}
Previous Sub-Questions and Answers: {history}
```

Output:

```
Next Sub-Question: {generated_sub_question}
```

Prompt B.2: Answer Extraction

You are given a question and a paragraph that contains the answer. Extract the relevant part of the paragraph that answers the sub-question. Ensure that the answer is as concise and accurate as possible.
`(Few-shot Examples: Question + Retrieved Paragraph)`
Input:

```
Question: {sub_question}
Retrieved Paragraph: {retrieved_paragraph}
```

Output:

```
Answer: {extracted_answer}
```

Figure 4: Decomposition and Answer Extraction Prompt Templates. Few-shot examples include similar multi-hop problems with previously answered sub-questions and answers, demonstrating a consistent step-by-step structure. We provide a custom decomposition instruction for the first hop and provide custom 4 few-shot samples for each additional hop.

GPT4o generations as part of our anonymous GitHub Repository.

Evaluation. For evaluation, we follow a standard hits@k metric at each hop. We compare all models on their ability to retrieve the correct evidence at hop 1, then at hop 2, and so forth. To ensure a fair comparison, we do not rely on the self-correctness of decomposition-based methods as they inherently involve autoregressive generation, which allows multiple retries. In contrast, our decomposition-free approach computes a single dense embedding per step, making it significantly more efficient. While self-correction could improve performance, it introduces additional inefficiencies, contradicting the goal of comparing methods under the most efficient setting. Importantly, decomposition-based methods already require separate models for generation and embedding, further increasing computational cost.

B Training of GRITHopper

In this section, we describe how GRITHopper was trained and how we derived the used training setup.

B.1 Hard Negative Mining and Curriculum Learning

de Souza P. Moreira et al. (2024) have shown that mining difficult hard negatives is essential for achieving good dense retrieval performance. We employ the strongest GRITHopper model from our preliminary experiments, which has only been trained with distractors as hard negatives, to search via dense search the most difficult examples across the entire dataset for our final training run. For datasets like MuSiQue that provide entire decompositions (sub-questions with sub-answers for each hop), we filter distractors that contain the sub-answer. For other datasets where we are not able to filter this way, we filter negatives that have a cosine similarity higher than 0.95 to the positive paragraph. We select 10 hard negatives for the contrastive loss for each positive sample and add the most difficult one to our generative loss. We find that this is essential for making the causal reward learning work. Initially, we employed a curriculum learning approach: after each epoch, we used the current model's predictions to mine new negatives for the subsequent epoch. However, longer training (beyond two or more epochs) led to overfitting and hindered out-of-distribution performance. We also tried taking the model checkpoints from one epoch to mine negatives, and then re-initializing a fresh model with those mined negatives. This approach did prove beneficial and improved 4% on MusiQue in preliminary experiments.

B.2 Causal vs Dense Retrieval Performance

We find that when training on all datasets, the peek performance on causal performance is only reached after 3 times longer training than for optimal embedding performance, leading to overfitting. To not sacrifice embedding generalization, GritHopper on all datasets has, therefore, a slightly weaker end-to-end performance at 71.22 than its MuSiQue Only version at 75. We observe this also in the re-ranking performance which is significantly lower at 59.04, and although extended training improves re-ranking to up to 76.78, it still does not surpass the embedding performance while leading to overfitting on the dense retrieval task.

C Detailed Dataset adaptations

We first discuss the evaluation dataset specifics for evaluation and then our Training Dataset construction.

C.1 Detailed dataset statistics for Evaluation

We show the evaluation dataset statistics in Table 7. We use all paragraphs used for solving the multi-hop problems as negatives for our open retrieval setting. We do not add even more examples as this would make a comparison to the current state-of-the-art model BeamRetriever impossible. This gives us a candidate pool between 2000 samples for MoreHopQA and up to 20000 samples in Explainable Fever in our experiments. This already can lead to BeamRetriever requiring up to 400 hours to solve one dataset as we discuss in Appendix E.

C.1.1 Training Dataset

We use the entire dataset of MuSiQue, HotpotQA as well as Hover. In Hover and ExFever, however, we find that not all hops are multi-hop if we remove duplicated evidence in the same sample, resulting in some 1-hop problems. The 2WikiMultiHopQA consist of only 2 hop and 4 hop problems, as we have a large amount from 2 hop problems already from HotpotQA, we only take 4 hop problems from there to not further unbalance the length of hops. While the post-retrieval information for MultiHop Question answering is clear, for fact-checking, we adapt whether the claim is supportive or unsupportive as the final answer. From Hover, we only use supporting paragraphs as it has no refuted label, making incomplete/irrelevant as positives unsuitable for contrastive learning.

Dataset	Total Samples	Samples Per Hop			
		Hop 1	Hop 2	Hop 3	Hop 4
MuSiQue	19,938	0	14,376	4,387	1,175
HoVer	10,280	3,762	5,579	883	56
HotpotQA	90,447	0	90,447	0	0
ExFever	28,774	1,272	17,444	10,058	0
2WikiMultiHopQA	34,942	0	0	0	34,631
Total	**184,070**	**5,034**	**127,846**	**15,328**	**35,862**

Table 6: Training dataset statistics, including the total number of samples and the distribution of samples across different hop depths for each dataset. The final row shows the aggregate totals, providing an overview of the dataset scale when training across all datasets.

C.1.2 Open Evaluation statistics

Dataset	total	Samples Per Hop			
		1	2	3	4
MoreHopQA	1,118	0	1,118	0	0
ExFever	8,038	166	4,671	3,201	0
MuSiQue	2,417	0	1,252	760	405
MultiHopBench	2,556	0	1,079	778	398
Hover	1,885	617	919	323	26

Table 7: Dataset statistics for the open retrieval evaluation setup. The table includes the number of multi-hop problems and the distribution of samples across different hop depths for each dataset.

In this section, we compare the computational complexity of a cross-encoder-based multi-hop retriever (e.g., Beam Retriever) and a dense bi-encoder-based multi-hop retriever (e.g., GRITHopper and MDR) under the scenario where both must consider the entire corpus of P passages at each retrieval hop. This corresponds directly to the setting in our experiments, where the Beam Retriever processes all P passages at every hop without a first-stage filter, resulting in prohibitively long runtimes.

D Algorithm Dataset formatting

Algorithm 1 Dataset Construction for Multi-Hop Retrieval. For each multi-hop problem, the algorithm iterates through each hop (decomposition step). At each hop, it creates a contrastive sample consisting of the current retrieval prompt, a positive paragraph (supporting evidence), and a mined hard negative paragraph. Additionally, it generates a causal (generative) negative sample indicating the irrelevance of the mined negative paragraph. After processing all hops, the final generative positive sample includes the complete retrieval chain followed by the final answer. One random negative generative sample from the set of causal negatives is also selected to balance the dataset.

Input: Multi-hop dataset $\mathcal{D} = \{(q, \mathcal{P}, a)\}$, where q is the question, $\mathcal{P}$ is the set of paragraphs, $\mathcal{P}_s \subseteq \mathcal{P}$ are supporting paragraphs, and a is the final answer.

Output: Generative samples $\mathcal{S}_g$, Contrastive samples $\mathcal{S}_r$.

Initialize $\mathcal{S}_g \leftarrow \emptyset$, $\mathcal{S}_r \leftarrow \emptyset$
Set instructions $Inst_Q$, $Inst_D$, and actions
for $(q, \mathcal{P}, a) \in \mathcal{D}$ **do**
    $P \leftarrow Inst_Q + q$
    ▷ Initialize retrieval prompt
    $\mathcal{S}_{neg} \leftarrow \emptyset$
    for $i = 1$ to $|\mathcal{Q}_d|$ **do**
    ▷ Iterate through decomposition steps $\mathcal{Q}_d$
        $P \leftarrow \mathcal{Q}_d[i]$
        $D_{neg} \leftarrow mine\_negative(P, \mathcal{P})$
        $D_{pos} \leftarrow \mathcal{P}_s[i]$
        $\mathcal{S}_r \leftarrow \mathcal{S}_r \cup (P, D_{pos}, D_{neg})$
        $P_{neg} \leftarrow P +$ `Document:` D_{neg}
        $P_{neg} \leftarrow P_{neg} +$ `Eval(neg)`
        $\mathcal{S}_{neg} \leftarrow \mathcal{S}_{neg} \cup P_{neg}$
        ▷ next continue with positive chain
        $P \leftarrow P +$ `Document:` D_{pos}
        $P \leftarrow P +$ `Eval(pos)`
        if $i \neq |\mathcal{Q}_d|$ **then** ▷ Final step
            $P \leftarrow P +$ `Retr`
        end if
    end for
    $P_{final} \leftarrow P +$ `Answer:` a
    $\mathcal{S}_g \leftarrow \mathcal{S}_g \cup P_{final}$
    $\mathcal{S}_g \leftarrow \mathcal{S}_g \cup$ `random_select`$(\mathcal{S}_{neg})$
    ▷ to balance positive and negatives
end for
return $\mathcal{S}_g, \mathcal{S}_r$

E Complexity Analysis

Notation:

- Q: Number of queries.
- H: Average number of hops per query.
- P: Total number of passages in the corpus.
- L_q: Length (in tokens) of the query plus previously retrieved context.
- L_p: Length (in tokens) of a passage.
- $\mathcal{C}_{model}(L)$: Compute cost of a single forward pass on an input of length L.
- $\mathcal{C}_{search}(P)$: Compute cost of searching P pre-encoded embeddings (sub-linear in P using ANN indexes).

E.1 Cross-Encoder (Beam Retriever)

The cross-encoder must re-encode each passage together with the query at every hop. Without any pre-retrieval pruning, it compares against all P passages each time:

$$O(Q \cdot H \cdot P \cdot \mathcal{C}_{model}(L_q + L_p)).$$

Since every passage is processed through the cross-encoder at every hop, runtime grows linearly with P and H. For large P, this becomes extremely time-consuming (e.g., hundreds of hours).

E.2 Dense Bi-Encoder (GRITHopper)

Dense retrieval encodes all P passages *once* offline:

$$O(P \cdot \mathcal{C}_{model}(L_p)).$$

At inference time, each hop only requires encoding the query and performing a vector search over P:

$$O(Q \cdot H \cdot [\mathcal{C}_{model}(L_q) + \mathcal{C}_{search}(P)]).$$

Because the passages are already encoded, the cost per hop is dominated by a single query encoding and efficient similarity search. This typically takes orders of magnitude less time than re-encoding P passages at every hop.

E.3 Discussion

Under identical conditions, considering all P passages at each hop, the Beam Retriever's complexity grows as $O(Q \cdot H \cdot P)$ with a high per-pass token cost, resulting in very long runtimes (e.g., over 400 hours in our ExFever open-retrieval experiments). In contrast, GRITHopper amortizes passage encoding and relies on fast search structures, completing the same task in 8 minutes and 20 seconds. This substantial practical difference in runtime reflects the asymptotic advantage of dense retrieval for large-scale, multi-hop scenarios.

F Training Time Comparison

See Table 8.

Model	Trained Epochs	Best Perf. Epoch	# GPUs	Training Time (h)	Total GPU Hours
GritHopper (7B)	5	1-2	8	181	1448
BeamRetriever DeBERTa XL	10 (default: 20)	-*	1	452	452
BeamRetriever DeBERTa Large	20	14	1	289	289
BeamRetriever DeBERTa Base	20	7	1	112	112

Table 8: Training time comparison of different retrieval models trained on all datasets. The table shows the base model, the number of trained epochs, the best performance epoch, the number of GPUs used, the total training time in hours, and the total GPU hours (number of GPUs $\times$ training time). -* performance plateau was not reached.