Generative Retrieval for Long Sequences

Anonymous ACL submission

Abstract

Text retrieval is often formulated as mapping the query and the target items (e.g., passages) to the same vector space and finding the item whose embedding is closest to that of the query. In this paper, we explore a generative approach as an alternative, where we use an encoder-decoder model to memorize the target corpus in a generative manner and then finetune it on query-to-passage generation. As GENRE (Cao et al., 2021) has shown that entities can be retrieved in a generative way, our work can be considered as its generalization to longer text. We show that it consistently achieves comparable performance to traditional bi-encoder retrieval on diverse datasets and is especially strong at retrieving highly structured items, such as reasoning chains and graph relations, while demonstrating superior GPU memory and time complexity. We also conjecture that generative retrieval is complementary to traditional retrieval, as we find that an ensemble of both outperforms homogeneous ensembles.

1 Introduction

Document or passage retrieval is often formulated as the task of encoding both the query and the retrieval sequences to a common vector space and then finding the sequences whose embedding is the closest to that of the query. This *bi-encoder* approach for retrieval is often considered as *de facto* standard, where heavy computations such as obtaining the dense embeddings of the retrieval sequences in the corpus can be done offline, and one can search over a large number of items with low latency through the nearest neighbor search (NNS) or maximum inner product search (MIPS) (Chen et al., 2017; Karpukhin et al.; Lewis et al., 2020; Chen et al., 2020; Wu et al., 2020; Xiong et al., 2021; Roller et al., 2021).

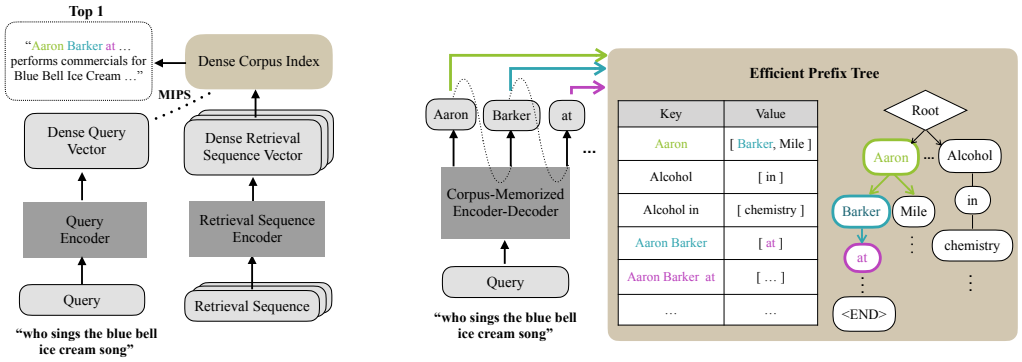
Recently, Cao et al. (2021) have proposed GENRE, which formulates entity retrieval task as generating the entity text in a generative manner with the query as the input to an encoder-decoder model. Such generative formulation has several advantages over traditional bi-encoder approaches: it can cross-encode the input and the output (retrieval sequence) efficiently without information loss while using a smaller storage footprint, and it only needs the prefix tree constructed by corpus set, which is not dependent on the model parameters and is much faster to build. However, they apply generative retrieval to only entities whose length is around three words on average.

In this work, we explore the generalization of generative retrieval (Cao et al., 2021) to (1) longer and diverse types of retrieval sequences, including highly structured forms, and (2) tasks that require an arbitrary number of retrieval iterations (retrieval steps). Our proposed generative retriever modified to adapt to long sequences is called Generative Retrieval for Long Sequences (GRLS). In order to make generative retrieval suitable for long-sequence multi-step retrieval, we propose an *efficient* constrained beam search that reduces the search time complexity of potential next tokens list from $O(h)$, where h is the height of the prefix tree, to $O(1)$. We also propose *retrieval corpus memorization* which learns the target corpus using the standard language modeling objective function before finetuning on the target retrieval task for the model to be fully aware of what it needs to retrieve in a generative manner.

We experiment on six retrieval datasets under three settings: single-step, fixed multi-step, and dynamic multi-step. Single-hop datasets such as Natural

single-step setting. In *fixed multi-step*, the number of items to retrieve is given as an oracle, whereas in *dynamic multi-step*, the model also needs to determine when to stop retrieving the next item. The main findings of our paper are:

- Generative retrieval can be effective for not only short sequences with the single-step set-



(a) Bi-Encoder Retrieval (BE)

(b) Generative Retrieval for Long Sequences (GRLS)

Figure 1: The overall process of bi-encoder (BE) and Generative Retrieval for Long Sequences (GRLS). Given a query, BE retrieves the retrieval sequence most relevant to the query by performing MIPS over the corpus index, and GRLS generates the most relevant sequence inf the corpus by referring to the prefix tree. We make the prefix tree more efficient by using the previously retrieved tokens as the key and the potential next tokens as the value of the prefix tree, reducing the search time complexity from $O(h)$ to $O(1)$ (dictionary table of (b)). GRLS is also modified to be applied to various retrieval tasks, including the retrieval of lengthy sequences and multi-step retrieval. To let the model learn in advance what information would be at the end of the sequence to generate, GRLS uses retrieval corpus memorization before training on target retrieval tasks.

ting but also for long sequences with the multi-step setting.

- Generative retrieval especially shows strong performance on retrieval sequences seen during training and for retrieving multiple highly structured retrieval sequences.
- Generative retrieval and bi-encoder retrieval are often complementary to each other, as an ensemble of them outperforms homogeneous ensembles.

Given that GRLS can have better GPU memory and time efficiency than bi-encoder retrieval, these findings suggest that generative retrieval has the potential to be a practical alternative for retrieving diverse types of sequences.

2 Related Work

Traditional text retrieval has focused on sparse term-based retrieval that uses bag-of-words representations of the texts to measure the relevance between the texts in the corpus and the query, such as TF-IDF and BM25 (Robertson, 2008). Sparse feature vectors are often handled via inverted index by looking up the items with common hot dimensions.

Neural information retrievals utilize neural models to perform information retrieval, where a bi-encoder form is commonly used for large-scale retrieval. It maps queries and sequences in the corpus to shared vector space using encoders (Karpukhin et al.; Xiong et al., 2021). Bi-encoder retrievers can store the dense embedding vectors of the corpus offline and retrieve relevant texts through maximum inner product search (MIPS) or nearest neighbor search. However, they have several limitations: they suffer from information loss when condensing the text into a fixed-size vector, need to renew the stored embeddings when parameter changes, and the latency of performing exact MIPS, which is a linear-time search and the storage footprint for storing the embedding vector become nontrivial as the corpus increases.

Various studies have been conducted to improve the search time efficiency of dense retrieval. Sublinear-time nearest neighbor search has been studied to reduce the latency of performing the exact MIPS. In metric space (L1, L2), Locality Sensitive Hashing (LSH) (Gionis et al., 1999) is a classic search algorithm that hashes the nearby vectors into the same cell. Asymmetric LSH (aLSH) (Shrivastava and Li, 2014) apply LSH to non-metric space such as inner product space by transforming MIPS into minimizing L2 distance with a trick of increasing the vector dimension by one. Yet, applying these approximate search methods still requires time for adopting the ad-hoc process and additional memory (Malkov and Yashunin, 2018).

Cao et al. (2021) first propose a generative retrieval model, GENRE (Generative Entity REtrieval), which is free from the aforementioned limitations of bi-encoder retrieval. It achieves com-

parable or higher performance on entity retrieval tasks than bi-encoder models. To ensure that all generated retrieval sequences are from the corpus, they perform *constrained decoding*, which masks out the tokens that do not form any of the texts in the corpus at each time step during inference.

With the autoregressive formulation, a generative retriever cross-encodes the input and output efficiently, capturing the relation between the two without information loss. When given the same corpus, a bi-encoder retriever has to store all dense embeddings of the corpus while generative retrieval only requires storage for prefix tree to perform constrained decoding, which takes up a smaller storage footprint (Cao et al., 2021). Moreover, while bi-encoder retrieval needs to renew all embedding vectors when the model parameter changes, which often takes long, generative retrieval uses a prefix tree that is not only faster to build but also built only once for a corpus set (thus does not depend on the model parameters).

The main difference of our work from that of Cao et al. (2021) is that we generalize the problem setting of generative retrieval to longer and diverse types of retrieval sequences (including highly structured forms) and tasks that require an arbitrary number of retrieval steps. It is both nontrivial and under-explored to study whether generative retrieval can be well-adopted to general retrieval tasks; this serves as the motivation of our paper.

3 Generative Retrieval for Long Sequences

GRLS (Figure 1b) formulates retrieval tasks as text generation. In training step, the objective is to maximize score(q,p) which is the probability of the parameters θ to generate the length L retrieval sequence p that consists of target tokens y_i, $i = 1, \cdots, L$ in a generative manner: score$(q,p) = P_\theta(p|q) = \prod_{i=1}^{L} P_\theta(y_i|y_{<i}, q)$. Here, the query q is the input to the encoder of the encoder-decoder model, and the ground-truth target tokens of previous iterations $y_{<i}$ are given as the input to the decoder following the teacher forcing approach (Sutskever et al., 2011).

In the inference step, as shown in Figure 1, to generate the second token, *Barker*, it finds the potential next tokens ([Barker, Miller]) by searching through the prefix tree with the previously generated tokens. We mask out tokens that are not in the potential next tokens and find the token with the maximum score from unmasked tokens, which in this example is *Barker*. Finally, when it retrieves the <*END*> token, the generation ends, and the generated output is the retrieval sequence of the query.

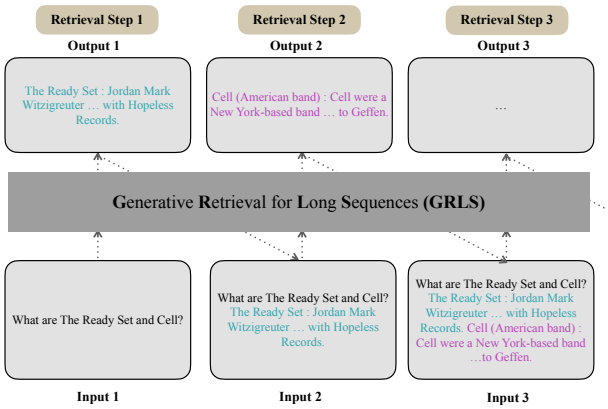


Figure 2: Multi-step retrieval process of GRLS. To retrieve a k-th step sequence, previously retrieved sequences ($1, 2, .., k-1$-th step) are concatenated to a query and used as new input. In the example above, when a query *"What was The Ready Set and Cell?"* is given in the first retrieval step, the model predicts the output about *Ready Set* and appends it to the query to use it as the second retrieval step input. The passage about *Cell* is retrieved in the second step and appended to the input, etc. Special tokens are omitted in the above example because of the space limit.

Previous work (Cao et al., 2021) is limited to retrieving short sequences such as entities in a single-step setting. In this section, we describe our approach for generalizing it to multi-step setting (Section 3.1) and longer sequences through more efficient constrained beam search (Section 3.2) and pre-finetuning memorization of the retrieval target corpus (Section 3.3). We name our model as **GRLS** (**G**enerative **R**etrieval for **L**ong **S**equences).

3.1 Multi-Step Generative Retrieval

In this work, we generalize generative retrieval to cover not only single-step retrieval (Cao et al., 2021) but also *multi-step* retrieval (Figure 2). In single-step retrieval tasks, sequences can be retrieved by only conditioning on the input, whereas in multi-step retrieval tasks, additional conditioning on the previously retrieved sequences is necessary to retrieve the next retrieval sequence (Appendix A.1, A.2). For the additional conditioning, we modify the aforementioned objective function of $P_\theta(p|q)$; the model is trained to generate the tokens $y_i^{(t)}$ of the ground truth text to retrieve $p^{(t)}$ at

retrieval step $t = 1, \cdots, T$, given the query q to the encoder and all of the generated tokens up to the previous step, $y_{<i}^{(<t)}$ as the input to the decoder.

$$\text{score}(q, p^{(1)}, \cdots, p^{(T)}) = \prod_{t=1}^{T} P_\theta(p^{(t)}|p^{(<t)}, q)$$
$$= \prod_{t=1}^{T}\prod_{i=1}^{L^{(t)}} P_\theta(y_i^{(t)}|y_{<i}^{(<t)}, q)$$

Note that $T = 1$ for single-step retrieval.

At the inference step of multi-step retrieval, we generate the retrieval sequences with a beam size ≥ 1. Retrieval sequence ($p^{(t)}$) with the highest score is retrieved and is added to the end of the decoder input. The process continues until a special token *DONE* is generated, which means the generation process also decides when to stop retrieving more items (*dynamic multi-step*). For better comparison with some of previous work (Yang et al., 2018; Saha et al., 2021; Xiong et al., 2021), we also consider *fixed multi-step*, where the number of items to retrieve is fixed for the entire task.

3.2 Efficient Constrained Beam Search

At the inference step, we use constrained beam search proposed in Cao et al. (2021), which is a modified beam search (Sutskever et al., 2014) algorithm that masks out tokens that form the texts that do not exist in the corpus. The purpose of constrained beam search is to ensure that all retrieval sequences are from the given corpus. Specifically, a prefix tree is built by aggregating tokenization results of text in the corpus (the rightmost component in Figure 1). Tokens that create strings that are not a sub-string of any text in the corpus are masked out, and only the next top-k tokens from the unmasked and thus valid set of tokens are passed to the model as the potential next tokens list.

As the texts in the corpus have become longer sequences, the height of the prefix tree (h), which is the maximum length of retrieval sequences in the corpus, becomes very long. Therefore, the search time complexity of finding the potential next tokens ($O(h)$) become nontrivial. By flattening all the paths into a separate key and the potential next tokens list as the corresponding values, the search time complexity reduces to $O(1)$, showing an average of 56% inference time reduction on HotpotQA.[1]. Details are in Appendix A.3.

[1]We call this using the same term, prefix tree, for the rest of the paper

3.3 Retrieval Corpus Memorization

Since generative retrieval generates sequences in a uni-directional way (left to right), the models cannot know the information at the end of a sequence in advance. This may negatively affect the performance, especially when the length of the texts (sequences) in the corpus gets longer. To solve the issue, we perform retrieval corpus memorization before training on the target retrieval task.

During the process, encoder-decoder model is trained on texts in the corpus using the standard language modeling objective function: when a corpus C with texts p ($p \in C$) is given, the model learns to maximize the language modeling probability $P_\theta(p) = \prod_{i=1}^{L} P_\theta(y_i|y_{<i})$ for all p in C. This can help the models recognize the contents to generate at the later part of the text beforehand and improve the performance in certain cases as shown in Section 5. To make the input similar to that of the real task, which maximizes $P_\theta(p|q) = \prod_{i=1}^{L} P_\theta(y_i|y_{<i}, q)$ such that q is input to the encoder, the front part of the text to generate serves as the encoder input when maximizing $P_\theta(p)$. Relevant details are in Appendix A.3.

4 Experimental Setup

We describe the overall experimental setup in the following section. In Section 4.1, we show a brief explanation of the six datasets we use and the setting for training and inference. In Section 4.2, we explain the bi-encoder retrieval models we use to compare the performance with GRLS. In Section 4.3, we describe the metric to evaluate the performance. We explain the hyperparameter setting of the model in Appendix A.4.

4.1 Datasets

We use six datasets with various characteristics: different number of hops, unseen rate, corpus size, average retrieval steps, and granularity. Table 1 shows the overall statistics and the features of the datasets. Below are brief descriptions of each dataset. Appendix A.2 and A.5 show examples and detailed description on the train and test settings of each dataset.

Natural Questions (NQ) Kwiatkowski et al. (2019) propose a single-hop open domain question answering dataset, where the questions are mined from real google search queries, and the answers

Table 1: Overview of the six datasets. **Seq Len** column shows the average number of retrieval sequence tokens. **Step** column shows the average number of retrieval steps for a query in the test set. **Unseen** column shows the rate of test queries consisting of only the retrieval sequences unseen during the training process. Details of the datasets are in Section 4.1

Dataset	Corpus (MB)	Seq Len	Step	Unseen
NQ	13,252	160.8	1	55.8%[a]
HotpotQA	1,595	78.6	2	18.9%
EntailBank	0.7	12.5	4.6	2.7%
StratgyQA	7.0	13.1	2.7	98.2%
Explagraphs-Open	0.5	9.6	4.5	95.5%
RuleTaker-Open	0.7	13.1	-	0.0%[b]

[a]For the overlap calculation, we only use the subset of test dataset where gold evidence is provided (6515 out of 8757 test datasets).

[b]We calculate the rate with prediction result (retrieval sequences) since there is no gold retrieval sequences.

are passages of Wikipedia articles[2].

HotpotQA Yang et al. (2018) propose an open domain multi-hop question answering dataset, which requires seeing multiple Wikipedia passages through logical reasoning or sequential processing. The number of retrieval sequences is fixed to two.

Entailment TreeBank (EntailBank) Dalvi et al. (2021) propose a reasoning tree construction task where it forms a tree with the hypothesis as the root node and evidence sentences are leaf nodes. We experiment on the leaf node retrieval of Task3: retrieval of leaf nodes (sentence) from the given corpus given a question and an answer as the input. We call the dataset EntailBank in short.

StrategyQA Geva et al. (2021) propose a multi-hop open-domain question answering dataset where the reasoning steps are implicit in the question, and thus relevant strategies are required to answer the question. Given a question, the model retrieves the evidence sentences from the corpus.

RuleTaker-Open Clark et al. (2021) propose a synthetic rule-based dataset to measure the model's reasoning ability over the rules expressed in natural language. Based on the released dataset, we create a new task, RuleTaker-Open, to make the task close to a real-world setting. Given a query, the model retrieves nodes of the graph, which are sentences from the corpus, and the nodes are connected in order to construct a graph.

[2]We add title in front of each passage for NQ and Hotpot corpus (Karpukhin et al.; Izacard and Grave, 2021)

Explagraphs-Open Saha et al. (2021) propose a generative and structured commonsense-reasoning task. We reformulate the task to open-domain retrieval setting and name it Explagraphs-Open, considering a single path (subject-relation-object) as a retrieval sequence.

4.2 Bi-Encoder Retrieval Models

For each dataset, we compare the results with bi-encoder retrieval models as the baselines. For NQ and HotpotQA datasets, we use DPR and MDR, respectively, which are widely used bi-encoder retrieval models for the corresponding dataset. For the rest of the datasets, we compare with Sentence-T5 (ST5), a bi-encoder retrieval model using T5 (Raffel et al., 2020).

DPR Karpukhin et al. is a simple bi-encoder retriever trained with in-batch negatives and a few hard negatives selected with BM25.

MDR Xiong et al. (2021) propose an iterative bi-encoder retrieval model, MDR, which extends DPR to a multi-step setting.

ST5 ST5 is a encoder-decoder model (Ni et al., 2021)[3] that uses the first decoder output as the sequence embedding. It serves as the base architecture of our baseline bi-encoder to compare the performance with GRLS using the same number of parameters.

Baseline Bi-Encoder Retriever (BE) In order to compare the multi-step generative retrieval to a bi-encoder retrieval, we create a simple counterpart such that the bi-encoder retrieval can be as well adapted to fixed multi-step and dynamic multi-step retrieval tasks. For fixed multi-step retrieval, we train the bi-encoder (BE) to maximize $P_\theta(p^{(t)}|p^{(<t)}, q)$ like GRLS, but by concatenating the query q and the retrieval sequences of the previous steps $p^{(<t)}$ to make the input to the query encoder at step t like in MDR (Xiong et al., 2021). For dynamic multi-step retrieval, we add the special single-token text *DONE* to the corpus as done in GRLS. When training the model, one extra retrieval step is added at the end as well; at the point when the retriever retrieves all the target texts, the model has to retrieve *DONE* text using MIPS. At inference, the model retrieves texts until it retrieves the special token or the number of retrieval reaches the predefined maximum retrieval step. Details are in Appendix A.8.

[3]We use ST5-EncDec from Ni et al. (2021)

Table 2: Retrieval sequence recall rate (R@5) of both single-step (s-*) and multi-step (m-*) method on test set. BE is the result of ST5. The bold text shows the best score and the underline text shows the second best score of the dataset.

Model	EntailTree	StrategyQA	Explagraphs-Open
s-BE	36.0	**56.6**	**40.8**
m-BE	31.5	37.4	27.0
s-GRLS	44.8	43.0	27.3
s-GRLS + mem	44.1	45.0	27.0
m-GRLS	<u>53.6</u>	44.9	<u>32.9</u>
m-GRLS + mem	**54.3**	<u>45.5</u>	32.4

Table 3: Retrieval sequence F1 score of dynamic multi-step retrieval on test set. For scores marked with *, we use metric in Appendix A.6 instead of F1 since there is no ground truth evidence set. We fix the maximum retrieval step to 20[4]. **Missing *DONE*** shows the missing rate of retrieving the *DONE* token before the maximum retrieval step. The bold text shows the best F1 score and lowest Miss *DONE* rate.

Dataset	Model	F1	Missing *DONE*
EntailTree	BI	16.9	5.6%
	GRLS	**52.5**	**2.4%**
	GRLS + mem	52.2	2.7%
StrategyQA	BI	36.5	39.0%
	GRLS	46.6	23.2%
	GRLS + mem	**47.1**	**22.4%**
Explagraphs-Open	BI	25.4	28.2%
	GRLS	**41.5**	5.0%
	GRLS + mem	41.3	**0.3%**
RuleTaker-Open	BI	17.0*	39.0%
	GRLS	51.0*	24.5%
	GRLS + mem	**65.5***	**23.0%**

4.3 Evaluation Metric

In *a fixed-step setting*, for NQ and HotpotQA, we follow the evaluation metric of the bi-encoder model we compare: answer recall (Karpukhin et al.) and retrieval sequence recall (Xiong et al., 2021) respectively. For multi-hop datasets with varying numbers of ground truth retrieval steps (Explagraphs-Open, EntailBank, and StrategyQA), we first calculate the retrieval sequence recall rate of each query and average over the number of queries (Dalvi et al., 2021; Saha et al., 2021). Furthermore, in *a dynamic multi-step retrieval setting*, since the number of predicted retrieval sequences varies, we measure the retrieval sequence F1 score. For RuleTaker-Open, we newly define an evaluation metric (Appendix A.6) that measures the graph construction success rate since we do not have the ground truth retrieval sequences information.

5 Experimental Results

The main contribution of this work is that we generalize the generative retrieval proposed by Cao et al. (2021) to longer retrieval sequences (reasoning path, sentence, passage) and various tasks (single-step, fixed multi-step, dynamic multi-step). Section 5.1 explores when GRLS performs well. In Section 5.2, we analyze whether Generative Retrieval for Long Sequences (GRLS) and bi-encoder retriever have different characteristics and show that a simple ensemble method can often boost the performance. Lastly, Section 5.3 shows the efficiency of GRLS compared to bi-encoder retrieval.

5.1 When does generative retrieval perform well?

Table 2 shows the retrieval sequence recall rate of bi-encoder (BE) and GRLS variants on three fixed multi-hop retrieval datasets where the task is sentence or reasoning path retrieval.[5] Retriever variants of single-step and multi-step retrieval are tested with or without corpus memorization. In summary, the results show that GRLS often outperforms bi-encoder retrievers when the ratio of retrieval sequences unseen during training is low, the number of retrieval steps is high, and dynamic multi-step retrieval is required.

Effect of Unseen Rate The unseen rate indicates the rate of queries in the test set, which requires the retrieval of the sequences never seen during training as the ground truth target. Therefore, the datasets with high unseen rates can be considered similar to a zero-shot retrieval setting. The degree of unseen rate in Table 1 and the performance of BE and GRLS in Table 2 shows GRLS can outperform the bi-encoder retriever when the unseen rate is low, which implies that it is crucial for GRLS to pretrain on the retrieval targets (also see Appendix A.9 and A.10 for analysis on EntailBank).

Single- vs. Multi-Step Table 2 also shows that GRLS consistently performs better when applied with a multi-step approach rather than a single-step approach, while the trend is opposite for the case of BE. It is worth noting that the multi-step approach goes through multiple iterations of retrieval by appending the previous output to the current input. Unlike bi-encoder retrieval, which suffers

[5] RuleTaker-Open is excluded because recall cannot be calculated as the dataset lacks ground-truth sequence to retrieve at each retrieval step.

from information loss by condensing the texts into a fixed-size vector (Luan et al., 2021), generative retrieval can efficiently cross-encode the input and output. We, therefore, assume that when there is high connectivity between the input and the output or between the outputs, GRLS may show high performance by capturing the relation between the inputs from multiple steps.

Single-Step BE Analysis The fact that the best models for StrategyQA and Explagraphs-Open (multi-hop datasets) are single-step BE is counter-intuitive. Therefore, we perform a manual analysis of the datasets. Appendix A.11 shows that about 2/3 of the randomly sampled queries from StrategyQA and Explagraphs-Open can be answered by looking at only the query (similar to comparison type questions rather than bridge type questions in HotpotQA). [6] Moreover, as the unseen rates of the two datasets are very high (near 100%), the effect of error propagation from multi-step iterative retrieval could have been destructive, even offsetting the benefits from the iterative approach.

Dynamic Multi-Step Retrieval As described in Section 3.1, dynamic multi-step retrieval has several benefits. The results in Table 3 show that GRLS is good at capturing where to stop retrieval with fewer cases of missing the *DONE* token (which decides the point to stop the retrieval) and a higher F1 score than BE on EntailTree, StrategyQA, and Explagraphs-Open. Also, on RuleTaker-Open, where the task is constructing a reasoning graph, the success rate[7] of GRLS on constructing the reasoning graph outperforms BE by a large margin.

HotpotQA and NQ We further test whether the advantages of generative retrieval generalize to passage retrieval tasks: HotpotQA and NQ (Table 5 and 6). First, the recall of multi-step GRLS is much higher than single-step GRLS on HotpotQA[8]. In addition, the performance of best GRLS models is comparable to MDR- and DPR-random in both NQ and HotpotQA. Note that MDR and DPR, which are different in that they are trained with *hard* negative samples, significantly outperform GRLS. Unlike bi-encoder retrieval, it is not obvious how hard negative samples should be used during training for generative retrieval and is thus an important future direction to close the gap.

Effect of Corpus Memorization We analyze whether corpus memorization is helpful for performance. We find that memorization is constantly helpful in NQ (Table 6) and StrategyQA (Table 2, 3), while the gain is inconsistent in other datasets. Note that NQ is a single-hop dataset, and although StrategyQA is a multi-hop dataset, a single-step approach is sufficient to perform well on the dataset

Table 4: BE-BE is a homogeneous ensemble model between two single-step BE and GRLS-GRLS is a homogeneous ensemble model between two multi-step GRLS. BE-GRLS is an ensemble model of two different approaches: single-step bi-encoder and multi-step GRLS. We score the retrieval sequence recall rate (R@5) on test set.

Model	EntailBank	StrategyQA	Explagraphs-Open
GRLS-GRLS	**54.6**	47.3	33.1
BE-BE	38.5	58.4	42.2
GRLS-BE	53.5	**61.3**	**42.8**

Table 5: Retrieval sequence recall rate of HotpotQA official full-wiki dev set. Scores except for GRLS and GRLS +* are from Table 3 of Xiong et al. (2021). s-* indicates single-step and and m-* indicates multi-step. MDR- indicates a variant of MDR without linked negatives, memory bank, and shared encoder.

Method	Top-2	Top-10	Top-20
DPR	25.2	45.4	52.1
MDR-	59.9	70.6	73.1
MDR	65.9	77.5	80.2
s-GRLS	11.3	23.5	29.5
s-GRLS + mem	10.2	22.9	26.6
m-GRLS	57.7	68.8	73.9
m-GRLS + mem	55.0	65.3	71.4

Table 6: Answer recall rate of NQ dev set. All models are single-step retrievers. Scores of DPR-* and DPR are from Table 3 of Karpukhin et al.. DPR-* is the score without in-batch training and * is the method of finding 7 negative sentences.

Method	Top-5	Top-20	Top-100
DPR-Random	47.0	64.3	77.8
DPR-BM25	50.0	63.3	74.8
DPR-GOLD	42.6	63.1	78.3
DPR	65.8	78.0	84.9
GRLS	46.5	60.0	70.2
GRLS + mem	46.5	61.3	70.4

[6] Details of comparison and bridge type questions are in Appendix A.5. StrategyQA has rationale types where a multi-step method is not necessary, and Explagraphs-Open, though the structure of evidence sentences is a reasoning graph as in RuleTaker-Open, has various topics which the evidence texts could be retrieved by the topic of the query itself without the previous retrieval texts.

[7] F1 cannot be calculated on RuleTaker-Open because the ground-truth retrieval sequence is not known at each step.

[8] The effectiveness of multi-step on HotpotQA can be seen from the large gap of performance between DPR (single-step) and MDR (multi-step expansion of DPR).

as described in "Single-Step BE Analysis".[9] We hypothesize that corpus memorization is more effective on these single-step datasets than multi-step ones because the training objective function for memorization is not consistent with the multi-step setting. That is, the memorization objective function $P_\theta(p) = \prod_{i=1}^{L} P_\theta(y_i|y_{<i})$ resembles the single-step retrieval training objective function of maximizing $\text{score}(q,p) = P_\theta(p|q)$ rather than that of the multi-step retrieval, $\text{score}(q, p^{(1)}, \cdots, p^{(T)})$, which goes through multiple retrieval steps. We leave the exploration of a better memorization strategy for multi-step retrieval as a future work.

5.2 Do GRLS and BE behave differently?

By analyzing the result of the GRLS and BE, it can be seen that the sequences that the two models retrieve have different characteristics. First, we compare the top-2 prediction of GRLS and MDR (bi-encoder retriever) on HotpotQA. Appendix A.12 shows that MDR mostly gets wrong by failing to retrieve the second hop target even though the first hop prediction is correct, whereas GRLS mostly gets wrong when the first-hop target is not explicitly expressed in the query. Second, on RuleTaker-Open (Appendix A.6), GRLS shows a higher success rate with a more complex and diverse reasoning graph, suggesting that GRLS is strong at retrieving highly structured items such as reasoning chains and graph relations.

From the observation that GRLS and BE retrieve sequences with different aspects, we compare the performance of ensembles of the two models. We use a simple ensemble method of considering the retrieved sequences from two ensembles models one-by-one, starting from the sequence retrieved at the top. We use this simple iterative prediction aggregation method because BE and GRLS have different scoring methods.[10]

Figure 7 in Appendix A.13 shows that BE and GRLS tend to retrieve different sequences. Table 4 shows that when BE and GRLS are ensembled, in most cases, it results in better performance than the ensembles of the same type of models (homogeneous ensembles). Appendix A.14 further shows that the gap between different retrieval types (bi-encoder retrieval and generative retrieval) is generally larger than the gap between different retrieval steps (single and multi-step retrieval). In summary, bi-encoder retrieval and generative retrieval can be complementary to the other, as the ensemble of the two is often more advantageous than a homogeneous ensemble.

5.3 Efficiency of GRLS

We compare two bi-encoder retrievers (ST5, MDR) and GRLS on their time complexity and GPU memory usage. For offline computation, bi-encoder retrievers (BE) need to create a large index of embeddings and store it in GPU, whereas GRLS only needs to build a prefix tree. This gives generative retrieval both time and GPU memory efficiency advantage; GRLS with optimization is about 100 times faster and uses 79.5% less GPU memory than ST5 (FP32) with the same number of parameters.

During inference time, GRLS can be time-inefficient if it has to generate every word in the retrieval target text. In practice, however, one can stop generation as soon as the partially generated text can uniquely identify the target text. By leveraging this optimization, GRLS with greedy search is able to achieve 40% inference time reduction with respect to ST5 (FP32) with the same number of parameters. Note that in the absence of the optimization process, GRLS is 24.6 times slower than ST5, signifying the importance of early stopping.

6 Conclusion

We show that generative retrieval, which has been originally proposed for retrieving short sequences such as entities, can also be considered for retrieving longer sequences. We particularly find that generative retrieval can have an advantage over bi-encoder in certain situations, such as retrieving structured information (e.g., reasoning chains or graphs) and retrieving an arbitrary number of items. Given that generative retrieval inherently has GPU memory and speed benefits, it can be a practical alternative for general retrieval tasks in the future.

[9] Although Explagraphs-Open can also be tackled with a single-step approach, it still has more multi-hop characteristics than StrategyQA. For example, comparing the relative performance difference of multi and single-step GRLS in Table 2, it is about 20% on EntailBank and Explagraphs-Open, while only 1% on StrategyQA.

[10] BE uses the inner product between two dense embeddings, and GRLS uses the aggregation of generation probabilities. While a more sophisticated method such as using a re-ranker, calibration, or interpolation of the two model predictions can be used when the scoring method is different (Seo et al., 2019; Cheng et al., 2021), we use a simple ensemble method. This is to eliminate the effect of another model or hyperparameter choice from the analysis.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *ICLR*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Qianglong Chen, Feng Ji, Haiqing Chen, and Yin Zhang. 2020. Improving commonsense question answering by graph-based iterative retrieval over multiple knowledge sources. In *COLING*.

Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2021. Unitedqa: A hybrid approach for open domain question answering. In *ACL-IJCNLP*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. In *IJCAI*.

Bhavana Dalvi, Peter Alexander Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *EMNLP*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *TACL*.

Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *Vldb*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun KIM, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards continual knowledge learning of language models. In *ICLR*.

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 *EMNLP*.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *TACL*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *TACL*.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *CoRR*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Stephen Robertson. 2008. On the history of evaluation in ir. *Journal of Information Science*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric Michael Smith, Y.-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *EACL*.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. Prover: Proof generation for interpretable reasoning over rules. In *EMNLP*.

Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. Explagraphs: An explanation graph generation task for structured commonsense reasoning. In *EMNLP*.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NeurIPS*.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the ACL-IJCNLP*.

Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book qa? In *ACL-IJCNLP*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Ledell Yu Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *EMNLP*.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. Worldtree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *LREC*.

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 *ICLR*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

A Appendix

A.1 Multi-Step Retrieval Examples

There are many cases where multi-step retrieval is necessary for multi-hop datasets because one output affects the selection of the subsequent output. For the first example of Table 8, to find the answer, we first need to look at what music **Die Rhöner Säuwäntzt** played and then find where the music originated from. Similarly, for the second example of Table 8, to find the answer, we need to find who was starred in **Gunmen from Laredo** and then find who narrated **the Frontier**.

A.2 Dataset Examples

Examples of each dataset (input and output forms) are in Table 9.

Algorithm 1 Constrained Beam Search

Require: retriever, input query q and prefix tree T, which is a hash where the key is a token_id and the value is the subtree taking the corresponding token_id (key) as the root
retrieval_sequence := An empty list to store the token ids of the sequence to retrieve
next_token_id := NULL

while next_token_id is not <*END*> and *retrieval_sequence*.length < max_length **do**
  token_prob_list := retriever (q, *retrieval_sequence*)

  // $O(h)$
  for all token_id in *retrieval_sequence* **do**
    T := T[token_id]
  end for

  potential_next_token_ids := T.keys
  potential_next_token_prob_list := filter(
    token_prob_list, potential_next_token_ids)
  next_token_id := argmax(
    potential_next_token_prob_list)
  retrieval_sequence.append(next_token_id)
end while

return *retrieval_sequence*

Algorithm 2 Efficient Constrained Beam Search

Require: retriever, input query q and efficient prefix tree $\mathfrak{T}$, which is a hash where the key is a list of token_ids and the value is a list of potential next tokens
retrieval_sequence := An empty list to store the token ids of the sequence to retrieve
next_token_id := NULL

while next_token_id is not <*END*> and *retrieval_sequence*.length < max_length **do**
  token_prob_list := retriever (q, *retrieval_sequence*)

  // $O(1)$
  if *retrieval_sequence*.length = 0 **then**
    potential_next_token_ids := [$\mathfrak{T}$.root]
  else
    potential_next_token_ids
    := $\mathfrak{T}$[*retrieval_sequence*]
  end if

  potential_next_token_prob_list := filter(
    token_prob_list, potential_next_token_ids)
  next_token_id := argmax(
    potential_next_token_prob_list)
  retrieval_sequence.append(next_token_id)
end while

return *retrieval_sequence*

A.3 Details of GRLS

Constrained Beam Search We use a T5-large tokenizer from huggingface (Wolf et al., 2020) when constructing prefix tree. By using efficient constrained beam search, it shows an average of 56% efficiency on inference time, but only a 16% increase in offline time complexity due to

Table 7: Single-Step and Multi-Step Examples on EntailBank, Explagraphs-Open, and StrategyQA. The bold texts in multi-step examples show parts where it cannot be retrieved by single-step retrieval.

Step	Dataset	Input	Output
Single-Step	EntailBank	Which energy source is considered nonrenewable? fossil fuel	fossil fuels are a nonrenewable resource, fossil fuels are a nonrenewable resource, an energy source is a kind of resource
	StrategyQA	Is Freya a combination of Athena and Aphrodite?	Athena was the Greek goddess of war, Aphrodite was the Greek goddess of love, Freya was the Norse goddess of war, love, and fertility
	Explagraphs-Open	belief: Entrapment shouldn't be legalized since it puts people into false situations. argument: Entrapment is really a trick	entrapment is trick, trick is false situations, trick not capable of be legalized
Multi-Step	EntailBank	Which of these is a way the people of Virginia can help restore a natural ecosystem? Plant native plants	planting native plants has a positive impact on an ecosystem, to restore means to return to a better state, **better means good**, **good means positive**, **helping something has a positive impact on that something**
	StrategyQA	Was the first Vice President of the United States an Ottoman descendant?	The first Vice President of the United States was John Adams, The Ottomans were a Turkic group that conquered Constantinople in 1453, **John Adams was descended from English Puritansy**
	Explagraphs-Open	belief: Racial profiling is biased against anyone who isn't white. argument: Racial profiling is not an acceptable way to codify people as criminals.	racial profiling; is a; prejudiced, **prejudiced; synonym of; biased**, **prejudiced; has context; who isn't white**, racial profiling; is not a; acceptable

Table 8: Cases where multi-step retrieval is necessary.

Input	Output
Where did the form of music played by **Die Rhöner Säuwäntzt** originate?	Output 1
	<TITLE> **Skiffle** </TITLE> Skiffle is a music genre with jazz, blues, folk and American folk influences, usually using a combination of manufactured and homemade or improvised instruments. **Originating as a term in the United States** in the first half of the 20th century, it became popular again in the UK in the 1950s, where it was associated with artists such as Lonnie Donegan, The Vipers Skiffle Group, Ken Colyer and Chas McDevitt. Skiffle played a major part in beginning the careers of later eminent jazz, pop, blues, folk and rock musicians and has been seen as a critical stepping stone to the second British folk revival, blues boom and British Invasion of the US popular music scene.
	Output 2
	<TITLE> **Die Rhöner Säuwäntzt** </TITLE> Die Rhöner Säuwäntzt are a Skiffle-Bluesband from Eichenzell-Lütter in Hessen, Germany. The line-up consists of Martin Caba, Christoph Günther and Christoph Leipold playing **Skiffle-Blues** with lyrics based on Rhön Mountains dialect and other Hessian dialects varieties. The expression "Säuwäntzt" means pork belly and refers also to untidy or unruly children and youth.
Gunmen from Laredo starred which narrator of "Frontier"?	Output1
	<TITLE> **Walter Coy** </TITLE> Walter Darwin Coy (January 31, 1909 – December 11, 1974) was an American stage, radio, film, and, principally, television actor, originally from Great Falls, Montana. He was best known for **narrating the NBC western anthology series, "Frontier"**, which aired early Sunday evenings in the 1955–1956 season.
	Output2
	<TITLE> **Gunmen from Laredo** </TITLE> Gunmen from Laredo is a 1959 American western film produced and directed by Wallace MacDonald, which stars Robert Knapp, Maureen Hingert, and **Walter Coy**.

its overhead of splitting all paths into separate keys.

Multi-Step Retrieval Method We add special tokens <*QUESTION*> and </*QUESTION*> at the front and the end of the input query. When adding the output sequence at the end of the input from the second multi-step retrieval step, special tokens <*EVIDENCE*> and </*EVIDENCE*>

Table 9: Dataset examples

Task	Input	Output
Single-Hop Paragraph Retrieval (NQ)	who was the comic book series amar chitra katha created	<TITLE> Amar Chitra Katha </TITLE> Amar Chitra Katha Amar Chitra Katha (ACK, ""Immortal Captivating (or Picture) Stories"") is one of India's largest selling comic book series, with more than 100 million copies sold in 20 Indian languages. Founded in 1967, the imprint has more than 400 titles that retell stories from the great Indian epics, mythology, history, folklore, and fables in a comic book format. It was created by Anant Pai, and published by India Book House. In 2007, the imprint and all its titles were acquired by a new venture called ACK Media. On 17 September 2008, a new website by ACK-media was launched
Multi-Hop Paragraph Retrieval (HotpotQA)	Step 1 Input (a query)	Step 1 output (evidence passage)
	<QUESTION> The Oberoi family is part of a hotel company that has a head office in what city? </QUESTION>	<TITLE> Oberoi family </TITLE> The Oberoi family is an Indian family that is famous for its involvement in hotels, namely through The Oberoi Group.
	Step 2 Input (a query with previous output)	Step 2 Output (evidence passage)
	<QUESTION> The Oberoi family is part of a hotel company that has a head office in what city? </QUESTION> <EVIDENCE> <TITLE> Oberoi family </TITLE> The Oberoi family is an Indian family that is famous for its involvement in hotels, namely through The Oberoi Group. </EVIDENCE>	<TITLE> The Oberoi Group </TITLE> The Oberoi Group is a hotel company with its head office in Delhi. Founded in 1934, the company owns and/or operates 30+ luxury hotels and two river cruise ships in six countries, primarily under its Oberoi Hotels & Resorts and Trident Hotels brands.
Multi-Hop Sentence Retrieval (EntailmentBank, StrategyQA)	Step 1 Input (a query)	Step 1 output (evidence sentence)
	<QUESTION> Does a dentist treat Bluetooth problems? </QUESTION>	A dentist is a surgeon who specializes in dentistry, the diagnosis, prevention, and treatment of diseases and conditions of the oral cavity
	Step 2 Input (a query + Step 1 Output)	Step 2 Output (evidence sentence)
	<QUESTION> Does a dentist treat Bluetooth problems? </QUESTION> <EVIDENCE> A dentist is a surgeon who specializes in dentistry, the diagnosis, prevention, and treatment of diseases and conditions of the oral cavity </EVIDENCE>	Technological problems are typically handled by IT professionals
	Step 3 Input (a query + Step 1 & Step 2 Output)	Step 3 Output (evidence sentence)
	<QUESTION> Does a dentist treat Bluetooth problems? </QUESTION> <EVIDENCE> A dentist is a surgeon who specializes in dentistry, the diagnosis, prevention, and treatment of diseases and conditions of the oral cavity </EVIDENCE> <EVIDENCE> Technological problems are typically handled by IT professionals </EVIDENCE>	Bluetooth is not a physical entity
Multi-Hop Reasoning Path Retrieval (RuleTakers, Explagraphs)	Step 1 Input (a query)	Step 1 output (evidence sentence)
	<QUESTION> belif: marriage is the best for a family unit. argument: Marriage is a predictor of health and happiness. </QUESTION>	marriage; created by; love
	Step 2 Input (a query + Step 1 Output)	Step 2 Output (evidence sentence)
	<QUESTION> belif: marriage is the best for a family unit. argument: Marriage is a predictor of health and happiness. </QUESTION> <EVIDENCE> marriage; created by; love </EVIDENCE>	love; causes; health and happiness
	Step 3 Input (a query + Step 1 & Step 2 Output)	Step 3 Output (evidence sentence)
	<QUESTION> belif: marriage is the best for a family unit. argument: Marriage is a predictor of health and happiness. </QUESTION> <EVIDENCE> marriage; created by; love </EVIDENCE> <EVIDENCE> love; causes; health and happiness </EVIDENCE>	health and happiness; used for; family unit

are added at the front and the end of the output sequence. (Wang et al., 2021)

Retrieval Corpus Memorization Step For the path retrieval task (RuleTaker-Open, Explagraph-Open), the subject and the relation are given, and the model generates the object and for the sentence and paragraph retrieval task (NQ, HotpotQA, EntailBank, StrategyQA), the first 70% of the sentence is given as input, and the model generates the rest.

A.4 Experimental Setup Details

We train both ST5 and GRLS using pre-trained T5-large checkpoint from Wolf et al. (2020) as the initial checkpoint. We use the same hyperparameter setting when training GRLS and ST5 model for a fair comparison. We observe that hyperparameter change does not change the tendency of results after experimenting over a combination of settings used in previous models (Karpukhin et al.; Ni et al., 2021; Raffel et al., 2020). Also, we use different hyperparameters for different tasks: retrieval corpus memorization and retrieval. For all experiments, we use 8 32GB V100 GPUs.

Retrieval Corpus Memorization The retrieval corpus memorization step aims to show GRLS a corpus it will retrieve and save it implicitly before the retrieval step. We keep the learning rate to 1e-5, which is relatively low than the retrieval step, to maintain the linguistic ability the model learned during pre-training (Jang et al., 2022). We train the model from T5 pre-trained checkpoint for every dataset using Adafactor with a constant learning rate of 1e-5 with batch size 240 till the maximum of 3 epochs.

Increasing the retrieval corpus memorization epoch does not always lead to higher performance. This is because as the model is trained on a new dataset, catastrophic forgetting of previously learned parts occurs (Kirkpatrick et al., 2017), and in this case, language ability of the model learned during the pre-training step. To prevent the following process from occurring, we follow Jang et al. (2022) and reduce the learning rate to 1e-5 and could observe that using the retrieval corpus memorization step of about epoch 3 as the initial checkpoint leads to the largest improvement on reasoning task.

Retrieval Step The retrieval step aims to retrieve the gold item from a large-scale corpus. For datasets where the retrieval corpus memorization step help improve performance (NQ, StrategyQA, RuleTaker-Open, and Explagraphs-Open), we use the checkpoint from the retrieval corpus memorization step. For the rest of the datasets, we use the T5 pre-trained checkpoint as the initial checkpoint. For both ST5 and GRLS, we train using Adafactor with a learning rate 1e-4 with a linear warm-up for the first 10% of training and then linear decay with batch size 120 till maximum 30 epochs.

A.5 Dataset Details

Natural Questions (NQ) Kwiatkowski et al. (2019) propose a single-hop open domain question answering dataset where the questions are mined from real google search queries, and the answers are passages of Wikipedia articles. We use the train/dev/test split and Wikipedia dump from Karpukhin et al.. A single-hop dataset requires a single piece of text evidence to answer the question. When the input query is given, it retrieves the evidence paragraph from the corpus.

HotpotQA Yang et al. (2018) propose an open domain multi-hop question answering dataset, which requires aggregating multiple Wikipedia passages through logical reasoning or sequential processing. The number of retrieval sequences is fixed to two. HotpotQA consists of two types of questions: comparison and bridge. Comparison questions, a rationale/evidence type of multi-hop dataset, do not necessitate iterative retrieval since the two entities can be retrieved by the query itself. However, bridge questions consist of evidence in the reasoning chain from where it has to retrieve the second step based on the first one. We use the official Wikipedia dump provided by Yang et al. (2018), use 2% of the official train dataset as a dev set, and report the scores on the official dev set.

Entailment TreeBank (EntailBank) Dalvi et al. (2021) propose a reasoning tree construction task where it forms a tree with a hypothesis as the root node and evidence sentences are leaf nodes. The dataset has three settings, and among them, we experiment on Task3, an open setting. Task3 consists of two steps; the first is to select a leaf node from the corpus set when given a question and an answer, and the second is to construct a reasoning tree through the selected leaf node. We perform the first step, the leaf node retrieval. Since the leaf

node and the root node are not directly connected, there is a less tight connection between the input query and gold outputs than other datasets. We experiment on the first step of Task3 (leaf node retrieval). As in the paper, we use both EntailBank and WorldTreeV2 (Xie et al., 2020) datasets when training a retrieval model. We compare the results with ST5 since there is no released bi-encoder model, and as in the paper, we use both EntailBank and WorldTreeV2 (Xie et al., 2020) datasets when training a retrieval model.

StrategyQA Geva et al. (2021) propose a multi-hop open-domain question answering dataset where the reasoning steps are implicit in the question and need some strategy to answer the question. When given a question, the model retrieves the evidence sentences from the corpus. Since only the train dataset contains evidence annotation, we split it into 75/5/20 (%) and used it as a train/val/test set, respectively. Also, based on the given corpus, we split the given paragraph-level corpus to sentence level using NLTK (Bird et al., 2009) to match the granularity of the evidence and add the annotated evidence sentences to the corpus.

RuleTaker-Open Clark et al. (2021) propose a synthetic rule-based dataset to measure the model's reasoning ability over the rules expressed in natural language. Based on the released dataset, we create a new task, RuleTaker-Open, to make the task close to a real-world setting. Given a query, the model retrieves nodes of the graph, which is a sentence from the corpus, and the nodes are connected in order to construct a graph. Details of the construction method are described in Appendix A.6.

Explagraphs-Open Saha et al. (2021) propose a generative and structured commonsense-reasoning task. When given a belief and an argument, a model predicts whether the argument supports or counters the belief and generates (retrieves) a reasoning graph to explain the prediction. While the original dataset needs generation on constructing the reasoning graph, which is limited to generative model only, we expand the task to an open-domain retrieval setting to compare with the bi-encoder models by constructing the corpus and name it Explagraphs-Open. We consider a single path (*subject-relation-object*) as a retrieval unit and construct the corpus by dumping all the possible paths provided from the dataset.

A.6 RuleTaker-Open

RuleTaker dataset is a synthetic rule-based dataset used to measure the model ability on reasoning over rules (Clark et al., 2021; Tafjord et al., 2021; Saha et al., 2020). Given a small corpus of textual facts and rules, the model has to answer the question, retrieve, and construct the graph-structured proofs. As in Tafjord et al. (2021), we use the maximum depth dataset *D5* for training. To evaluate the model performance in the open-setting, *i.e., Task3 in Dalvi et al. (2021)*, we newly construct a large corpus and divide the train/dev/test dataset by the unique query set from the original D5 dataset.

Dataset Construction We dump all the facts and rules from the original D5 train/dev/test datasets to construct the corpus and collect 1621 unique queries, which we split into 1300/121/200. We remove cases with NAF and FAIL cases for rule-based evaluation, remove graphs with less than two nodes to ensure that the fact from the corpus itself could not be the proof, and remove graphs with more than ten nodes to fit in the maximum length of T5 model. Also, we added *DONE* at the end of graph construction for dynamic stopping as in Section 3.1.

Evaluation Metric In RuleTaker-Open, there are various possible answer graphs for a query, unlike the previous RuleTaker dataset. Therefore, to check whether the prediction graph is correct, a new evaluation metric is necessary. Since each textual sentence can be divided into a simple format, *subject-relation-object*, when considering the constructed method (Clark et al., 2021), we evaluate the result by a new rule-based method.
We check whether the constructed graph is well-constructed by four steps.

- *Node Num Error*: The number of evidence should be larger than 2.
- *Start Node Error*: First word (*subject*) should be the same.
- *End Node Error*: Last word (*object*) should be the same.
- *Missing Edge Error*: There should be no missing edge.

Table 10 shows the rate on each constraint for both the bi-encoder model and GRLS. Each error in the

Table 10: Error rate for each error type in RuleTaker-Open. Results are from 200 test sets.

Error Rate (%)	GRLS	ST5
Node Num Error	0.5	5
Start Node Error	9.5	0
End Node Error	20	28
Missing Edge Error	19	50
Success	51	17

table corresponds to the item on top with the same name .

Missing Edge Error is evaluated by Algorithm 3; when given a prediction graph (P), we divide the sentences into rules and facts and check for the missing edge in the prediction order. When the algorithm returns *True*, the graph is considered to have no missing edge.

Algorithm 3 Finding the missing edge

Require: Input Corpus P

T := An empty list to append or remove facts from P

for all sentence $s \in P$ **do**
  if s is a rule **then**
    divide s to assumptions A and result r
    for all assumption $a \in A$ **do**
      if a in T **then**
        T.remove(a)
      else
        return False ▷ Missing edge
      end if
    end for
    T.append(r)
  else
    T.append(s)
  end if
end for

if T is empty **then**
  return True ▷ No missing edge
else
  return False ▷ Missing edge
end if

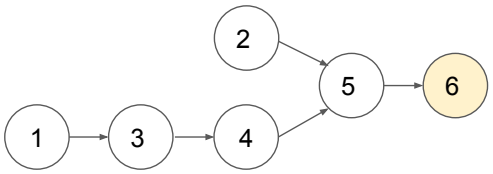
Predicted reasoning graph of GRLS and Bi-encoder retrieval (ST5) are in Appendix A.7

A.7 RuleTaker-Open Prediction Results

The prediction result from the model, predicted corpus (P), is in the gray box, and the final node is colored in yellow. The Missing nodes are colored in red, and the leftover nodes are colored in blue. If there is a red or blue node, it means that it failed to construct the reasoning graph. We show two examples for each retrieval method and success and failure cases (missing edge error case) in Figure 3, 4, 5, and 6.

Predicted Corpus (P):

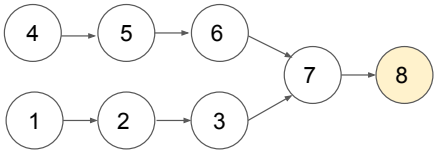
1. The cat is kind
2. The cat is kind
3. If something is kind then it chases the cat
4. If something chases the cat then it is young.
5. If something is kind and young then it is cold.
6. If something is cold then it visits the dog.



(a) Example1

Predicted Corpus (P):

1. The lion is young.
2. If something is young then it eats the lion.
3. If something eats the lion then it is kind.
4. The lion is young.
5. If something is young then it eats the lion.
6. If something eats the lion then it likes the lion.
7. If something is kind and it likes the lion then the lion eats the cow.
8. If something eats the cow then it eats the rabbit.



(b) Example2

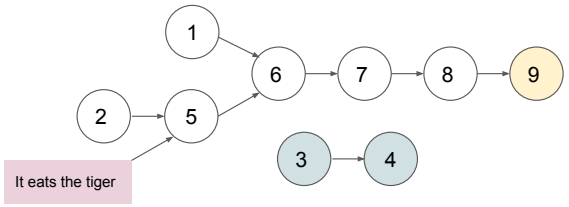
Figure 3: Success Examples of GRLS

A.8 Details of Bi-Encoder Retrieval Models (ST5)

We use ST5 model (Ni et al., 2021) as the architecture of the bi-encoder baseline to compare the performance with GRLS using the same number of parameters. The input text is fed into T5-encoder, and the first decoder output of the T5-decoder is taken as the sentence embedding. We follow the implementation details in Ni et al. (2021) except for two settings: (1) When calculating the similarity, instead of using cosine similarity, we use the inner product as in Karpukhin et al. since cosine similarity shows a low recall rate during multi-step

Predicted Corpus (P):

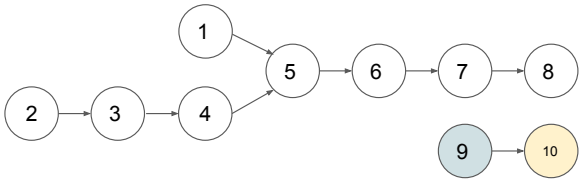
1. The mouse is cold.
2. The mouse is cold.
3. The mouse is cold.
4. If something is cold then it eats the dog.
5. If something is cold and it eats the tiger then the tiger is kind.
6. If something is cold and kind then it is red.
7. If something is red then it sees the mouse.
8. If something sees the mouse then the mouse is green.
9. If something is green then it sees the squirrel.



(a) Example1: Leftover node (blue) and missing nodes (red)

Predicted Corpus (P):

1. The bear is kind.
2. The bear is kind.
3. If something is kind then it chases the bear.
4. If something chases the bear then it is big.
5. If something is kind and big then it is rough.
6. If something is rough then it likes the bald eagle.
7. If something likes the bald eagle then it likes the tiger.
8. If something likes the tiger then it likes the lion.
9. If something likes the bear and it likes the bald eagle then it is round.
10. If something is round then it likes the bear.



(b) Example2: Leftover node (blue)

Figure 4: Failure Examples of GRLS

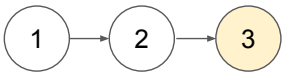
(iterative) training. (2) We change the hyperparameters for a fair comparison with GRLS.

A.9 EntailBank with Different Unseen Rates

To check how the performance changes when unseen rate changes inside the same dataset, we create a subset of the EntailBank dataset with different unseen rates by using the subset of the training dataset (reducing the number of graphs in the training dataset to 1,313). To check whether the results are general, we create two different datasets with high and low unseen rates, an average of 32.4% and 8.6%, respectively. Specific numbers of unseen rates and the performance of recall rate (R@5) are in Table 11. We observe that GRLS consistently outperforms the bi-encoder model for the tasks with a low unseen rate, whereas the bi-encoder retrieval performs better for the tasks with a high unseen rate.

Predicted Corpus (P):

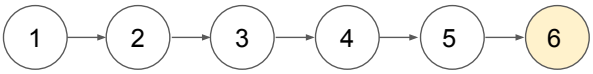
1. The rabbit is blue.
2. If something is blue then it sees the rabbit.
3. If something sees the rabbit then it is big.



(a) Example1

Predicted Corpus (P):

1. The rabbit is big.
2. If someone is big then they need the rabbit.
3. If someone needs the rabbit then they are big.
4. If someone is big then they like the rabbit.
5. If they like the rabbit then the rabbit is kind.
6. If someone is kind then they visit the rabbit.



(b) Example2

Figure 5: Success Examples of Bi-encoder (ST5) Retrieval

A.10 Retrieval Sequence Recall Rate on Seen and Unseen Subsets

We analyze the recall rates of the best GRLS on unseen and seen subsets. Table 12 shows the performance on seen subset is consistently higher than that on the unseen subset.

A.11 Ratio of Single-Step-Solvable Questions

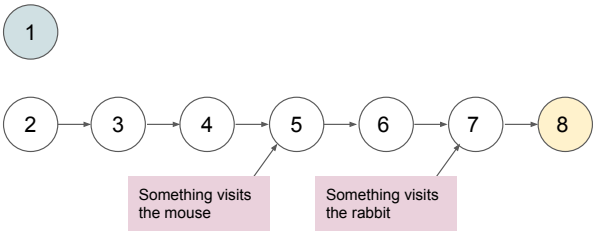
We conduct a manual analysis on the ratio of questions that are answerable by single-step retrieval in multi-hop datasets, which are questions that can be answered considering only the question without other retrieval sequences. In other words, they are

Table 11: The table shows the Evidence Recall score (R@5) of GRLS and BE on EntailBank subset with different unseen rate. **Unseen** column shows the rate of queries with unseen retrieval sequences over total number of queries. GRLS shows high performance for tasks with high unseen rate (High 1 and High 2) and low performance for tasks with low unseen ratio (Low 1 and Low 2)

Task	Unseen Rate	GRLS	BE
High 1	33.8%	23.2	30.6
High 2	30.9%	25.8	35.4
Low 1	8.2%	33.0	42.5
Low 2	9.0%	31.2	26.1

Predicted Corpus (P):

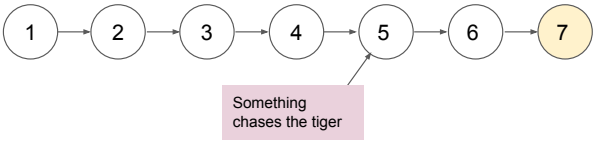
1. The mouse eats the rabbit.
2. The rabbit eats the mouse.
3. If something eats the mouse then it visits the rabbit.
4. If something visits the rabbit then it eats the rabbit.
5. If something visits the mouse and it eats the rabbit then the rabbit is cold.
6. If something is cold then it eats the rabbit.
7. If something visits the rabbit and it eats the rabbit then the rabbit is kind.
8. If something is kind then it eats the mouse.



(a) Example1: Leftover node (blue) and missing nodes (red)

Predicted Corpus (P):

1. The mouse chases the dog.
2. If the mouse chases the dog then the mouse is red.
3. If the mouse is red then the mouse visits the tiger.
4. If something visits the tiger then the tiger chases the mouse.
5. If something chases the tiger and the tiger chases the mouse then it sees the mouse.
6. If something sees the mouse then it sees the dog.
7. If something sees the dog then it chases the mouse.



(b) Example2: missing node (red)

Figure 6: Failure Examples of Bi-encoder (ST5) Retrieval

Table 12: Retrieval sequence recall rate (R@k) of GRLS for each dataset divided by *seen* queries and *unseen* queries. We use the model with the highest score for each dataset. K is 2 for HotpotQA and 5 for the other datasets.

Dataset	Seen	Unseen
EntailBank	54.9	34.4
StrategyQA	59.3	43.8
Explagraphs-Open	42.2	32.5
HotpotQA	70.0	24.7
NQ	83.6	13.7

similar to comparison type questions rather than bridge type questions in HotpotQA.

The analysis is done on randomly sampled 30 questions from test datasets of EntailBank, StrategyQA, and Explagraphs-Open datasets. Table 13 shows that StrategyQA and Explagraph-Open have a high rate of single-step-solvable questions,

Table 13: Rate of single-step-solvable and multi-step-solvable questions from randomly sampled 30 questions on EntailBank, StrategyQA, Explagraphs-Open.

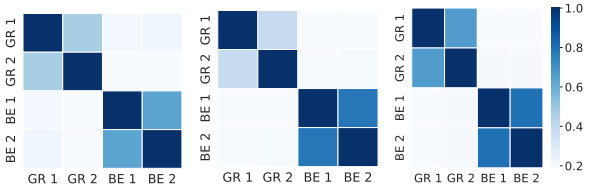
	Single-Step-Solvable	Multi-Step-Solvable
EntailBank	40.0	60.0
StrategyQA	70.0	30.0
Explagraphs-Open	76.7	23.3

and EntailBank shows a low rate. In Table 7, we show examples of single-step and multi-step input and output on EntailBank, Explagraphs-Open, and StrategyQA.

A.12 Manual Analysis on HotpotQA

We conduct manual analysis on HotpotQA by comparing the top-2 prediction result of the GRLS and MDR, a bi-encoder retriever. From the two question categories in HotpotQA (bridge and comparison questions), we manually inspect 30 sampled examples where one model is fit while the other model is wrong in Appendix A.12. MDR mostly got wrong by missing the second hop item though it got the first hop correct and GRLS was wrong for cases where the first-hop item is not written explicitly in the query but by sharing a specific part of a sentence. When the item is written explicitly in the query, GRLS tend to get it correct, which shares with the result that GRLS shows a higher score on comparison questions than MDR. We suggest this result is because GRLS can directly cross-encode between the input and the output without any information loss.

To be specific, we divide the error case into four:
(1) When the first-hop retrieval item is not written explicitly in the query but by sharing a specific part of a sentence.
(2) Though it is written explicitly in the query, it retrieves the wrong document by giving attention to an irrelevant part of the query.
(3) Detail of the title is wrong (i.e., when the gold document has title *Do you Love Me (Not That I Can Dance)*, the model retrieves a document with the title *Do you Love Me (2NE1 song)* instead; when *do you love me* is in a query, the model misses to correctly understand the details.)
(4) The retriever got the first hop correct but failed to retrieve the second hop item correctly.
When comparing the number of models matched in the bridge question with each error case, among the four cases, MDR is often wrong in the second (1.3 times) and fourth cases (2.2 times), and the GRLS



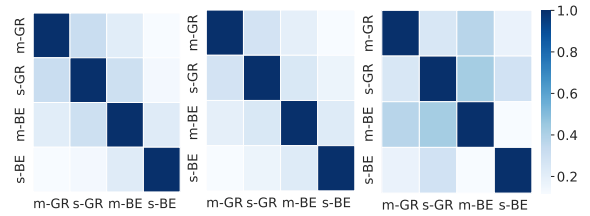
(a) EntailBank (b) StrategyQA (c) Explagraphs-Open

Figure 7: Pairwise prediction similarity ratio over two different models: multi-step generative retrieval (GRLS) and single-step bi-encoder retrieval (BE) over three datasets. The dark color indicates there is a higher similarity between the predictions of the two models and is calculated by retrieval sequence recall rate. The same model with different numbers is trained with the same model but different seeds. We can see that the same models with different seeds have darker colors (higher similarity) compared to the ones with different models. GRLS 1 and BE 1 are models trained with seed 101 and GRLS 2 and BE 2 are models trained with seed 42.

is most often wrong in the first case (6 times) along with the third case (2.8 times)[11].

A.13 Pairwise Prediction Similarity Between BE and GRLS

Figure 7 shows the pairwise predictions similarity ratio between two models among two single-step BE and two multi-step GRLS trained with different random seeds. The result demonstrates that BE and GRLS tend to retrieve different retrieval sequences.



(a) EntailBank (b) StrategyQA (c) Explagraphs-Open

Figure 8: Pairwise prediction similarity ratio over four different models: multi-step GRLS (m-GR), single-step GRLS (s-GR), multi-step BE (m-BE), and single-step BE (s-BE). Dark color indicates high similarity between the two model predictions. The similarity is calculated based on retrieval sequence recall rate.

A.14 Pairwise Prediction Similarity Between Four Models

In this section, we compare the similarity between the four models: multi-step GRLS, single-step GRLS, multi-step BE, and single-step BE. Figure 8 shows how similar the four models are on three datasets: EntailBank, StrategyQA, and Explagraphs-Open. Darker color indicates a high similarity between the prediction results of the paired two models.

For all three datasets, multi-step GRLS (m-GR) and single-step BE (s-BE) show low similarity, which indicates that the two models capture different aspects for the same query, which further shows improvement by a simple ensemble method. Also, Explagraphs-Open shows high similarity over the four models compared to other datasets.

[11] the value in parentheses shows the ratio of the error rate compared to the other model