Momentum Posterior Regularization for Multi-hop Dense Retrieval

Zehua Xia[*1], **Yuyang Wu**[*1,2], **Yiyun Xia**[1,3], **Cam-Tu Nguyen**[†1,3]

[1]State Key Laboratory for Novel Software Technology, Nanjing University
[2]School of Computer Science, Nanjing University
[3]School of Artificial Intelligence, Nanjing University

Emails: {zehuaxia,wuyuyang,xiayiyun}@smail.nju.edu.cn; ncamtu@nju.edu.cn

Abstract

Multi-hop question answering (QA) often requires sequential retrieval (multi-hop retrieval), where each hop retrieves missing knowledge based on information from previous hops. To facilitate more effective retrieval, we aim to distill knowledge from a posterior retrieval, which has access to posterior information like an answer, into a prior retrieval used during inference when such information is unavailable. Unfortunately, current methods for knowledge distillation in one-time retrieval are ineffective for multi-hop QA due to two issues: 1) Posterior information is often defined as the response (i.e. the answer), which may not clearly connect to the query without intermediate retrieval; and 2) The large knowledge gap between prior and posterior retrievals makes existing distillation methods unstable, even resulting in performance loss. As such, we propose MoPo (Momentum Posterior Regularization) with two key innovations: 1) Posterior information of one hop is defined as a query-focus summary from the golden knowledge of the previous and current hops; 2) We develop an effective training strategy where the posterior retrieval is updated along with the prior retrieval via momentum moving average method, allowing smoother and effective distillation. Experiments on HotpotQA and StrategyQA demonstrate that MoPo outperforms existing baselines in both retrieval and downstream QA tasks[‡].

1 Introduction

LLMs demonstrate strong language capabilities, but their knowledge is not frequently updated, which makes it ineffective in responding to time-sensitive questions. In addition, LLMs still suffer from hallucinations, especially in knowledge-intensive question-answering tasks. Although augmenting LLMs with retrieval is a promising solution, a single round of retrieval is often insufficient for complex, multi-hop queries (Yang et al., 2018). Recent approaches have focused on multi-hop reasoning with LLM (Trivedi et al., 2023; Zhang et al., 2024), multi-hop reranking (Zhang et al., 2024), or multi-hop dense retrieval (Xiong et al., 2021). The first two approaches, when being used without a retrieval, are less efficient than multi-hop dense retrieval. This is because they require costly attention computation between the query and documents during inference. In contrast, multi-hop dense retrieval separately encodes queries and documents, enabling the precomputation of document embeddings for offline indexing and efficient search.

Inspired by the success of distilling posterior information (e.g. answers) to one-time retrieval (Chen et al., 2020; Feng et al., 2020), we target an unexplored research question of *"how can we effectively distill posterior information to facilitate multi-hop dense retrieval in complex, multi-hop QA?"* Our intuition is that such posterior information and the query can play as "anchor information," thus reducing the "semantic shift" issue (Xiong et al., 2021) and facilitating multi-hop retrieval. Unfortunately, two main challenges hinder the adoption of existing methods in multi-hop QA. Firstly, most previous methods treat responses as the posterior information to train the posterior retrieval. However, in multi-hop QA, the response (answer) is not directly related to the original query. As a result, the posterior retrieval trained on such information may misguide the prior retrieval, consequently missing the important information at the intermediate hops. Secondly, the knowledge gap between prior and posterior retrievals makes it difficult for knowledge distillation. Although this issue exists in one-time retrieval, the gap is generally larger in multi-hop retrieval. Our empirical results

[*]These authors contributed equally to this work and should be regarded as co-first authors.
[†]Corresponding author.
[‡]Our code is available at https://github.com/zeaver/mopo

Proceedings of the 31st International Conference on Computational Linguistics, pages 8255–8271
January 19–24, 2025.

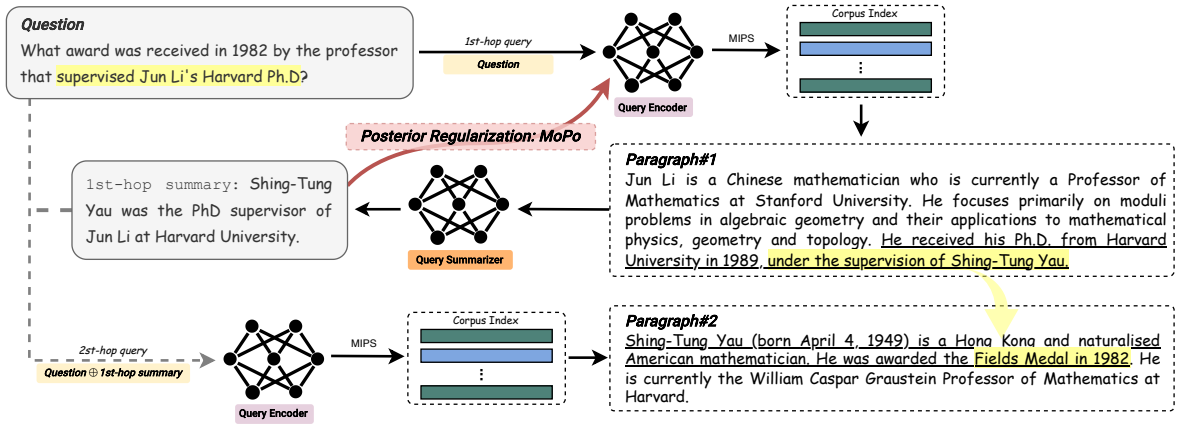


Figure 1: An example from HotpotQA benchmark. Given the 2-hop question, an iterative retriever is expected to retrieve the 1st and 2nd golden paragraph sequentially. After every retrieval step, a query-focused summary combining the query and the retrieved paragraph is generated, which is a kind of posterior information before conducting its retrieval. We utilize it to enhance the retriever training as shown by the red arrow.

show that, even with a suitable posterior information, the existing methods (Chen et al., 2020) do not work well for multi-hop retrieval, even resulting in a drop in retrieval performance.

To address the aforementioned issues, this study incorporates posterior information to multi-hop dense retrieval with two main ideas. Firstly, rather than defining the final answer as the posterior information, we employ query-focused summary of the gold knowledge in the previous and the current hops as the posterior information for the current hop (see Figure 1). By doing so, we ensure a stronger correlation between the posterior information and the current hop context. To facilitate training, we develop PostSumQA, a dataset derived from HotpotQA with 22,696 questions. In PostSumQA, we provide posterior summary annotation at every hop for all multi-hop questions. Secondly, we propose *Momentum Posterior Regularization* (MoPo), which exhibits smoother convergence and is easier to train compared to existing posterior regularization (Chen et al., 2020). In MoPo, the posterior model is updated at each training step using a momentum-based moving average of the prior model, thus reducing the knowledge gap between the prior and posterior retrieval models. Experimental results on HotpotQA and StrategyQA show that MoPo effectively exploits posterior information, resulting in better retrieval performance compared to previous multi-hop retrieval methods and existing posterior regularization methods. When being used in the traditional retrieval-reranking-generation, MoPo helps improve the downstream tasks, the reranking and QA tasks. Particularly, a simple pipeline with MoPo as the retrieval outperforms contemporary methods based on multi-hop reranking and multi-hop reasoning with LLM.

Our contributions can be summarized as follows:

- To our knowledge, we are the first to introduce posterior query-focused summary for multi-hop dense retrieval. Toward this end, we present PostSumQA, a high-quality dataset comprising 22,696 entries in English, tailored for training models on posterior information.
- This study proposes Momentum Posterior Regularization (MoPo), a simple yet effective posterior regularization framework for multi-hop dense retrieval.
- Our method is empirically tested on the HotpotQA and StrategyQA datasets, outperforming recent baselines both in retrieval and downstream tasks (reranking and QA).

2 Related Works

Multi-hop Dense Retrieval Recent studies extend the dense retrieval framework (Devlin et al., 2019) to support multi-hop QA. MDR (Qi et al., 2021) proposes an iterative framework, where the retrieval of each hop depends on the previous retrieved documents in the previous steps. Here, MDR exploits a dual-encoder (Karpukhin et al., 2020) as the single-time retrieval for each hop retrieval. BeamDR (Zhao et al., 2021) is similar to MDR in that it aims to retrieve the next document

depending on the document candidates in the beam. Its optimization objective, however, is based on contrasting positive retrieval chain to the negative retrieval chain, rather than contrasting hop-level samples as in MDR. Our research builds on these studies, yet we aim to incorporate posterior information into the retrieval process.

Multi-hop Reranking It is common to exploit an inefficient but effective cross-encoder to rerank the retrieved candidates (Xiong et al., 2021) for better knowledge selection. The cross-encoder model requires the expensive cross-attention between a query and a document at the inference time; thus, is only efficiently used with a small set of candidates, typically from an efficient retrieval. Recently, Ma et al. (2023) proposes Chain-of-Skills (CoS) that adopts a similar framework with MDR but performs reranking after every hop. Unlike CoS, which interleaves between retrieval and reranking, BeamRetrieval (Zhao et al., 2021) performs direct multi-hop reranking from a set of document candidates, that is obtained in advance.

Multi-hop Reasoning with LLM Thanks to the strong reasoning capability of LLM (Wei et al., 2022), recent studies treat LLM as a sophisticated retrieval-generation agent for multi-hop QA. Representative works being SelfAsk (Press et al., 2023), IRCoT (Trivedi et al., 2023), FLARE (Jiang et al., 2023), and BeamAggR (Chu et al., 2024). In general, these methods combine LLM and retrieval in two ways: 1) Performing question decomposition with LLM and using a retrieval to help generate K answers for each simple question in the question tree (Chu et al., 2024); 2) Interleaving between query reformalization and retrieval (Press et al., 2023; Trivedi et al., 2023; Jiang et al., 2023). The latter bears some resemblance to our framework, yet we focus on improving the retrieval with posterior information. It is noteworthy that these methods are much more costly than MoPo due to the use of resource-intensive LLM for reranking, query formalization, and answer generation.

Posterior Knowledge Enhancement Posterior knowledge has been used to refine knowledge selection in dialogue systems using (Kim et al., 2020; Chen et al., 2020). In general, these methods aim to perform knowledge distillation between the posterior and the prior retrievals by minimizing the KL divergence. Unlike these methods, however, we focus on mult-hop dense retrieval setting.

3 Methodology

3.1 Problem Definition

For open-domain multi-hop question answering retrieval: given the question q and a large textual corpus $\mathcal{D}$, a retriever need to retrieve a sequence $\mathbf{D}_{seq} = \{d_1, \ldots, d_L\}$ of L relevant documents to construct the reasoning chain and finally find the target answer a. In practice, the retriever returns the K documents with the highest scores as candidates for downstream modules, like reranker or reader/generation, where $|\mathcal{D}| \gg K$.

3.2 Iterative Multi-hop Dense Retriever

Inspired by MDR (Xiong et al., 2021), we model the probability of a sequence of documents given the query based on the dense retrieval model $\mathbf{M}_\theta$ with the parameters θ:

$$P_\theta(\mathbf{D}_{seq}|q) = \prod_{t=1}^{L} P_\theta(d_t|q, d_1, ..., d_{t-1}) \quad (1)$$

where, d_t represents the retrieved document at step t, and when $t = 1$, query is the original question. After finishing $t-$th retrieval, we apply some post-processing to q_t for next time. We define G_s as the query post-processing module:

$$q_t = G_s(q_{t-1}, d_{t-1}) \quad (2)$$

Accordingly, the Equation (1) is simplified as:

$$P_\theta(\mathbf{D}_{seq}|q) = \prod_{t=1}^{L} P_\theta(d_t|q_t) \quad (3)$$

Then the InfoNCE contrastive loss (van den Oord et al., 2019) function for a tuple within a batch $(q, \mathbf{D}_{seq}) \sim \mathbf{B}$ is as follows

$$\mathcal{L}_{\text{InfoNCE}}(\theta, \mathbf{B}) \quad (4)$$
$$= \mathbb{E}_{r\sim\mathbf{B}}\Big[\sum_{t=1}^{L} -\log \frac{\exp\left(f_\theta(q_t, d_t^+)\right)}{\sum_{d\in d_t^\pm} \exp\left(f_\theta\left(q_t, d\right)\right)}\Big]$$

with $r = (q, \mathbf{D}_{seq})$ where q is the original question, q_t, d_t^+ and d_t^- represent the query at the t-hop and the corresponding positive ($d_t^+ \in \mathbf{D}_{seq}$) and negative documents at t-th hop, respectively. Here, $f_\theta(\cdot)$ indicates the similarity function, which exploits $\mathbf{M}_\theta$ to map the query and the document into two dense vectors for similarity measurement like Exact Inner Product in MoPo.

3.3 Posterior Summary Utilization

We introduce posterior summary to reformulate query and enhance the retrieval capability of $\mathbf{M}_\theta$.

Query Reformulation In MDR, the post-processing module G_s performs a simple concatenation of the query and retrieved documents. However, such a method will cause an increment in the query length. On the other hand, if G_s is just a simple summarization, the semantic drift issue is highly likely to occur. As a result, we retain the original question q, and concatenate the summary s_{t-1} generated from retrieved documents as the query for the $t-$th step retrieval:

$$q_t = q \oplus s_{t-1} = q \oplus G_s(s_{t-2}, d_{t-1}, q) \quad (5)$$

The symbol $\oplus$ represents concatenation operation. Intuitively, the syntactic and semantic consistency across retrieval steps can be ensured in this way. For syntactic consistency, the query length will not increase significantly thanks to the summarization operation. In addition, except for the first-hop retrieval step, subsequent queries all have the same structure — the original question and the summary of all previously retrieved documents. For semantic consistency, the original question is included at every step, subsequently mitigating the semantic drift issue. Equation (3) is then rewritten as:

$$P(\mathbf{D}_{seq}|q) = \prod_{t=1}^{L} P(d_t|s_{t-1}, d_{t-1}, q) \quad (6)$$

Posterior Summary Enhanced Retriever It is intuitive that a retrieval that exploits both the question and the answer (the posterior information) is more effective than a retrieval that only uses a question for the search (Chen et al., 2020). In this paper, we define the posterior information for each hop (step) as the query-aware summary up to that step. Specifically, for the retrieval step t, as the search query q_t is formed by the original query q and the previous step summary s_{t-1}, the summary s_t is the posterior information for this particular step. Here, we denote the posterior-enhanced retriever with parameters ϕ by $\mathbf{M}_\phi$, which has the same architecture and similarity function with $\mathbf{M}_\theta$ for simplicity. Given a tuple $(s_t, d_t^+, \mathbf{d}_t^-)$, similar in Equation (4), the posterior similarity during training is:

$$p_\phi(d_t|s_t) = \frac{\exp\left(f_\phi(s_t, d_t^+)\right)}{\sum_{d \in d_t^\pm} \exp\left(f_\phi\left(s_t, d\right)\right)} \quad (7)$$

To train $\mathbf{M}_\theta$, we sample a tuple $(q, \mathbf{S}_{seq}, \mathbf{D}_{seq}) \sim \mathbf{B}^{'}$, where $\mathbf{S}_{seq}$ indicates the query-focused summary sequence. We then calculate the posterior regularized InfoNCE loss as follows:

$$\mathcal{L}(\theta, \mathbf{B}^{'}) = \mathcal{L}_{\text{InfoNCE}}(\theta, \mathbf{B}^{'}) + \lambda \cdot \mathbb{E}_{r' \sim \mathbf{B}'}\big[\sum_{t=1}^{L} D_{\text{KL}}(p_\phi(d_t|s_t) \| p_\theta(d_t|q_t))\big] \quad (8)$$

with $r^{'} = (q, \mathbf{S}_{seq}, \mathbf{D}_{seq})$, and λ is the Kullback–Leibler (KL) divergence loss weight.

3.4 Momentum Posterior Regularization

Usually, we adopt a two-stage training strategy for Posterior Regularization (PR): firstly train a posterior model $\mathbf{M}_\phi$ on the whole training tuples $\{(\mathbf{S}_{seq}, \mathbf{D}_{seq}, \mathbf{d}^-)\}$, and then employ $\mathbf{M}_\phi$ to compute the KL divergence with the retriever $\mathbf{M}_\theta$ training on $\{(q, \mathbf{S}_{seq}, \mathbf{D}_{seq})\}$. However, this solution yields poor results on retrieval evaluation.

Analysis of PR Loss We analyze the training loss to understand the failure cause of 2-stage PR strategy. In the first stage, during the training of the posterior model $\mathbf{M}_\phi$, we observe that the training loss converges rapidly. In contrast, in the second stage, which involves training the prior model $\mathbf{M}_\theta$, both the InfoNCE and Kullback-Leibler (KL) losses exhibit slow convergence rates. This is particularly pronounced for the KL term. Moreover, we note that the KL term consistently accounts for a high proportion of the total loss, even when we reduce its weight λ. This behavior is unexpected and potentially problematic, as the KL term is intended to serve as a regularization component rather than dominate the training process.

Momentum Update We hypothesize that this suboptimality may be attributed to the overly influential supervised training signal from the posterior summary. For this reason, it is desirable that the posterior regularization is smooth and not overly strong. Therefore, we propose a momentum posterior regularization framework to address this issue. At training step τ, given the momentum coefficient m, we update ϕ by:

$$\phi^{(\tau)} \leftarrow m\phi^{(\tau-1)} + (1-m)\theta^{(\tau-1)} \quad (9)$$

The posterior distribution is obtained only by passing the query with posterior information (s_t) through M_ϕ in forward-passing. In essence, only the parameters of prior model θ are updated by

Algorithm 1: Momentum Posterior Regularization

Input: Momentum coefficient m; Prior model $\mathbf{M}_\theta$
Data: Training dataset $\mathbf{X}$

$\mathbf{M}_\phi \leftarrow \mathbf{M}_\theta$; // Initialize posterior model $\mathbf{M}_\phi$
for $(q, \mathbf{D}_{seq}, \mathbf{S}_{seq})$ ***in*** $\mathbf{X}$ **do**
  $P_\theta(\mathbf{D}_{seq}|q) \leftarrow M_\theta(q, \mathbf{D}_{seq})$; // Compute prior logit in Equation (4)
  $\phi \leftarrow m\phi + (1-m)\theta$; // Momentum update $\mathbf{M}_\phi$ in Equation (9)
  $P_\phi(\mathbf{D}_{seq}|\mathbf{S}_{seq}) \leftarrow M_\phi(q, \mathbf{S}_{seq}, \mathbf{D}_{seq})$; // Compute posterior logit in Equation (7)
  $\mathcal{L}(\theta) \leftarrow \mathcal{L}_{\text{InfoNCE}}\left[P_\theta(\mathbf{D}_{seq}|q)\right] + \lambda \cdot \text{KL}\left[P_\phi(\mathbf{D}_{seq}|\mathbf{S}_{seq}) \| P_\theta(\mathbf{D}_{seq}|q)\right]$; // Equation (8)
  Update M_θ with $\mathcal{L}(\theta)$ by Adam optimizer (Kingma and Ba, 2015);
end

back-propagation. By doing so, it simplifies the original two-stage training for posterior regularization to one-stage training strategy, making the training simpler compared to PR.

4 Data Preparation with Backward Summary Generation

MoPo training requires a dataset with posterior information annotation, i.e. query-focused summaries. In this paper, we aim at an automatic method to construct such a dataset from a multi-hop QA dataset. More specifically, given a sequence of $(q, d_1, d_2, ..., d_L, a)$, where q, a are the question, answer and $(d_1, d_2, ..., d_L)$ is the document sequence containing key information to derive the answer, the objective is to generate $(q, \ldots, d_t, s_t, \ldots, a)$, where s_t is the summary at the t-th hop. The direct intuition is to exploit LLM to help generate such summaries in the forward direction. In other words, we can aim to generate s_t given q, d_t, s_{t-1}. However, doing so can lead to semantic drift as LLM may include redundant information from d_t, for it does not know which key information is needed for the final answer.

In this paper, we propose a *backward summary generation* for data generation. Specifically, for the last hop (the L-th hop), we use the rule-based method QA2D (Demszky et al., 2018) to generate s_L from q and a, making sure that there is no redundant information in d_L included in s_L. For the intermediate t-th step, given s_{t+1} summary in the next hop, the query q, and the current hop document d_t, we ask LLM to generate s_t from q, d_t, s_{t+1} by removing redundant information in s_{t+1} that is not included in d_t. Our experience shows that having access to the "look ahead" information in s_{t+1} facilitates the hop summary generation. Experiments in Appendix A show the advantage of backward summary generation in generating high-quality data compared to the forward summary process. By applying the above process to a subset of the HotpotQA training set (Yang et al., 2018), a multi-hop QA set in English, we obtain 22,696 data points for MoPo training. We refer to this dataset as PostSumQA and publish it for future research. Detailed information and analysis of this dataset are provided in the Appendix.

5 Retrieval Experiments

5.1 Experimental Setup

Datasets We test the retrieval and comprehensive reasoning ability of MoPo on two datasets: HotpotQA (Yang et al., 2018) and StrategyQA (Geva et al., 2021). HotpotQA is a multi-hop question answering dataset in the open domain, necessitating information from two separate Wikipedia pages to respond to a query. It includes 113K multi-hop questions and ~5M documents. Both dev/test sets of HotpotQA have ~7K samples. StrategyQA is another open-domain multi-hop question answering dataset with 2,780 examples and ~36M documents, where the reasoning process is not explicitly stated in the question, requiring 2 hops of information retrieval and strategic thinking to derive the answer. On StrategyQA, we directly utilize the model fine-tuned on HotpotQA for evaluation. As a result, we consider StrategyQA is a held-out dataset that allows us to the generalization of compared methods.

Metrics Like (Xiong et al., 2021), we use Recall and Exact Match (EM) metrics to evaluate the performance. Retrieval EM measures the percentage of test queries of which *at least one of the retrieved sequences exactly matches that of the golden document sequence*. On the other hand, Recall measures the percentage of test queries of which at least one

Model	R@2	R@20	R@50	R@100	EM@2	EM@20	EM@50	EM@100
				HotpotQA				
BeamDR	-	-	-	92.90	60.70	-	-	79.20
$\text{MDR}_{\text{origin}}$	65.90	80.20	-	-	-	-	-	-
MDR_{zero}	92.12	92.64	93.77	94.62	46.11	58.27	69.37	73.09
MDR	94.34	94.85	95.63	96.38	55.96	71.73	76.82	79.70
MDR_{sum}	94.66	95.27	95.85	96.38	60.04	72.65	76.98	79.94
PR_{fixed}	94.49	95.07	95.88	96.55	57.67	72.18	76.93	79.95
PR_{dyn}	94.52	95.13	95.69	96.61	57.33	72.67	77.13	80.17
MoPo	**94.77**	**95.43**	**96.27**	**96.70**	**63.03**	**76.74**	**80.27**	**82.20**
				StrategyQA				
MDR_{zero}	42.80	43.15	43.52	43.85	27.96	32.44	35.83	37.34
MDR	42.64	42.85	43.21	43.70	25.31	31.87	35.29	36.06
MDR_{sum}	42.85	43.31	43.66	43.87	28.88	32.92	36.46	37.40
PR_{fixed}	42.38	42.92	43.39	43.80	25.76	32.31	36.05	36.53
PR_{dyn}	42.80	43.14	43.47	43.78	25.81	32.84	36.14	37.14
MoPo	**43.36**	**43.61**	**43.94**	**44.15**	**31.91**	**35.61**	**37.90**	**39.24**

Table 1: Retrieval performance in recall and EM at k retrieved passages within 10×20 search space. Results of $\text{MDR}_{\text{origin}}$ and BeamDR come form their own paper. We also evaluate retrieval performance in different base models in Table C3 and Table C4.

of the retrieved sequences containing *at least one document of the golden document sequence*. More detailed information can be found in Appendix B.2. As multi-hop QA requires information from all the hops to get the answer, EM is a more important metric for multi-hop retrieval. Additionally, the beam size, or the number of candidate documents to achieve at each hop, is an essential parameter for final retrieval performance. We follow the golden passages order of MDR (Xiong et al., 2021) but reduce the beam size from 50×50 to 10×20. EM@K and Recall@K are EM and Recall metrics measured by retrieving top K/L sequences from 10×20 candidates, where sequence scores are measured by Equation 1 and $L = 2$ is the maximum number of hops in HotpotQA and StrategyQA.

Implementation Details All the experiments are conducted on 4×32G V100 GPUs. We initialize MoPo with a powerful pre-trained text embedder E5-v2-base (Wang et al., 2024) as retrieval model. In addition, flan-t5-large (Chung et al., 2022) is used as summary-generation model. After training, we exploit the retrieval model M_θ to obtain document embeddings and index them with the Exact Inner Product (IndexFlatIP) in FAISS (Johnson et al., 2021) for efficient search. More implementation details are shown in Appendix B.

5.2 Baselines

There are two groups of baselines for retrieval:

Models without posterior information This group of baselines contains BeamDR (Zhao et al., 2021), MDR_{origin} (Xiong et al., 2021), and several variants of MDR. Here, MDR is the variant of the original one where we exploit E5-v2-base as the hop dense retrieval instead of RoBERTa (Liu et al., 2019) in the original MDR. MDR_{sum} and MDR_{zero} are the variants of MDR with two modifications: 1) The hop retrieval is E5-v2-base; 2) We exploit the summary model as in MoPo for query formalization instead of concatenating queries with retrieved documents as in MDR. Different from MDR and MDR_{sum}, the retrieval model of MDR_{zero} are not finetuned on PostSumQA. The inclusion of MDR_{zero} is for measuring the zero-shot performance when applying E5-v2-base as the hop retrieval in the MDR framework without further training. Note that once MoPo is trained, it is used for inference in the same way with MDR_{sum}. In other words, there is no additional inference cost associated with MoPo in comparison with MDR_{sum}.

Models with Posterior Information The baselines in this group includes PR_{fixed} and PR_{dyn},

which are MDR with two-stage training for posterior regularization (Chen et al., 2020). PR$_{fixed}$ (Chen et al., 2020) is trained with a fixed λ=0.3, determined through grid search over the range $[0.1, 1.0]$ and a step size of 0.1. For PR$_{dyn}$, we design a linear decay scheduler, analogous to learning rate scheduling, where λ decreases from 0.3 to 0.1 over the course of training.

5.3 Retrieval Results

Table 1 shows that MoPo outperforms all other baselines on both datasets. From the results, several findings can be obtained: (1) The fact that MDR is better than MDR$_{origin}$ shows that the adoption of the powerful (single-time) dense retrieval model, E5-v2-base, is essential for multi-hop retrieval. Notably, even MDR$_{zero}$ attains decent performance without training; (2) MDR$_{sum}$ is significantly better than MDR, demonstrating that our query re-formalization is effective; (3) Although the dynamic adjustment of the proportion of PR term does enhance retrieval performance, both PR methods exhibit inferior performance compared to MDR$_{sum}$. PR$_{dyn}$ is better than PR$_{fixed}$ but worse than MoPo, showing that dynamic scheduling can help but not that much. (4) MoPo introduces a more effective training strategy for posterior knowledge distillation, resulting in superior performance compared to other baselines, particularly in EM metrics, which are critical for multi-hop QA systems.

Further evaluation of the held-out dataset, StrategyQA, provides some insightful observations: (1) The superior performance of MDR$_{zero}$ over MDR, which is trained on HotpotQA, indicates that MDR exhibits limited generalization to StrategyQA. The possible reason is that MDR concatenates the query and the retrieved document for next-hop retrieval, making it prone to over-fitting to dataset-specific characteristics such as document length or irrelevant information. In contrast, MDR$_{sum}$ and MoPo, which use a summary model to rewrite the query, may avoid this issue. (2) Despite being trained exclusively on PostSumQA, a derived set of HotpotQA training data, the summary model still facilitates retrieval performance, as being seen by the superior performance of MDR$_{sum}$ compared to MDR$_{zero}$; (3) MoPo, trained on PostSumQA, also demonstrates its robustness when being tested on StrategyQA, significantly outperforming all other baselines in EM metrics.

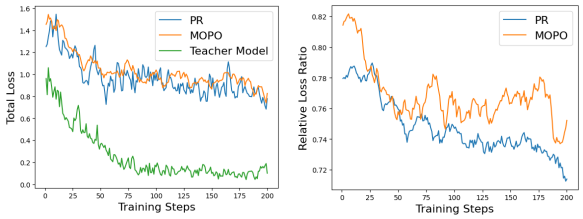


Figure 2: Total absolute and relative loss curve of PR and MoPo, λ=0.3, where relative Loss Ratio = InfoNCE Loss / Total Loss. All curves are processed with the same smoothing factor.

5.4 Analysis of Training Loss Curve

Our experimental results show that PR$_{fixed}$ struggles to improve over MDR$_{sum}$ even with hyper-parameter tuning. To further explore the reason, we draw the total and relative loss curves of PR$_{fixed}$ with λ=0.3, the grid-searched value. Here, the relative loss measures the ratio of InfoNCE loss over the total loss (InfoNCE plus PR term). The lower the loss ratio is, the bigger the role of PR term is in optimization.

From the total loss plot in Figure 2, we see that the posterior model (the teacher model) converges much faster compared to the student models. Specifically, the teacher model converges around ~80 steps while the student models (PR, MoPo) do not. From the relative loss plots, it is observable that the posterior regularization (PR) term always occupies a big ratio in the total loss. As training progresses, the proportion of PR terms increases faster than the decrease rate of total loss. By dynamically adjusting the weights of PR terms, we can mitigate this issue, resulting in a better results for PR$_{dyn}$ compared to PR$_{fixed}$ (see Table 1), yet the performance is still not satisfactory. Figure 2 also shows that MoPo total loss is smoother compared to PR$_{fixed}$, and the higher relative loss curve indicates that MoPo can limit the domination of PR term during optimization.

5.5 Analysis on Hyper-parameter Sensitivity

Less Sensitive to λ We study how the the performance of PR_{fixed} and MoPo change with different values of the posterior regularization weight λ. The results in Table 2 show that MoPo has much lower performance gap compared to PR_{fixed}, verifying the robustness of MoPo in multi-hop dense retrieval.

Momentum Update is More Effective We measure the impact of the parameter m in Equation

	R@2	R@20	R@50	R@100
		PR		
$\lambda = 0.3$	94.49	95.07	95.88	96.55
$\lambda = 0.5$	93.85	94.39	95.13	95.75
$\lambda = 1.0$	93.93	94.59	95.29	95.85
PG(‰) ↓	-6.773	-7.153	-7.822	-8.286
		MoPo		
$\lambda = 0.3$	94.76	95.42	96.17	96.85
$\lambda = 0.5$	94.97	95.69	96.48	96.98
$\lambda = 1.0$	94.77	95.43	96.27	96.70
PG(‰) ↓	**-2.211**	**-2.821**	**-3.213**	**-2.887**

Table 2: Retrieval performance on HotpotQA with different λ values. PG means *Performance Gap*, the ratio that deviates the most from the best performance among all performances

Models	R@2	R@20	R@50	R@100
PR_{fixed}	94.49	95.07	95.88	96.55
$\text{PR}_{dynamic}$	94.52	95.13	95.69	96.61
$m = 0$	94.53	95.27	95.92	96.62
$m = 0.5$	94.61	95.36	96.24	96.68
$m = 0.9$	94.69	95.33	96.25	96.67
$m = 0.99$	**94.77**	**95.43**	**96.27**	**96.70**
$m = 1.0$	94.73	95.33	96.21	96.67

Table 3: Retrieval performance of MoPo on HotpotQA with different momentum coefficient m.

9 on MoPo. Utilizing a grid search on a smaller validation set, we identified several representative values for testing as shown in Table 3. Note that, m=0 means the posterior model is the same as prior during every training step, and m=1 means that the posterior model is kept to be the initial encoder during training. As depicted in Table 3, all models are better than $\text{PR}_{dynamic}$ on Recall@100. Besides, m=0.99 and 0 performs the best and worst, respectively. The best value of m falls into the range between 0.9 and 1, indicating that MoPo prefers slower updating.

6 Downstream Tasks

6.1 Reranking

We utilize the pre-trained jina-reranker-v2-multilingual (Günther et al., 2023) as a reranker to test MoPo retrieval performance. After the retrieval stage, for each retrieved sequence $\{(d_1, d_2, \ldots, d_L)\}$, we use the reranker model to calculate the relevance score of each doc and calculate the sequence score as follows:

$$P(d_1, \ldots, d_L, q) \propto \prod_t P(d_t, q)$$
$$P(d_t, q) = \text{Reranker}([q, d_t]) \quad (10)$$

Although this simple reranking assumes an independence assumption among documents in the sequence, its documents are retrieved in the sequencial order with DPR. In this experiment, we compare our results with two other competitive retrieval models that incorporate reranking techniques: Beam Retrieval (Zhang et al., 2024), Chain-of-Skills (CoS) (Ma et al., 2023). Both Beam Retrieval and CoS exploit MDR as the retrieval framework and perform reranking at every hop. In contrast, we perform only ranking at the last hop. In addition, CoS incorporates multiple additional tasks, such as entity span prediction and linking, making the model much more complicated than MoPo+reranking. To be consistent with previous work, we set the beam size here at 50. In other words, for two hops in HotpotQA, the search space is 50×50. Experimental results in Table 4 demonstrate the advantages of MoPo over the baselines regardless of being more efficient compared to the stronger baselines (Beam Retrieval and CoS). It is noteworthy that, in comparison with CoS, MoPo achieves better performance with less computing requirement. Specifically, we need 4 V100 32G for training MoPo while CoS requires 16 V100 32G for training.

Methods	EM@2
MDR (reranking) (Xiong et al., 2021)	81.2
Beam Retrieval (Zhang et al., 2024)	82.2
Chain-of-Skills (Ma et al., 2023)	88.9
MoPo (reranking)	**89.4**

Table 4: Fullwiki HotpotQA reranked retrieval results. The beam size here is 50×50. Results of baseline methods come from their own paper.

6.2 Question Answering

We evaluate MoPo when being used in a traditional retrieval-reranking-generation for QA task. Here, we utilize Flan-T5-large(Chung et al., 2022) as the generation for answer and supporting sentence generation.

Table 5 shows the results of MoPo and other baselines on the whole test set of HotpotQA with

Methods	Dev						Test					
	Ans		Sup		Joint		Ans		Sup		Joint	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
MDR (Xiong et al., 2021)	62.3	75.1	56.5	79.4	42.1	66.3	62.3	75.3	57.5	80.9	41.8	66.6
MDR(new)	64.2	76.7	60.1	82.7	42.3	67.5	-	-	-	-	-	-
IRRR+ (Qi et al., 2021)	-	-	-	-	-	-	66.3	79.9	57.2	82.6	43.1	69.8
HopRetriever-plus (Li et al., 2021)	66.6	79.2	56.0	81.8	42.0	69.0	64.8	77.8	56.1	81.8	41.0	67.8
TPRR (Zhang et al., 2021)	67.3	80.1	60.2	84.5	45.3	71.4	67.0	79.5	59.4	84.3	44.4	70.8
AISO (Zhu et al., 2021)	68.1	80.9	61.5	86.5	45.9	72.6	67.5	80.5	61.2	86.0	44.9	72.0
Chain-of-Skills (Ma et al., 2023)	68.2	81.0	61.1	85.3	46.4	72.3	67.4	80.1	61.3	85.3	**45.7**	71.7
MoPo	**68.4**	**81.5**	**61.9**	**86.9**	**46.7**	**72.7**	**67.6**	**80.8**	**61.4**	**86.1**	**45.7**	**72.1**

Table 5: Downstream QA results on HotpotQA fullwiki set. MDR(new) means MDR reproduced with the same base models as MoPo. Results of other models come from their own paper.

Model	Overall	Bridge	Comp.
Self-Ask (Press et al., 2023)	49.4	45.3	68.6
IRCoT (Trivedi et al., 2023)	56.2	53.4	69.6
FLARE (Jiang et al., 2023)	56.1	54.2	54.4
BeamAggR (Chu et al., 2024)	62.9	60.5	74.2
MoPo	**64.3**	**61.2**	**76.2**

Table 6: Downstream QA performance in F1 on HotpotQA subset of 100 samples (Trivedi et al., 2023). Results of other models come from (Chu et al., 2024).

fullwiki setting, where MoPo establishes the state-of-the-art (SOTA) benchmark on both HotpotQA dev and test datasets. Furthermore, we observe a 1.2% enhancement in performance when the base models of MDR are updated in MDR (new). Nevertheless, MoPo surpasses the revised MDR by a margin of 5.2%, which demonstrates the effectiveness of our methodology.

In addition, we compare MoPo-based pipeline with some other latest methods that exploit LLM for reasoning. We test on the same test set provided by IRCoT, which contains only 100 instances. Table6 shows the results, where MoPo outperforms all baselines on HotpotQA set while maintaining a more efficient inference time.

7 Conclusion

This paper introduces Momentum Posterior Regularization (MoPo), the first attempt to distill posterior information into multi-hop dense retrieval. In MoPo, we define posterior information for each hop as a query-focused summary, and introduce a smooth and effective training strategy based on momentum-based moving average method. To facilitate MoPo training, we automatically construct PostSumQA from HotpotQA using a novel method, namely *backward summary generation*. Our experimental results show the effectiveness of our method in exploiting posterior information, resulting in an improvement in retrieval performance. When being used in a traditional pipeline of retrieval-reranker-generation (Re2G), MoPo-based Re2G exhibits superior performance compared to strong baselines for two downstream tasks, reranking and QA.

Limitations

The present study is subject to two principal limitations. First, although our training method is proven to be useful, the theory behind training controllability needs further exploration. We plan to further explore this in our future work. Second, a comprehensive evaluation should include detailed analysis on the inference time of MoPo-based pipeline and other baselines in Reranking and QA tasks. However, the diversity of the baselines and the cost of such models prevent us to perform such analysis at the present. We aim to address these limitations in future research endeavors.

Ethics Statement

MoPo aims to improve the performance of multi-hop dense retrieval and the training processing of posterior regularization. We exclusively utilized existing datasets from previously published works. No new data collection was conducted for this study. All experiments and query-focused summary constructions were performed strictly within the confines of these pre-existing datasets. The nature of our generation process ensures that even in cases of inaccuracy, the outputs remain controllable and pose no potential harm. This is due to the constrained scope of the input data and the controlled

nature of our experimental environment. The current model operates solely in English, which inherently limits its practical applications in diverse, multilingual real-world scenarios.

References

Xiuyi Chen, Fandong Meng, Peng Li, Feilong Chen, Shuang Xu, Bo Xu, and Jie Zhou. 2020. Bridging the gap between prior and posterior knowledge selection for knowledge-grounded dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3426–3437, Online. Association for Computational Linguistics.

Zheng Chu, Jingchang Chen, Qianglong Chen, Haotian Wang, Kun Zhu, Xiyuan Du, Weijiang Yu, Ming Liu, and Bing Qin. 2024. BeamAggR: Beam aggregation reasoning over multi-source knowledge for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1229–1248, Bangkok, Thailand. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *Preprint*, arXiv:1809.02922.

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.

Shaoxiong Feng, Hongshen Chen, Kan Li, and Dawei Yin. 2020. Posterior-gan: Towards informative and coherent response generation with posterior generative adversarial network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:7708–7715.

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. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *Preprint*, arXiv:2310.19923.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Preprint*, arXiv:2112.09118.

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

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020. Sequential Latent Knowledge Selection for Knowledge-Grounded Dialogue. In *ICLR*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2021. Hopretriever: Retrieve hops over wikipedia to answer complex questions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 13279–13287.

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. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Kaixin Ma, Hao Cheng, Yu Zhang, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2023. Chain-of-skills: A configurable model for open-domain question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1599–1618, Toronto, Canada. Association for Computational Linguistics.

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.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Peng Qi, Haejun Lee, Tg Sido, and Christopher Manning. 2021. Answering open-domain questions of varying reasoning steps from text. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3599–3614, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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)*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zehua Xia, Qi Gou, Bowen Yu, Haiyang Yu, Fei Huang, Yongbin Li, and Nguyen Cam-Tu. 2023. Improving question generation with multi-level content planning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 800–814, Singapore. Association for Computational Linguistics.

Wenhan Xiong, Xiang Lorraine Li, Srinivasan Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. *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.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Liu Yong, and Shen Huang. 2024. 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.

Xinyu Zhang, Ke Zhan, Enrui Hu, Chengzhen Fu, Lan Luo, Hao Jiang, Yantao Jia, Fan Yu, Zhicheng Dou, Zhao Cao, and Lei Chen. 2021. Answer complex questions: Path ranker is all you need. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 449–458, New York, NY, USA. Association for Computing Machinery.

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III. 2021. Multi-step reasoning over unstructured text with beam dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4635–4641, Online. Association for Computational Linguistics.

Yunchang Zhu, Liang Pang, Yanyan Lan, Huawei Shen, and Xueqi Cheng. 2021. Adaptive information seeking for open-domain question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3615–3626,

Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Methods and Analyses of Data Synthesis

A.1 Detailed pipeline of data synthesis

Rule-based method: QA2D The second-hop summaries of all bridge questions are generated by the rule-based method QA2D(Demszky et al., 2018). The original intention of this method is to create a large-scale semi-supervised NLI (Natural Language Inference) dataset. Compared with the single NLI dataset, there are more publicly available manually annotated QA datasets. QA2D can convert question-answer pairs into corresponding declarative forms. Although the original QA2D paper only conducted experiments on SQuAD(Rajpurkar et al., 2016), Xia (Xia et al., 2023)also verified the effectiveness of this method on the HotpotQA dataset. The inherent disadvantage of rule-based methods is poor generalization. The answers to comparison-type question-answer pairs in the HotpotQA dataset are mainly "yes" and "no", while SQuAD is mainly factoid question answering, and the answers are extractive. QA2D developed on SQuAD cannot cover all examples in HotpotQA, especially comparison-type question answering pairs. Therefore, this paper only uses this method to generate the second-hop summary of the bridging problem. If we encounter a sample that QA2D cannot calculate, we will directly discard it.

LLM-based method We use GPT-3.5-Turbo to complete the first-hop and second-hop summary generation of all comparison type samples, and the first-hop summary generation task of all bridge type samples. We write the prompt words for summary generation according to the following points, taking the prompt words for the first-hop summary generation of bridge type samples as an example:

1. Complex task decomposition: The query-centric summary generation task with multiple thinking steps is decomposed into multiple subtasks, namely reasoning step generation and query-centric summary generation.

2. The requirements should be clear. Use Markdown syntax to write (1) a detailed explanation of each item entered; (2) clearly state each step required to complete the task;

3. Two examples are provided: the examples are input in order: `Context`, `Statement`, `Reasoning steps` and `Output`. The context only uses the title and crowd-sourced supporting fact sentences instead of the entire supporting paragraph to provide more effective information.

4. Write the steps of the Chain of Thought into the example. And the output should include the Reasoning Steps.

5. If an input example is too difficult for the LLM, skip that example.

We provide a detailed example in Appendix A.2.

Semi-supervised data generation method for comparing type samples Considering that comparison type problems are relatively difficult problems, we add a specific interpretation of comparison type problems to the prompt words for the first-hop summary generation of bridge type samples, and modify the execution steps and requirements of specific tasks. Considering that multi-task text generation, that is, it is difficult to generate the first-hop and second-hop summaries at the same time, and to avoid the mutual influence of the generation results, we perform two tasks separately. The input for the first-hop summary generation is the input question and a supporting document, and the input for the second-hop summary generation is the question, the answer, and two supporting documents. The reasoning step requirements in the prompt words of the two tasks are shown in Figure A3. In general, the specific interpretation of comparison type problems has been given, and the overall idea of the two generation tasks is to find common content first and then generate summaries.

A.2 Data synthesis example

LLM-based method example Figure A1 shows the prompt words for generating the first-hop summary of the bridge type. For the sake of convenience, the examples and input forms in the prompt words are ignored. The reasoning chain of the example is completed using GPT and manually corrected. Taking the content in Figure 1 as an example, the reasoning process for generating the first-hop summary is shown in Figure A2, and its reasoning steps are strictly carried out in accordance with the requirements in Figure A1. Intuitively, this can ensure the consistency of the gener-

```
"""You're a content writer, and
    your task is to re-write the
    ###Statement by removing the
    contents that ###Context does
    not mentions.

The input contains:
###Context: A title and its
    corresponding sentences.
###Statement: A statement but
    contains the information that
    is not appears in the , ###
    Context and maybe has some
    grammar errors.

You can finish this as follow
    steps:
1. Find the common contents that
    both ###Context and ###
    Statement contain.
2. Remove the noise information
    that is not appears in ###
    Context.
3. Fix the grammar error, and
    keep content consistency.
"""
```

Figure A1: Main content of the prompt words generated by the first hop summary of the bridge type sample

```
""" 1. Common contents: Both the
    context and statement mention
    Jun Li and his association
    with Harvard University.
2. Removing irrelevant
    information: The statement
    mentions the Fields Medal
    received in 1982, but this
    information is not present in
    the context. Therefore, it is
    removed from the output.
3. Grammar and content
    consistency: The output
    sentence is rephrased to
    maintain grammatical
    correctness and content
    consistency. It states that
    Shing-Tung Yau was the PhD
    supervisor of Jun Li at
    Harvard University, which is
    the relevant information
    mentioned in the context."""
```

Figure A2: Example of reasoning steps for first-hop summary generation in bridge-type question answering

ated content. The specific explanations of Figure A1 and Figure A2 are as follows:

- First, find the common content between the context and the complete answer, mainly "*Jun Li*" and "*Harvard University*";
- Second, find the content mentioned in the complete answer but not covered by the context, which is "*Fields Medal received in 1982*", and remove it from the complete answer;
- Finally, considering that the complete answer is based on the output of the rule model, there may be minor errors such as grammar, which can be corrected using LLM.

The method shown in Figure A1 does not directly use the context and the given question to directly generate the first-hop summary, but instead inputs the complete answer constructed based on the QA2D method, which is equivalent to removing redundant content in the complete answer based on the context content, that is, content not included in the context.

A.3 Quality analysis of LLM-filtering data

In Table A2, rule filtering primarily refers to GPT's string outputs unaccounted for by QA2D rules or unparseable in JSON format; LLM filtering denotes tasks that GPT deems unanswerable based on input instructions. The sample type distribution reveals that rules filtered approximately 3% of the data, while LLM filtered around 9%, predominantly comparison type data. This could stem from the increased complexity of generating comparison type summaries compared to bridge type, with LLM potentially struggling to comprehend or extract the compared entities.

Considering that the proportion of samples filtered out by LLM in Table A2 is relatively high, we further analyze the proportion of first-hop and second-hop summary generation failures in failed samples. We found that LLM filtered 1210 first-hop summaries and 59 second-hop summaries. The loss of 59 second-hop samples is normal data iteration loss, so we manually screened the reasons

```
First_hop = """
1. Identify what the ###Question
   asks, and find the relevant
   information in the ###Context.
2. Identify the common contents
   that the ###Context contains
   and the ###Question asks.
3. Ingore or move the information
    that ###Question asks but ###
   Context does not mention.
4. According to the common
   contents, write a question-
   oriented statement of the ###
   Context."""

Second_hop = """
1. First check the ###Answer is
   yes/no or some specific
   contents.
2. Identify the relevant content
   to the ###Question ask in ###
   Context1.
3. Identify the relevant content
   to the ###Question ask in ###
   Context2.
4. Combine the ###Answer and ###
   Question according to the
   relevant content"""
```

Figure A3: Main content of inference hint words for summary generation of comparative question-answer pairs

Key	Content
Question	Who was an American former professional baseball left fielder, Andrea Martin or Chuck Essegian?
Context	Andrea Martin. Andrea Louise Martin (born January 15, 1947) is an American actress, singer, author and comedian, best known for her work in the television series "SCTV".
Decline Reason	(1) Identify what the Question asks: The Question asks for a comparison of nationality between two individuals, Andrea Martin and Chuck Essegian. (2) Identify the common content between the Context and the Question: Andrea Martin. (3) Remove the information that the Question asks but the Context does not mention: Chuck Essegian. (4) No common content found between the Context and the Question.
2nd Hop summary	Charles Abraham 'Chuck' Essegian was an American former professional baseball left fielder, while Andrea Martin is an American actress, singer, author, and comedian.

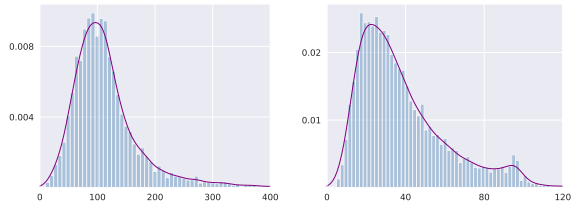
Table A1: Example of first-hop summary generation failure in LLM filter comparison type

Stage	Bridge	Comparison
Original	20165(80.66%)	4835(19.34%)
Rule-filtering	19398(80.11%)	4815(19.87%)
LLM-filtering	19120(84.24%)	3576(15.76%)

Table A2: Distribution of two types of questions in the semi-supervised data production process

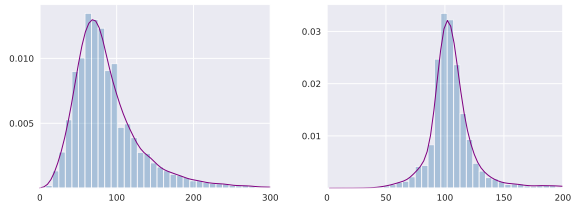
for the failure of the first-hop samples. Table A1 shows an example of a comparison type where the first-hop summary generation failed. GPT made two misjudgments. The first was that the question was about nationality, and the second was that the given context and the question had no common content. The possible reason is that the input question specifies the specific content of the occupation, while the input context is only related to nationality, which is too difficult for GPT. When generating the second summary, both target paragraphs are used as input, so it is easier to generate the second-hop summary.

In addition, we also explored the effect of directly using GPT-3.5-Turbo to generate first-hop summaries. The first step of the input instruction is to summarize the common content, and the second step is to remove the content not mentioned in the input context.The results in Table A3 show that this is worse than the indirect method used in this paper: stripping the full answer into a first-hop summary.Manual evaluation found that the first-hop summary output by this method still carries content that is not mentioned in the input context but included in the question. Figure A4 shows 9559 statistics of first-hop summary lengths. As can be seen from Figure A4b, the first-hop summary generated by GPT is shorter than the input long context in terms of text length, which is in line with the summary task scenario. However, as can be seen from A4a, the text length of the first-hop summary is almost the same as the input question. Considering that the input is a multi-hop question involving two related paragraphs, the statistical result is consistent with the result of manual evaluation. In FigureA5, the relative length distribution results of



(a) Ratio of first-hop summary to question length (b) Ratio of first-hop summary to supporting facts

Figure A4: Relative length distribution of first-hop summaries generated using only GPT



(a) Ratio of first-hop summary to question length (b) Ratio of the second hop summary to question length

Figure A5: Relative length distribution of first- and second-hop summaries and questions in the semi-supervised dataset

22696 semi-supervised data are shown. FigureA5a shows that the relative length of the first-hop summary produced by the final method is concentrated around 0.8 to 0.9. FigureA5b shows that the relative length of the second-hop summary is closer to a normal distribution with a smaller variance, further verifying the effectiveness of our semi-supervised dataset production method.

B Training Details

B.1 Implementation Details

Base Model We use E5-base(Wang et al., 2024), Jina-Reranker-v2(Günther et al., 2023) and Flan-T5-large(Chung et al., 2022) as the base models for retrieval tasks, reranking tasks and query-centric summary generation tasks respectively. The network structure of E5-base is the same as BERT, with 110M parameters. Its pre-training data does not include HotpotQA, and its performance has achieved excellent results on MTEB(Muennighoff et al., 2023) and BEIR(Thakur et al., 2021). The Jina-Reranker-v2 is a transformer-based model that has been fine-tuned for text reranking task, which has demonstrated competitiveness across a series of benchmarks targeting for text retrieval against other reranker models. Flan-T5-large is a generative model that is fine-tuned based on T5-large and has excellent performance in various natural language processing tasks. We use a simple instruction template and removes the samples and specific explanations of the tasks in the previous data augmentation instructions.

	Correctness	Coverage
PostSumQA	9.36	9.03
GPT-generated	9.23	8.52

Table A3: Human review between summaries in PostSumQA and GPT-generated data

Learning Rate	2e-5
Data batch size	128
Paragraph cutoff size	350
Query cutoff size	350
Learning rate warm-up rate	0.1
Gradient clipping	2.0
Training steps	200
Weight Decay	0.1

(a) Retrieval model training hyperparameters

Learning Rate	2e-5
Data batch size	32
Gradient accumulation times	4
Number of training rounds	1
Learning rate warm-up rate	0.1
Enter the cutoff length	400
Maximum output length	60
Whether to sample	No

(b) Query-centric summary generation model training hyperparameters

Table B1: Training hyperparameter settings

Parameter Setting In the retrieval task, the baseline model MDR, the retrieval model enhanced by summary enhancement m_θ, the teacher model m_ϕ, the basic posterior regularization model (hereinafter referred to as PR) and MoPo are all trained for 200 steps. The momentum factor m is set to 0.99. Other hyperparameters are shown in TableB1a. In the query-centric summary generation task, two sample examples are used, which is consistent with the instruction format when semi-supervised data is produced. All linear layers in the attention mechanism are enhanced using LoRA(Hu et al., 2022) and only trained once. The training hyperparameters are shown in TableB1b. All model optimizers are AdamW(Loshchilov and Hutter, 2019).

	Relevance	Correctness	Accuracy	Coverage
ChatGLM4 review				
QFS	**2.98**	**5.97**	**5.61**	**5.49**
DBS	2.56	4.66	3.92	4.26
Human review				
QFS	**2.40**	**8.40**	**4.30**	**4.91**
DBS	1.80	5.71	3.70	4.12

Table C1: Human and LLM review results of query-focused summary and decomposition-based summary. QFS means query-focused summary, DBS means decomposition-based summary

B.2 Details of Evaluation Metrics

EM measures whether the predicted answer exactly matches the ground truth answer. For a single query, if the predicted answer is identical to the ground truth (ignoring case and punctuation), the score is 1; otherwise, it is 0. The final EM score is the average of exact matches across all queries:

$$EM = \frac{\sum_{i=1}^{N} \mathbb{1}(\text{Predict}_i = \text{Answer}_i)}{N}$$

where N is the total number of queries.

Recall measures how many relevant answers from ground truth are retrieved by the model. For Top-K retrieval, Recall@K is calculated as:

$$R@K = \frac{\sum_{i=1}^{N} \mathbb{1}(\text{Answer}_i \in \text{Top-K Preds}_i)}{N}$$

where N is the total number of queries, and Top-K Predictions are the top K returned answers.

C Additional Experiments and Details

C.1 Ablation Study

Influence of posterior summary utilization. Results in Table 1 have already shown that Posterior Summary Utilization can greatly improve the performance of MDR, especially when reasoning is required. In this section, we further evaluate the difference between query-focused summary generation and decomposition-based summary generation originally used by MDR through reviewed by 3 human experts and ChatGLM4, a powerful multilingual LLM. Results are shown in Table C1. Our method achieves better performance. This result demonstrates the superiority of our method. Furthermore, the overall standard deviation of our method is 1.65, while that of decomposition-based summary generation is 2.86. This further proves the stability of our method.

	Bridge	**Comparison**	**Total**
MDR	89.26	96.26	90.44
PR	77.46	97.20	80.67
MoPo	**93.74**	**99.13**	**94.67**

Table C2: Retrieval performance in Recall@100 on HotpotQA using 1st hop golden summary

Model	R@2	R@20	R@50	R@100
MDR	94.38	94.85	95.76	96.49
MDR *w/* S	94.71	95.31	96.08	96.58
PR	94.49	95.22	95.91	96.51
MoPo	**94.71**	**95.34**	**96.12**	**96.81**

Table C3: Retrieval performance in recall at k retrieved passages with Contriever as embedding model on HotpotQA dev set

Influence of momentum posterior regularization. To evaluate the effectiveness of Momentum Posterior Regularization, we evaluate the 2nd hop retrieval performance using 1st hop golden summaries on HotpotQA dev set. The results are shown in Table C2. As the results show, MoPo shows a great improvement over MDR baseline model, while PR even performs worse than our baseline model. This experiment confirms Momentum Posterior Regularization's effectiveness.

C.2 Experiment with Different Base Model

We also evaluate the retrieval ability of different models with Contriever(Izacard et al., 2022) as base embedding model. Results are shown in Table C3. From the results. we can find that MoPo still has a better performance compared with other baseline models.

Moreover, we further evaluate the retrieval ability of different base embedding models using MDR framework in Table C4. The main difference between e5-base and e5-base-v2 in this experiment is that the pre-training data of e5-base-v2 contains HotpotQA while the other one does not. Compared with results in Table 1, the results show that multi-hop framework is more effective.

C.3 Loss Trend Comparison with MDR

Following the analysis in Section 5.4. We further compared the loss curve of MoPo, PR$_{fixed}$. As shown in Figure C1, teacher model converges much faster than other methods, which makes the training effect weakened.

Model	R@2	R@20	R@50	R@100
e5-base	49.32	68.52	73.24	76.41
e5-base-v2	55.26	75.92	80.55	83.62

Table C4: Retrieval performance in recall at k retrieved passages with different zero-shot embedding base models and using MDR framework on HotpotQA dev set

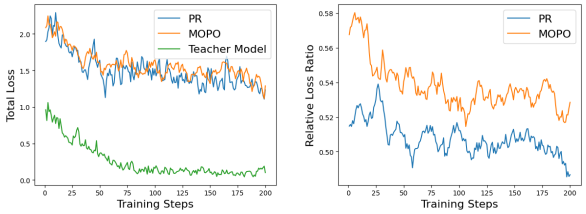


Figure C1: Total absolute and relative loss curve of PR and MoPo, λ=1.0, where relative Loss Ratio = InfoNCE Loss / Total Loss. All curves are processed with the same smoothing factor.

D Selected Baseline Details

SelfAsk (Press et al., 2023) exploits LLM to decide and generate next hop (follow-up) query, and uses a search engine for direct answer selection.

IRCoT (Trivedi et al., 2023) interleaves between CoT generation (Wei et al., 2022) and retrieving K documents.

FLARE (Jiang et al., 2023) iteratively uses LLM to predict the upcoming sentences to anticipate future content and a retrieval to obtain relevant documents for sentence generation if it contains low-confidence tokens.

BeamAggr (Chu et al., 2024) exploits LLM for question decomposition. For complex (multi-hop) questions, BeamAggr exploits a retrieval to get relevant documents for generating K (beamsize) answers for each question in the question tree.

MoPo+reranker+generation vs Reasoning with LLM LLM-based methods are much more costly compared to our method due to the use of LLM for reranking, query formalization and answer generation. For example, IRCoT (Trivedi et al., 2023) also requires L hops of iteration, each retrieves K documents and generate the next CoT (question reformalization) or the final answer. However, our method exploits lightweight generation models for query reformalization and answer generation, whereas IRCoT requires resource-intensive LLMs to reason over a set of K retrieved documents. As a result, given the same dense retrieval at each iteration, the cost of IRCoT outweighs MoPo significantly. To reduce computation cost, IRCoT (and other models) often exploit a light-weight BM25 retrieval. This case, however, the QA performance is limited by the weak retrieval capability of BM25.