Let Retrievers Think Before Action: Thought-Augmented Embedding for Dense Retrieval

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated that explicitly performing step-by-step thinking before producing final outputs can substantially improve performance on complex tasks, as exemplified by recent reasoning-oriented models such as OpenAI O1 and DeepSeek R1. Inspired by these advancements, we propose the O1 Embedder, a novel approach aiming to endow retrieval models with similar capabilities to address challenges like multi-task retrieval, zero-shot retrieval, and tasks requiring intensive reasoning of complex relationships. The O1 Embedder generates preliminary thoughts for input queries before document retrieval. To realize this objective, we address two fundamental challenges in integrating thinking mechanisms into dense retrieval. First, retrieval tasks lack explicit supervision for intermediate thinking processes, making it difficult to define thoughts that are truly useful for retrieval. We address this challenge with a data synthesis framework following an "Exploration-Refinement" process, ensuring alignment with retrieval utility. Second, effectively integrating thought generation with representation learning requires a unified modeling framework that can jointly support generation and embedding within a single model. O1 Embedder addresses this challenge by jointly optimizing thought generation and dense retrieval in an end-to-end manner, enhancing retrieval accuracy while reducing complexity through a single deployable model. Extensive evaluations across diverse datasets demonstrate significant performance improvements, highlighting the effectiveness and generalization capability of O1 Embedder.

1 Introduction

Information retrieval (IR) is fundamental to many important applications, such as search engines and question answering systems (Karpukhin et al., 2020; Kobayashi and Takeda, 2000). Recently, it draws even higher attention because of its critical role in augmenting large language models (LLMs) with external knowledge (Zhu et al., 2023; Liu et al., 2024; Zhao et al., 2024b), a paradigm known as retrieval-augmented generation (RAG) (Gao et al., 2023; Zhang et al., 2023; Zhao et al., 2024a). Over the past decade, IR techniques have experienced tremendous progresses. One important breakthrough is made by dense retrieval, where relevant data can be effectively retrieved via vector search. With the popularity of open-source models in this field (Neelakantan et al., 2022; Wang et al., 2023b; Xiao et al., 2024), dense retrieval has become a practical and widely adopted solution for real-world retrieval systems.

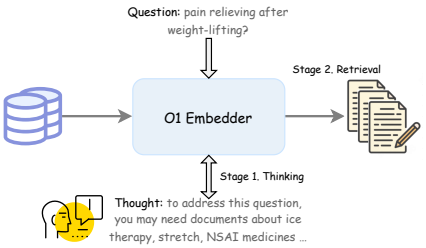

Figure 1: O1 Embedder. First of all, the model generates the thoughts about the question (thinking). Next, the model produces the embedding for dense retrieval (retrieval).

However, dense retrieval is still subject to many limitations in this stage. First, existing methods struggle with *zero-shot retrieval* tasks in unseen scenarios, which differ significantly from their source domains. For instance, a well-trained embedding model from general datasets is prone to a limited performance when applied to a specialized problem, like medical or legal case retrieval (Maia et al., 2018; Li et al., 2024d; Voorhees et al., 2021). Second, the existing models are insufficient to discriminate *complex relationships*, as they cannot be identified directly from semantic meaning. For ex-

ample, the retrieval of useful code snippets for a computer program, or the retrieval of evidence to a multi-hop problem (Li et al., 2024e; Husain et al., 2019; Yang et al., 2018; Lee et al., 2022).

The above challenges can benefit from a prior deep thinking process, instead of making direct judgment of relevance. Recently, remarkable advancements were made in this direction with the introduction of reasoning LLMs, such as OpenAI O1/O3, and DeepSeek R1 (Guo et al., 2025). Particularly, when a complex task is presented, the LLM is prompted to generate long-form thoughts about the problem in the first place. Through this process, the LLM can progressively get close to the correct solution, thus enabling the production of high-quality answers in the end. This operation is conceptualized as the test-time-scaling by recent studies (Snell et al., 2024), which has driven major improvements in solving complex problems, such as coding and mathematical proofs.

With the above inspiration, we propose **O1 Embedder**, which is designed to introduce a slow-thinking capability for embedding models, akin to that of LLMs. Our approach integrates two essential functions within a single model: **Thinking** and **Embedding**. First, it generates useful thoughts towards the input query, which explicitly uncovers the hidden information needs about the query. Secondly, it produces a discriminative embedding for the query and the generated thoughts. By incorporating both elements, the resulting embedding enables precise retrieval of relevant documents that are challenging to identify using the query alone.

The training of O1 Embedder is technically challenging given the absence of appropriate long-form thought data for embedding models. To solve this problem, we introduce a **Data Synthesis** method following an "**Exploration-Refinement**" process. First, we prompt an LLM to explore initial thoughts for a query. Next, we employ a retrieval committee to refine the initial thoughts. Each committee member scores the relevance between initial thoughts and the target document, which indicates their usefulness in retrieving the target document. With the collection of all members' scoring results, the golden thought is selected by majority voting and added to the training set. *As such, we automatically create long-form thoughts of the best retrieval utility for O1 Embedder.*

Building on well-curated long-form thought data, we introduce a **Multi-Task Training Method**, which fine-tunes a pre-trained LLM into O1 Embedder. Our method introduces two parallel training tasks. One applies supervised fine-tuning for the LLM, which enables the generation of optimal thoughts for an input query. The other one employs contrastive learning, which produces discriminative embeddings to retrieve relevant documents for a thought-augmented query. *With proper optimizations in loss computation and training workflow, these two tasks are seamlessly unified in a cost-efficient manner, leading to effective development of thinking and embedding capabilities throughout the training.*

The effectiveness of O1 Embedder is comprehensive evaluated. In our experiment, O1 Embedder achieves **a substantial improvement over existing methods** across a broad range of retrieval tasks, especially those requiring complex reasoning. O1 Embedder also demonstrates **a strong generalization ability** when applied to out-of-domain scenarios.

To summarize, the main contributions of this paper are highlighted by the following perspectives.

- We propose O1 Embedder, a unified embedding framework that generates useful thoughts for input queries before producing discriminative embeddings for dense retrieval. To the best of our knowledge, this is the first attempt to equip embedding models with explicit thinking capabilities.

- We introduce an exploration-refinement data synthesis framework that constructs long-form thoughts optimized for retrieval utility, along with a multi-task training strategy that effectively unifies thought generation and representation learning.

- We perform comprehensive evaluations demonstrating the effectiveness and broad applicability of O1 Embedder across diverse retrieval scenarios.

2 Method

In this section, we first present the problem formulation of O1 Embedder. We then introduce the two main technical contributions of this work: the production of long-form thought data for O1 Embedder, and the multi-task training process of O1 Embedder.

2.1 Problem Formulation

Dense retrieval measures the relevance between query and document based on their embedding similarity. Given an query q and document d, an embedding model ($\mathcal{M}$) is used to encode them into latent representations $\boldsymbol{v}_q$ and $\boldsymbol{v}_d$: $\boldsymbol{v}_q \leftarrow \mathcal{M}(q)$, $\boldsymbol{v}_d \leftarrow \mathcal{M}(d)$. To retrieve the relevant document d^* from a massive dataset D, the following nearest neighbor condition needs to be satisfied:

$$d^* = \text{ARGMAX.}\ \{\langle \boldsymbol{v}_q, \boldsymbol{v}_d \rangle | d \in D\}, \quad (1)$$

where $\langle \cdot \rangle$ indicates the inner-product operator. As discussed, traditional embedding models are insufficient to handle the challenges regarding zero-shot or complex retrieval problems.

Our method tackles the above challenge by introducing the thinking operation to embedding models. That is to say, the embedding model $\mathcal{M}$ is equipped with two functionalities: thinking $\mathcal{M}$.think($\cdot$) and embedding $\mathcal{M}$.embed($\cdot$). For an input query q, the embedding model generates its thoughts (t) on how to address the information needs of the query in the first place:

$$t_i \leftarrow \mathcal{M}.\text{think}(q),\ i = 1, ..., k \quad (2)$$

By revealing critical semantic matching patterns with relevant documents, the generated thoughts are expected facilitate the retrieval process for complex queries. In this place, a total of k thoughts are independently generated with respect to the query, which enables useful patterns to be comprehensively covered.

On top of the embedding model $\mathcal{M}$ and an aggregation function AGG, the query and its thoughts are jointly transformed into a unified embedding, called the thought-augmented embedding ($\hat{\boldsymbol{v}}_q$):

$$\hat{\boldsymbol{v}}_q \leftarrow \text{AGG.}(q, \{t_i\}_k; \mathcal{M}.\text{embed}) \quad (3)$$

As a result, the relevance between query and document is computed with the thought-augmented embedding: $\langle \hat{\boldsymbol{v}}_q, \boldsymbol{v}_d \rangle$. Finally, our problem is formulated as the joint training of thinking and embedding capability of model $\mathcal{M}$, such that the end-to-end retrieval performance is optimized.

2.2 Data Production

The training of O1 Embedder involves two types of data. One is used for the embedding capability, which is made up of queries and their relevant documents, i.e., q-doc tuples. The other one is used for the thinking capability, which includes queries and their thoughts, i.e., q-thought tuples. Unlike q-doc tuples which have been widely existed, there are no available q-thought tuples in reality. To resolve this problem, we propose a data synthesis pipeline, leveraging LLMs' readily equipped thinking capacity to generate such datasets. Our method follows an "**exploration-refinement**" workflow, as demonstrated in Figure 2.

First, we employ a LLM to explore candidate thoughts for a given query q. To facilitate proper generations from the LLM, the system prompt is formulated with the following template, where both instruction and examples are incorporated:

$$\text{PROMPT} = \text{TASK} : \{\text{Ins}\};\quad \text{EXAMPLES} : \{\text{E}\};\quad \text{QUERY} : \{\text{q}\} \quad (4)$$

The instruction is used to explicitly declare the demand for the thinking task; for example, "*think about a plausible response to address the query*". While the examples are introduced to demonstrate the form of desirable output. In this place, we randomly select m samples from the training set of q-doc dataset:

$$\text{E} = \{\text{QUERY} : q_i,\ \text{RESPONSE} : d_i\}_m. \quad (5)$$

Note that although the relevant document d for query q may seem like a trivial solution, it is unsuitable to serve as a thought. This is because the generated thought will be further used in embedding task, whose goal is to discriminate the relevant document d based on the thought-augmented embedding. If d (or any rephrased version of it) is used, the training process would be circumvented, ultimately leading to the collapse of the embedding task.

The generated thoughts may not always enhance retrieval performance due to potential hallucinations by the LLM. To ensure the inclusion of useful thoughts, the exploration process is repeated multiple times, generating several independent thoughts for the given input query. To identify the most useful thoughts, a quality-control mechanism is introduced to filter the generated candidates. Specifically, we employ a diverse set of retrievers, denoted as R. For each retriever $r \in R$, a similarity score is computed between a relevant document d and a thought t_i: $\sigma^r(t_i, d)$. The thought with the highest similarity score is selected by each retriever, i.e., $t_r^* \leftarrow \text{ARGMAX}(\{\sigma^r(t_i, d)\}_{i=1...k})$. Fi-

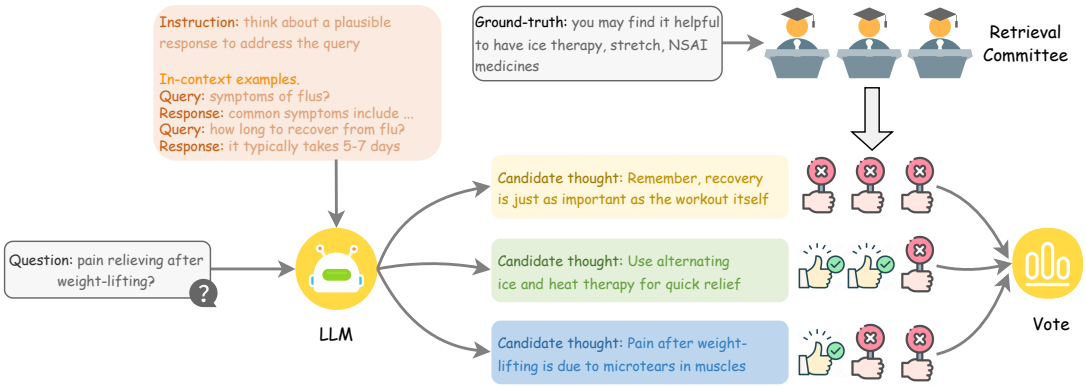

Figure 2: The production of thought data. In the first step, the LLM is prompted to generate candidates thoughts about the input question based on the instruction and in-context examples. In the second step, the retrieval committee is employed to evaluate the candidates by making comparison with the ground-truth document, i.e. the retrieval target. Finally, the candidate thought receiving the maximum votes is selected and incorporated to the training data.

nally, a majority voting process is conducted to determine the most useful thought. The thought that receives the highest number of nominations from the retrievers is selected as the final result: $t \leftarrow \text{VOTING}\{t_r^*\}_{r \in R}$.

By applying the above data synthesis workflow to an existing q-doc dataset: $D = \{(q_i, d_i)\}_N$, we can obtain an thought-augmented dataset composed of **q-thought-doc triplets**: $\hat{D} = \{(q_i, t_i, d_i)\}_N$, which offers long-form thoughts of the optimal retrieval utility.

2.3 Multi-Task Training

The O1 embedder is built upon a pre-trained LLM, leveraging the model's inherent generation and thinking abilities. Additionally, LLMs also show strong potential for fine-tuning as discriminative embedding models (Luo et al., 2024; Zhu et al., 2023). We apply the following multitask training for a pre-trained LLM backbone, which establishes its thinking capability via supervised behavior cloning and embedding ability through contrastive learning.

2.3.1 Behavior cloning

The foundation model is trained to generate thoughts for the input query through supervised fine-tuning. Given the dataset of q-thought-doc triplets: $\hat{D} = \{(q_i, t_i, d_i)\}_N$, a training sample x_i is formulated with the template below:

$$x_i = \text{<query>}q_i\text{<thought>}t_i\text{</s>}, \tag{6}$$

where <query>, <thought>, </s> are the special tokens to landmark the query, thought, and the completion of generation, respectively.

With the formulation of above training samples, the model is fine-tuned w.r.t. the following **generation loss**:

$$\mathcal{L}_{gen} = -\sum_{x_i} \sum_{j=|q_i|}^{|q_i|+|t_i|} \log P(x_{i,j}|x_{i,<j}), \tag{7}$$

where the next-token-prediction loss is minimized for each of the tokens starting from the beginning of the thought (i.e. $j \geq |q_i|$).

2.3.2 Contrastive learning

The pre-trained LLM is also fine-tuned to distinguish relevant documents from a query based on its generated embeddings. Traditionally, LLM-based embedders utilize the </s> token for text embedding. However, the </s> token has been designated to indicate the completion of generation process in our approach. As a result, incorporating an extra embedding task could lead to the collapse of training process. To avoid mutual interference between the two parallel training tasks, we employ another **special token <emb>** and append it to the end of input x (following </s>) to compute the text embedding:

$$\boldsymbol{v}_x \leftarrow \text{LLM}(x; \text{<emb>})[-1] \tag{8}$$

This simple modification substantially increases the compatibility, which is crucial to maintain the successful running of joint training process. Considering that people may want to leverage thought-augmented embedding to handle complex retrieval tasks while still relying on basic query embedding to process simple retrieval tasks, we generate the

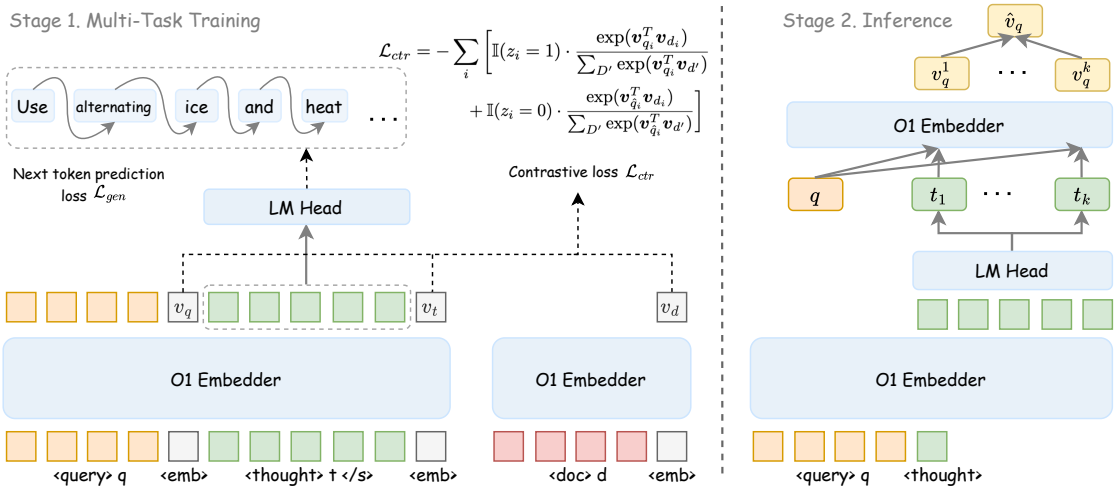

Figure 3: Multi-Task Training and Inference process of O1 Embedder. During the training process, O1 embedder minimizes two losses: the generation loss while decoding the thought, and the contrastive loss while discriminating the target document. During the inference process, multiple thoughts are generated for the query. The thoughts are used to produce thought-augmented queries, which are independently encoded by O1 Embedder and aggregated for retrieval.

two forms of query embeddings simultaneously to identify the relevant document d_i. As a result, we perform the following composite contrastive learning, where two **contrastive losses** are calculated based on the query and the thought-augmented query of each training sample, respectively:

$$\mathcal{L}_{ctr} = -\sum_i \Bigg[\mathbb{I}(z_i = 1) \cdot \frac{\exp(\boldsymbol{v}_{q_i}^T \boldsymbol{v}_{d_i})}{\sum_{D'} \exp(\boldsymbol{v}_{q_i}^T \boldsymbol{v}_{d'})} + \mathbb{I}(z_i = 0) \cdot \frac{\exp(\boldsymbol{v}_{\hat{q}_i}^T \boldsymbol{v}_{d_i})}{\sum_{D'} \exp(\boldsymbol{v}_{\hat{q}_i}^T \boldsymbol{v}_{d'})}\Bigg] \tag{9}$$

In this place, $\hat{q}_i$ indicates the thought-augmented query: $\hat{q}_i \leftarrow q_i + t_i$, D' stands for the collection of negative samples, including both in-batch negative samples and hard negative samples introduced by a pre-trained embedder. $z_i \sim \text{Bernoulli}(0.1)$ is a binary random variable, and $\mathbb{I}(\cdot)$ denotes the indicator function.

2.3.3 Joint training

The model is trained to minimize the linear combination of generation loss and contrastive loss:

$$\mathcal{L} = \lambda \mathcal{L}_{gen} + (1 - \lambda)\mathcal{L}_{ctr} \tag{10}$$

To enable precise retrieval in downstream scenarios, contrastive learning must be conducted with a large training batch. However, the native parallel running of two training tasks requires significant GPU memory, which severely limits the achievable batch size. To address this challenge, we propose a **memory-efficient joint training** to handle both tasks (Figure 3). Specifically, for each training sample $x_i = (q_i, t_i, d_i)$, we encode it once and share the encoding result between the two tasks. The generative loss is calculated based on each of the output embeddings from t_i tokens; while the contrastive loss is derived based on the output embeddings from <emb> tokens. This allows the generation task to consume almost no additional memory when trained alongside contrastive learning task, thereby enabling a substantial increase in batch size.

2.4 Inference

The well-trained O1 embedder $\mathcal{M}$ is applied for retrieval tasks through thinking and embedding. First, O1 embedder is prompted to generated multiple thoughts towards the input query which comprehensively uncover the query's hidden information needs: $t_i \leftarrow \mathcal{M}.\text{think(q)}, i = 1, ..., k$. Next, the thought-augmented queries are independently encoded and aggregated. In this place, we simply adopt mean pooling as the aggregation function in Eq. 3, which produces the following thought-augmented embedding:

$$\hat{\boldsymbol{v}}_q \leftarrow \sum_k \mathcal{M}.\,\text{enc}(q, t_i) \Big/ k \tag{11}$$

Finally, the top-N documents D^* are retrieved based on their embedding similarity with $\hat{\boldsymbol{v}}_q$:

$$D^* \leftarrow \text{top-N}(\{\langle \hat{\boldsymbol{v}}_q, \boldsymbol{v}_d \rangle | D\}) \tag{12}$$

3 Experiment

We make comprehensive evaluation of our approach with a focus on the following research questions.

RQ 1. Can O1 Embedder outperform baselines after fine-tuning, generalize out-of-domain, and excel at complex tasks?

RQ 2. What is the impact of the thinking operation?

RQ 3. How does joint multi-task training benefit performance? Is a unified model better than separate ones for thinking and retrieval?

With these research questions, we design our experimental studies, whose settings are presented in the Appendix B.

3.1 In-domain Performance

Method	MS MARCO Dev		DL'19	DL'20
	MRR@10	Recall@1k	nDCG@10	nDCG@10
BM25	18.4	85.3	50.6	48.0
HyDE	-	-	61.3	57.9
query2doc	41.5	98.7	74.9	72.5
ANCE	33.0	95.9	64.8	61.5
TAS-B	34.3	97.6	72.2	69.2
coCondenser	38.2	98.4	69.8	68.4
SimLM	41.1	98.7	71.4	69.7
RetroMAE	41.6	98.8	68.1	70.6
RepLLaMA	41.2	99.4	74.3	72.1
Promptriever	41.0	99.4	73.2	72.3
O1-E w/o T	41.7	99.4	73.7	72.3
O1-E	**43.1***	**99.5**	**75.3***	**74.4***

Table 1: In-domain evaluation results evaluated on MS MARCO, DL'19, and DL'20. O1-E means the O1 Embedder and O1-E w/o T indicates the variational form of O1 embedder, which disables the thinking operation. * indicates that the difference between O1-E and O1-E w/o T is statistically significant at 0.05 level by paired t-test.

The in-domain evaluation results are presented in Table 1, where the O1 Embedder outperforms all other models across all evaluation metrics, including MRR@10, Recall@1k, and nDCG@10, on the MSMARCO, DL'19, and DL'20 datasets. Specifically, our model achieves an MRR@10 of 43.1 (**+1.9%** improvement) and a Recall@1k of 99.5 on MSMARCO, along with nDCG@10 scores of 75.3 (**+1.0%** improvement) and 74.4 (**2.1%** improvement) on DL'19 and DL'20, respectively, compared to RepLLaMA. These results demonstrate the effectiveness of our model and its enhanced thinking capabilities in retrieval tasks. We also compared our method with some query expansion methods, including HyDE and query2doc. The main difference between our method and these methods is that they still rely on external large models, which increases the complexity of the chain. At the same time, our method also performed better in terms of results. Against query2doc (41.5 MRR@10, 74.9/72.5 nDCG@10), O1-E shows consistent gains in MRR@10 (**+1.6%**), DL'19 (**+0.4%**), and DL'20 (**+1.9%**), while maintaining top-tier Recall@1k (99.5). These improvements validate the effectiveness of our joint training strategy integrating thinking and embedding abilities.

Furthermore, when comparing LLM-based model with BERT-based model, it is clear that larger models, such as the LLM-based RepLLaMA and Promptriever, generally outperform smaller BERT-based models like SimLM, coCondenser. For example, RepLLaMA achieves an nDCG@10 of 74.3 on DL'19 and 72.1 on DL'20, surpassing all smaller models. The O1 Embedder without thinking (O1 embedder w/o T), a variation that disables the thinking operation, yields similar results as RepLLaMA, suggesting that the embedding capability is effectively established through multi-task training. However, with the addition of thinking, the full O1 Embedder demonstrates a substantial improvement, highlighting the power of the "thinking" enhancement.

3.2 O.O.D. Performance

The out-of-distribution (o.o.d.) evaluation results are shown in Table 2, where the O1 Embedder consistently outperforms across all nine datasets. On average, our model achieves a **2.3%** improvement over the baseline, which is a significant boost and underscores the strong generalization capabilities of our approach. Additionally, for each of the nine datasets, our model sets the highest performance except for FiQA, where it falls behind CPT-XL 175B. This highlights that our model excels across various scenarios, especially the retrieval tasks involving complex reasoning, such as HotPotQA (multi-hop question-answering) and CosQA (code-search).

Several interesting conclusions can be derived from the table. The thinking mechanism leads to more significant improvements on certain OpenQA

Method	Model size	T-Covid	NQ	HQA	FiQA	Touche	DBPedia	FEVER	SciFact	CosQA	Average
Contriever	0.1B	59.6	49.8	63.8	32.9	23.0	41.3	75.8	67.7	14.2	47.6
CPT-L	6B	56.2	-	64.8	45.2	30.9	41.2	75.6	74.4	-	-
CPT-XL	175B	64.9	-	68.8	**51.2**	29.1	43.2	77.5	75.4	-	-
OpenAI-Ada-002	-	81.3	48.2	65.4	41.1	28.0	40.2	77.3	73.6	28.9	53.7
RepLLaMA	7B	84.7	62.4	68.5	45.8	30.5	43.7	83.4	75.6	32.3	58.5
Promptriver	7B	84.6	62.6	69.5	46.6	32.0	45.2	82.8	76.3	32.8	59.2
O1-E w/o T	7B	84.5	62.9	69.8	45.0	33.8	44.4	82.5	75.8	32.9	59.1
O1-E	7B	**85.6***	**66.8***	**72.8***	46.6*	**36.7***	**47.3***	**84.9***	**77.4***	**34.1***	**61.4***

Table 2: Out-of-domain evaluation results measured by nDCG@10. The symbols are the same as those in Table 1.

datasets. For instance, the NQ dataset shows a **3.9%** improvement, while HotPotQA sees a **3.0%** boost. However, while there are improvements in some specialized domains, they are not as substantial. For example, in TREC-Covid (medical domain), FiQA (financial domain), and SciFact (scientific domain), the improvements brought by the thinking mechanism are only **0.9%**, **1.7%**, and **1.6%**, respectively. We believe this is because LLMs perform well on general OpenQA questions after training on large-scale corpora, but often lack specialized domain data. As a result, the generated content can sometimes be noisy or even hallucinated. While the thinking mechanism still provides some enhancement, its impact is not as pronounced in these domains. Thus, refining the generated thoughts will be an important issue for future research.

3.3 Complex Task Performance

	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	AVG
No Thought								
BM25	18.9	27.2	14.9	12.5	13.6	18.4	15	17.21
RepLLaMA	10.7	19.1	14.9	17.7	12.5	9.6	11.3	13.69
Promptriever	10.3	19.0	16.8	16.6	11.1	9.1	14.5	13.91
E5	18.6	26.0	15.5	15.8	16.3	11.2	18.1	17.36
GritLM	24.8	32.3	18.9	19.8	17.1	13.6	17.8	20.61
O1-E w/o T	23.4	32.9	16.0	19.6	16.7	15.2	16.9	20.10
With Thought Augmentation								
E5 + GritLM	24.1	36.7	18.3	21.1	10.7	16.1	14.7	20.24
GritLM	24.7	31.5	18.5	21.1	14.0	12.3	**20.7**	20.40
O1-E	**27.9***	**38.9***	**21.8***	**24.3***	**20.4***	**16.2***	18.7*	**24.03***

Table 3: Complex task evaluation results on Bright-StackExchange measured by nDCG@10. Except for O1-E, all other thinking enhancement methods use GritLM as the generation model to generate the thought, which is comparable in 7B size to O1-E. The symbols are the same as those in Table 1.

To address the research question of whether O1 Embedder can outperform baselines on complex tasks, we evaluate all models on the Bright-StackExchange dataset (Su et al., 2025), which focuses on challenging retrieval scenarios requiring deep reasoning across diverse domains (e.g., biology, economics, psychology, and sustainability). Results measured by nDCG@10 are presented in Table 3, with comparisons between models operating without explicit thought processes ("No Thought") and those leveraging thought augmentation ("With Thought Augmentation").

In the *No Thought* setting, O1-E w/o T performs comparably to strong dense baselines such as E5 and GritLM, while overall performance remains limited across models, highlighting the difficulty of the benchmark. With thought augmentation, **O1-E demonstrates clear and consistent improvements across domains**. Under a controlled comparison where all baselines employ GritLM (7B) for thought generation, O1-E achieves the highest nDCG@10 on six out of seven tasks, and obtains the best average performance overall. Specifically, O1-E reaches an average nDCG@10 of **24.03**, outperforming the strongest augmented baseline by **3.63** points and improving over its non-thought variant by **3.93** points.

These results show that O1-E generalizes effectively to complex, reasoning-intensive retrieval tasks, and can robustly leverage thought signals to achieve superior performance across diverse domains.

3.4 Impact of Thought

The comparisons between the O1 Embedder and the O1-E w/o T in Tables 1 and 2 clearly demonstrate the significant impact of incorporating the thinking mechanism. Specifically, on the MS MARCO benchmark (Table 1), the O1 Embedder shows substantial improvements across all metrics, with notable gains in MRR@10 (+1.4%) and nDCG@10 for both DL'19 (+1.6%) and DL'20 (+2.1%). Similarly, in the zero-shot evaluation

(Table 2), the O1 Embedder outperforms its counterpart across all datasets, achieving an average nDCG@10 score of 61.4, 2.3 points higher than the O1-E w/o T. This performance boost can be attributed to the thinking mechanism, which allows the model to generate additional contextual information during the encoding process. By incorporating this supplementary information, the model enhances its understanding of the user's query, leading to more accurate and effective retrieval. Additionally, it is worth noting that even in its "fast" mode (represented by O1-E w/o T), the model achieves results comparable to those of RepLLaMA. This highlights that our model maintains competitive retrieval performance, even without the generated thought, underscoring its flexibility and adaptability.

3.5 One Model vs Seperate Model

	Seperate Model			One Model		
	Base	with T	Δ	Base	with T	Δ
DL'19	74.3	72.4	-1.9	73.7	75.3	+1.6
DL'20	72.1	72.6	+0.5	72.3	74.4	+2.1
Trec-Covid	84.7	79.7	-5.0	84.5	85.6	+1.1
NQ	62.4	65.1	+2.7	62.9	66.8	+3.9
HotPotQA	68.5	71.7	+3.2	69.8	72.8	+3.0
FiQA	45.8	40.2	-5.6	45.0	46.6	+1.6
Touche	30.5	33.3	+2.8	33.8	36.7	+2.9
DBPedia	43.7	44.2	+0.5	44.4	47.3	+2.9
FEVER	83.4	85.5	+2.1	82.5	85.0	+2.5
SciFact	75.6	74.2	-1.4	75.8	77.4	+1.6
CosQA	32.3	33.0	+0.7	32.9	34.1	+1.2
Avg	61.2	61.1	-0.1	61.6	63.8	+2.2

Table 4: Exploration of joint training. With the joint training of thinking and embedding ability, O1 embedder achieves a substantial improvement in retrieval performance. In contrast, two seperate models leads to a suboptimal performance due to the incompatibility of the two modules. In this table, "Base" denotes retrieval directly with query, "with T" denotes retrieval using the query with thought, "Δ" denotes the improvement.

To analyze the impact from joint training, we make analysis on whether existing retrieval models can make effective use of the generated thoughts in a training-free manner. For this purpose, we introduce a stand-alone generator for a well-trained retriever, which generates thoughts for its presented queries. In our experiment, we leverage RepLLaMA as the retriever and GPT-4o-mini as the generator for our experiments.

The results of this approach are shown in Table 4. While incorporating thoughts results in mild improvements on datasets like NQ and HotPotQA, it causes declines on others, such as Trec-Covid and FiQA. Overall, this approach leads to only minor gains on some tasks but results in a slight decrease of 0.1 in overall performance. In contrast, our model consistently improves the retrieval performance across all datasets. This suggests that, with joint training, our model can better utilize the generated thoughts. The untrained RepLLaMA model, however, appears to be negatively impacted by the potential noise within the generated thoughts, leading to worse results, particularly in specialized domains like Trec-Covid, FiQA, and SciFact. In brief, the above observation indicates that our method not only generates useful thoughts for retrieval, but also learns to make effective use of the thoughts through joint training.

4 Conclusion

In this paper, we introduce O1 Embedder, a novel retrieval model that performs slow-thinking before executing retrieval actions. This approach allows the model to better understand the underlying information needs within the query, aiding in the identification of semantic relevance for complex retrieval tasks. Leveraging our tailored data production method, we generate long-form thoughts optimized for retrieval utility. Additionally, our proposed multi-task training method effectively establishes both the model's thinking and embedding capabilities. We conduct comprehensive experiments on popular evaluation benchmarks, and the results demonstrate that O1 Embedder significantly outperforms existing methods, achieving substantial improvements in retrieval performance across both in-domain, out-of-domain and complex task scenarios.

Our work lays the foundation for future research in advanced retrieval models with thinking capabilities. Future directions include expanding the thinking process to multi-round interactions, exploring lightweight distillation techniques, and applying the approach to other retrieval tasks. The O1 Embedder marks a promising paradigm for next-generation IR systems, showcasing the potential of integrating the classic dense retrieval methods with large language models' outstanding thinking abilities.

5 Limitations

This work has several limitations. First, the effectiveness of O1 Embedder depends on the quality of the thoughts generated by the expert LLM, which may be noisy or less informative in specialized domains where the LLM lacks sufficient domain knowledge. Second, introducing the thinking process incurs additional inference-time cost, as multiple thoughts must be generated and encoded, which may limit applicability in latency-sensitive or large-scale retrieval settings; the current framework does not yet adaptively decide when slow-thinking is necessary. Finally, the thinking module is restricted to single-round, unstructured thoughts with a fixed mean pooling aggregation, leaving more expressive reasoning structures and adaptive aggregation strategies unexplored.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and 1 others. 2020. Overview of touché 2020: argument retrieval. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*, pages 384–395. Springer.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv preprint*. ArXiv:2402.03216 [cs].

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024b. Are more llm calls all you need? towards scaling laws of compound inference systems. *Preprint*, arXiv:2403.02419.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. *Preprint*, arXiv:2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2019 deep learning track. *Preprint*, arXiv:2003.07820.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2024. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36.

Luyu Gao and Jamie Callan. 2022. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2843–2853, Dublin, Ireland. Association for Computational Linguistics.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. Dbpedia-entity v2: A test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1265–1268, New York, NY, USA. Association for Computing Machinery.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *arXiv preprint*. ArXiv:2104.06967 [cs].

Junjie Huang, Duyu Tang, Linjun Shou, Ming Gong, Ke Xu, Daxin Jiang, Ming Zhou, and Nan Duan. 2021. CoSQA: 20,000+ web queries for code search and question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5690–5700, Online. Association for Computational Linguistics.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*.

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Mei Kobayashi and Koichi Takeda. 2000. Information retrieval on the web. *ACM computing surveys (CSUR)*, 32(2):144–173.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. 2022. Generative multi-hop retrieval. *Preprint*, arXiv:2204.13596.

Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. 2024a. Llama2Vec: Unsupervised Adaptation of Large Language Models for Dense Retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3490–3500, Bangkok, Thailand. Association for Computational Linguistics.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024b. Making text embedders few-shot learners. *arXiv preprint arXiv:2409.15700*.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024c. Making Text Embedders Few-Shot Learners. *arXiv preprint*. ArXiv:2409.15700 [cs].

Lei Li, Xiangxu Zhang, Xiao Zhou, and Zheng Liu. 2024d. Automir: Effective zero-shot medical information retrieval without relevance labels. *Preprint*, arXiv:2410.20050.

Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Yichun Yin, Hao Zhang, Yong Liu, Yasheng Wang, and Ruiming Tang. 2024e. Coir: A comprehensive benchmark for code information retrieval models. *Preprint*, arXiv:2407.02883.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Zheng Liu, Shitao Xiao, Yingxia Shao, and Zhao Cao. 2023. RetroMAE-2: Duplex Masked Auto-Encoder For Pre-Training Retrieval-Oriented Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2635–2648, Toronto, Canada. Association for Computational Linguistics.

Zheng Liu, Yujia Zhou, Yutao Zhu, Jianxun Lian, Chaozhuo Li, Zhicheng Dou, Defu Lian, and Jian-Yun Nie. 2024. Information retrieval meets large language models. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 1586–1589, New York, NY, USA. Association for Computing Machinery.

Kun Luo, Minghao Qin, Zheng Liu, Shitao Xiao, Jun Zhao, and Kang Liu. 2024. Large language models as foundations for next-gen dense retrieval: A comprehensive empirical assessment. *arXiv preprint arXiv:2408.12194*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pages 2421–2425, New York, NY, USA. Association for Computing Machinery.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative Representational Instruction Tuning. *arXiv preprint*. ArXiv:2402.09906 [cs].

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative representational instruction tuning. *Preprint*, arXiv:2402.09906.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, and 6 others. 2022. Text and Code Embeddings by Contrastive Pre-Training. *arXiv preprint*. ArXiv:2201.10005 [cs].

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large Dual Encoders Are Generalizable Retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. 2023. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S Siegel, Michael Tang, and 1 others. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *arXiv preprint arXiv:2407.12883*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. 2025. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *Preprint*, arXiv:2407.12883.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *arXiv preprint*. ArXiv:2104.08663 [cs].

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. Trec-covid: constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54(1).

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023a. SimLM: Pre-training with Representation Bottleneck for Dense Passage Retrieval. *arXiv preprint*. ArXiv:2207.02578 [cs].

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. *arXiv preprint*. ArXiv:2212.03533 [cs].

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023b. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Liang Wang, Nan Yang, and Furu Wei. 2023c. Query2doc: Query Expansion with Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423, Singapore. Association for Computational Linguistics.

Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. 2024. Uniir: Training and benchmarking universal multimodal information retrievers. In *European Conference on Computer Vision*, pages 387–404. Springer.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Orion Weller, Benjamin Van Durme, Dawn Lawrie, Ashwin Paranjape, Yuhao Zhang, and Jack Hessel. 2024. Promptriever: Instruction-Trained Retrievers Can Be Prompted Like Language Models. *arXiv preprint*. ArXiv:2409.11136 [cs].

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 538–548, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–649, Washington DC USA. ACM.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv preprint*. ArXiv:2007.00808 [cs].

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. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve anything to augment large language models. *arXiv preprint arXiv:2310.07554*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K Qiu, and Lili Qiu. 2024a. Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your llms use external data more wisely. *arXiv preprint arXiv:2409.14924*.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024b. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60.

Junjie Zhou, Zheng Liu, Shitao Xiao, Bo Zhao, and Yongping Xiong. 2024. Vista: Visualized text embedding for universal multi-modal retrieval. *arXiv preprint arXiv:2406.04292*.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

A Related Work

In this section, we make discussions on the related literature from two perspectives: the progress on dense retrieval, and the introduction of reasoning capability to LLMs.

A.1 Dense Retrieval

Dense retrieval has made significant strides in retrieval precision, driven by the advancements in foundation models and training techniques. Early breakthroughs involved fine-tuning preliminary pre-trained models, such as BERT and RoBERTa (Devlin, 2018; Liu, 2019), for dense retrieval, which already demonstrated competitive performance compared to traditional methods like BM25. At the same time, the scope of dense retrieval was substantially expanded thanks to the adoption of multi-lingual (Izacard et al., 2022; Chen et al., 2024a) and multi-modal pre-trained models (Wei et al., 2024; Zhou et al., 2024). The introduction of more advanced training strategies, such as retrieval-oriented adaptation (Xiao et al., 2022; Liu et al., 2023; Wang et al., 2023a), hard negative mining (Xiong et al., 2020), batch size expansion (Qu et al., 2020), and knowledge distillation from cross-encoders (Hofstätter et al., 2021), has continually contributed to the improvement of dense retrieval's performance.

In addition to the improvement on retrieval accuracy, it becomes increasingly emphasized to develop multi-task retrievers for general-purpose retrieval applications. Recent studies showed that the retrievers' generalization ability can be substantially enhanced by scaling-up the training scale (Su et al., 2022) and model architecture (Ni et al., 2022). Based on these inspirations, people have made significant expansion of pre-training and fine-tuning tasks, leading to a series of popular retrievers for general-purpose applications, such as BGE, E5, and GTE (Wang et al., 2022; Li et al., 2023). Meanwhile, people also introduce large language models (LLMs) as the retrievers' backbones, which brings forth significant improvements in retrieval performance. For example, RepLLaMA presents a powerful dense retriever by directly fine-tuning a pre-trained Llama (Wang et al., 2023b). Llama2Vec further enhances RepLLaMA by incorporating unsupervised adaptation of the pre-trained Llama (Li et al., 2024a). Promptriver (Weller et al., 2024), built on RepLLaMA, equips the retrieval model with the capability to follow instructions. Methods like NV-Embed, ICL-Embedder and Qwen-3-Embedding achieves additional improvement through continual training with extensive fine-tuning data (Lee et al., 2024; Li et al., 2024b; Zhang et al., 2025). Today, LLM-powered retrievers have dominated nearly all major benchmarks in IR-related evaluation.

Despite these remarkable advancements, existing methods are primarily designed for direct semantic matching in popular applications like web search and question-answering. They still face challenges with zero-shot retrieval in completely new scenarios that differ significantly from their source domains (Gao et al., 2022; Zhu et al., 2023). In addition, they are insufficient for more complex retrieval tasks which require intensive reasoning to identify semantic relationships (Su et al., 2024).

A.2 LLMs' Reasoning Ability

The reasoning capabilities of large language models (LLMs) have been significantly enhanced with techniques that simulate human-like problem-solving processes. A major breakthrough in this area is Chain-of-Thought (CoT) (Wei et al., 2022), which prompts LLMs to tackle complex problems by decomposing them into multiple reasoning steps. Building on this progress, the Self-Consistency method improves reasoning robustness by sampling multiple reasoning paths from the LLM and selecting the final answer through majority voting (Feng et al., 2024). For scenarios requiring more exploratory reasoning, the Tree of Thoughts (ToT) method (Yao et al., 2024) extends CoT by structuring the problem-solving process as a tree. At each node, the LLM generates candidate intermediate steps, evaluates their feasibility, and backtracks from dead ends. Further advancing this paradigm, Graph of Thoughts (GoT) (Besta et al., 2024) replaces the tree structure with a directed acyclic graph (DAG), enabling LLMs to merge or refine reasoning steps as needed. The reasoning capability of large language models (LLMs), or the "think before action" workflow, represents a new paradigm that sets them apart from traditional language models. In addition to the usual strategies of scaling model size, datasets, and training computation (Kaplan et al., 2020; Hoffmann et al., 2022), the expansion of inference computation, or test-time scaling (Wu et al., 2024; Chen et al., 2024b; Sardana et al., 2023), becomes another important factor in driving the improvement of LLMs. This capability has been significantly enhanced and showcased by recent reasoning-capable LLMs, such as OpenAI's O1 and O3, DeepSeek's R1 (Guo et al., 2025), and Gemini 2.0/3.0[1]. These models adopt a "slow-thinking" approach when handling complex problems: instead of providing an immediate answer, they first generate verbose, structured reasoning before arriving at a final solution. This method has allowed LLMs to achieve elite-level performance in areas like coding and mathematical proofs.

The reasoning capability also offers a significant advantage in addressing the challenges posed by traditional retrieval methods. However, current embedding models primarily focus on generating discriminative data representations, which leaves the development of reasoning capabilities largely unexplored.

B Experimental Settings

B.1 Datasets

O1 embedder is trained by the thought-augmented queries created from the MS MARCO (passage retrieval) dataset (Bajaj et al., 2018). The well-trained model is evaluated based on in-domain, out-of-domain and complex tasks datasets. For in-domain evaluation, we utilize

[1]https://deepmind.google/technologies/gemini/flash-thinking/

MS MARCO (dev), TREC DL19 (Craswell et al., 2020), and TREC DL20 (Craswell et al., 2021) datasets. For out-of-domain evaluation, we incorporate the following eight question-answering datasets from BEIR (Thakur et al., 2021), including SciFact (Wadden et al., 2020), TREC-COVID (Voorhees et al., 2021), DBpedia (Hasibi et al., 2017), NQ (Kwiatkowski et al., 2019), HotPotQA (Yang et al., 2018), FiQA (Maia et al., 2018), Touche (Bondarenko et al., 2020), FEVER (Thorne et al., 2018), along with a popular dataset on code-search: CosQA (Huang et al., 2021). For complex tasks, we ultilize BrightStackExchange dataset (Su et al., 2025). This dataset contains seven challenging tasks, including difficult problems in biology, physics, coding, and other fields. Furthermore, this dataset does not contain any training data, so it relies entirely on the model's zero-shot capability to retrieve relevant documents. All of these datasets consist of asymmetric retrieval tasks, where the query and document are presented in very different forms. As a result, the relationships between query and document can be more effectively identified through appropriate reasoning. We exclude common paraphrasing datasets, such as Quora, as they only involve simple similarity comparisons. Following prior works (Ma et al., 2024; Weller et al., 2024; Muennighoff et al., 2024), we use MRR@10 and Recall@1k as metrics for MS MARCO-related tasks, and NDCG@10 for other datasets. Evaluations on o.o.d. datasets strictly adhere to the BEIR protocol, which prohibits task-specific fine-tuning.

B.2 Baselines

We choose a wide variety of popular retrievers as our baselines, such as BM25, a commonly used sparse retrieval method, and ANCE (Xiong et al., 2020), TAS-B (Hofstätter et al., 2021), coCondenser (Gao and Callan, 2022), SimLM (Wang et al., 2023a), which fine-tune BERT-based pre-trained models using MS MARCO dataset. We also compared some classic query expansion methods: HyDE (Gao et al., 2022) and query2doc (Wang et al., 2023c). This method uses an external large model for query expansion to generate possible candidate documents. In addition, We introduce the LLM-based methods, including RepLLaMA (Wang et al., 2024), Promptriver (Weller et al., 2024). RepLLaMA fine-tunes a pre-trained LLM on MS MARCO, resulting in superior retrieval performance across various downstream tasks. While Promptriever builds on RepLLaMA by enhancing the model's instruction-following capabilities, which further improves the retrieval performance on top of tailored prompts. Both LLM-based methods are fine-tuned from the same pre-trained backbone (Llama-2-7B) as our default setting, thus ensuring a fair comparison in terms of model scale. Note that we exclude several other popular retrievers from our O.O.D. evaluation on BEIR benchmark, including BGE (Xiao et al., 2024), E5 (Wang et al., 2022), M3 (Chen et al., 2024a), and recent LLM-based methods like E5-Mistral (Wang et al., 2023b), GRITLM (Muennighoff et al., 2025), ICL-Embedder (Li et al., 2024c), and NV-Embed (Lee et al., 2024). These models utilize significantly more training data beyond MS MARCO (the only training dataset used by our method and our included baselines), many having strong overlaps with the evaluation tasks on BEIR. This overlap makes it difficult to assess zero-shot retrieval performance in out-of-distribution (o.o.d.) settings.

B.3 Implementation Details

During the data preparation stage, we leverage a powerful open-source LLM: Llama-3.1-70B-Instruct[2], to generated the candidate thoughts. We employ BM25, BGE-EN-large-v1.5, GTE-large, and Stella-EN-1.5B-v5, to serve the retrieval committee. The evaluation is primary made based on a Llama-2-7B backbone (Touvron et al., 2023), with other alternative LLMs analyzed in the extended study. The training process follows RepLLaMA's recipe (Ma et al., 2024), where all projection layers (q_proj k_proj v_proj o_proj gate_proj down_proj up_proj) of the LLM are fine-tuned via LoRA, with rank set to 32 and alpha set to 64. We used BF16 for training, with learning rate set to 1×10^{-4}. The training process takes place on 8xA800 GPUs, with a batch size set to 64 (8 per-device). We introduce 15 hard negatives for each query. The maximum query length was set to 32, and the maximum passage length was set to 192. The max tokens were set to 256 for thought generation.

C Memory-Efficient Joint Training

During training, we employ a memory-efficient joint training strategy that shares query encodings between the thought generation and contrastive learning tasks(See Section 2.3.3). Table 5 show

[2]https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct

the specific experimental data of the strategy to save GPU memory.

Setting	Share query	No share
Qwen1.5B/BS=64	19.1G	27.8G

Table 5: GPU memory consumption comparison between joint training with and without query encoding sharing.

D Robustness

In this section, we verify the robustness of our method from two perspectives. First, we implement our method based on different pre-trained architectures, including Llama, Mistral, Qwen. Second, we also introduce LLMs of different sizes (ranging from 0.5B to 8B) for evaluation.

	In-Domain			o.o.d.
	MS MARCO	DL'19	DL'20	AVG
Llama-2-7B	43.1	75.3	74.4	61.4
Mistralv0.3-7B	43.5	77.0	75.6	61.4
Llama-3.1-8B	43.5	76.2	74.5	61.6
Qwen2.5-7B	43.3	76.4	74.7	61.2
Qwen2.5-3B	42.5	76.3	74.5	60.3
Qwen2.5-1.5B	41.9	74.0	73.5	58.7
Qwen2.5-0.5B	40.5	73.6	71.4	55.4

Table 6: The impact from using different backbone models of variant pre-trained architectures and model sizes. O1 Embedder well maintains a strong performance throughout these settings.

Impact of different model backbone. The previous experiments primarily used Llama-2-7B as the backbone, which is consistent with RepLLaMA and promptriever. To verify the generalizability of our approach, we repeat the same experiment with implementations on different backbone LLMs. The results in Table 6 demonstrate our effectiveness across different settings. Notably, our approach well maintains a strong retrieval performance in both in-domain and out-of-domain evaluations. This observation suggests that our method is generally effective with various architectures, regardless of their difference in pre-trained capabilities.

Impact of different model sizes. We can also clearly observe the significant impact of model size on performance in Table 6. As the size of the Qwen2.5 models decreases, there is a noticeable drop in effectiveness across all evaluated datasets. While the 3B model performs similarly to the 7B model, further reductions in size lead to more pronounced performance declines. This is partly because larger models tend to have better generalization capabilities, benefiting from training on more extensive corpora during pre-training. It's worth noting that even a 1.5B model, through retrieval with thought, performs comparably to the 7B RepLLaMA, which further highlights the advantage of our approach.

E Algorithm

Algorithm 1 Data Production Process

Require:
Query-document dataset $D = \{(q_i, d_i)\}_N$
Teacher model for generating thoughts LLM
Retrieval models $R = \{r_1, r_2, ..., r_{|R|}\}$
Example count m, candidate thoughts k
Prompt tamplate: PROMPT, Instruction: Ins
Functions:

- SAMPLEEXAMPLES: Samples m examples from D
- σ^r: Similarity score function of Retrieval model r
- VOTING: majority voting function

Ensure:
Enhanced dataset $\hat{D} = \{(q_i, t_i, d_i)\}_N$

Initialize $\hat{D} \leftarrow \emptyset$
for each $(q_i, d_i) \in D$ **do**
  for $j = 1$ to k **do**
    $E \leftarrow \text{SAMPLEEXAMPLES}(D)$
    $P \leftarrow \text{PROMPT}.format(Ins, E, q_j)$
    $t_j \leftarrow LLM.generate(P)$
  end for
  for $j = 1$ to $|R|$ **do**
    $t_r^* \leftarrow \text{ARGMAX}(\{\sigma^r(t_j, d)\}_{j=1...k})$
  end for
  $t_i \leftarrow \text{VOTING}(t_r^*)_{r \in R}$
  $\hat{D} \leftarrow \hat{D} \cup \{(q_i, t_i, d_i)\}$
end for
return $\hat{D}$

The Algorithm 1 outlined in the Data production process in section 2.2. For each query-document pair (q_i, d_i) in the dataset, the algorithm first samples m examples from D to create a prompt, which includes specific instructions. It then generates k candidate thoughts t_j by formatting the prompt with the sampled examples. Next, for each retrieval

model r, it calculates the similarity scores between the candidate thoughts and the document d_i, selecting the thought with the highest score as t_r^*. Finally, the algorithm aggregates these top thoughts through a majority voting mechanism to determine the final thought t_i. The enhanced dataset $\hat{D}$ is then constructed, consisting of the original queries, the generated thoughts, and their corresponding positive documents.

F Theoretical Analysis

We conduct a brief theoretical analysis to illustrate the effectiveness of the O1 Embedder. Given a query q and its relevant document d, their normalized embeddings are denoted by $\boldsymbol{v}_q$ and $\boldsymbol{v}_d$, respectively. The O1 Embedder generates k thoughts $\{t_i\}_{i=1}^k$, each of which is encoded into a corresponding normalized embedding $\boldsymbol{v}_{t_i} = \mathcal{M}.\text{enc}(t_i)$. To obtain the thought-augmented query embedding, each thought t_i is concatenated with the original query q to form a query-thought pair (q, t_i), which is then encoded into a vector. Then a mean pooling operation is applied over the resulting embeddings to get the final embedding:

$$\hat{\boldsymbol{v}}_q = \frac{1}{k}\sum_{i=1}^{k} \mathcal{M}.\text{enc}(q, t_i).$$

Since each thought is generated by a fine-tuned generator, its relevance to the relevant document is guaranteed. Therefore, we propose the following hypothesis:

Hypothesis 1 (Thought Alignment Hypothesis)**.** *There exists a constant $\kappa > 0$ such that for every generated thought t_i, its embedding satisfies*

$$\langle \boldsymbol{v}_{t_i}, \boldsymbol{v}_d \rangle \geq \kappa, \quad \forall i = 1, \ldots, k.$$

Motivated by empirical observations that transformer- based encoders exhibit a degree of linearity in their latent spaces, the encoding function $\mathcal{M}.\text{enc}(\cdot)$ is assumed to approximately satisfy a linear compositionality property with respect to concatenation, such that the embedding of the concatenated text (q, t_i) can be well approximated by a convex combination of the individual embeddings:

$$\mathcal{M}.\text{enc}(q, t_i) \approx \lambda \boldsymbol{v}_q + (1-\lambda)\boldsymbol{v}_{t_i}, \quad \lambda \in (0, 1),$$

where λ reflects the relative contribution of the original query embedding within the concatenated representation. Based on this, we make the following hypothesis about the final thought-augmented query embedding:

Hypothesis 2. *The thought-augmented query embedding, aggregated over k thoughts, can be approximated as*

$$\hat{\boldsymbol{v}}_q = \frac{1}{k}\sum_{i=1}^{k} \mathcal{M}.\text{enc}(q, t_i) \approx \lambda \boldsymbol{v}_q + (1-\lambda)\cdot\frac{1}{k}\sum_{i=1}^{k} \boldsymbol{v}_{t_i}.$$

Theorem 1 (Similarity Boost from Thought Augmentation)**.** *Under Hypothesis 1 and Hypothesis 2, it holds that*

$$\langle \hat{\boldsymbol{v}}_q, \boldsymbol{v}_d \rangle - \langle \boldsymbol{v}_q, \boldsymbol{v}_d \rangle \geq (1-\lambda)\left(\kappa - \langle \boldsymbol{v}_q, \boldsymbol{v}_d \rangle\right) - \mathcal{O}\left(\frac{1}{k}\right).$$

Proof. By linearity of the inner product, we have

$$\begin{aligned}
\langle \hat{\boldsymbol{v}}_q, \boldsymbol{v}_d \rangle - \langle \boldsymbol{v}_q, \boldsymbol{v}_d \rangle &= \hat{\boldsymbol{v}}_q^\top \boldsymbol{v}_d - \boldsymbol{v}_q^\top \boldsymbol{v}_d \\
&\approx \left(\lambda \boldsymbol{v}_q + (1-\lambda)\cdot\frac{1}{k}\sum_{i=1}^{k}\boldsymbol{v}_{t_i}\right)^\top \boldsymbol{v}_d - \boldsymbol{v}_q^\top \boldsymbol{v}_d \\
&= (1-\lambda)\left(\frac{1}{k}\sum_{i=1}^{k}\langle \boldsymbol{v}_{t_i}, \boldsymbol{v}_d \rangle - \langle \boldsymbol{v}_q, \boldsymbol{v}_d \rangle\right) \\
&\geq (1-\lambda)\left(\kappa - \langle \boldsymbol{v}_q, \boldsymbol{v}_d \rangle\right) - \mathcal{O}\left(\frac{1}{k}\right).
\end{aligned}$$

□

The above analysis explains why the O1 Embedder improves retrieval performance: by augmenting the original query with thoughts closely aligned to the target document, the resulting embedding achieves higher similarity with relevant documents. The guaranteed relevance of each thought ensures that meaningful semantic information is injected. As more high-quality thoughts are added, approximation error decreases, leading to more stable and consistent similarity gains.

G Prompts

Table 7 shows the example of complete prompts and the response of the teacher model used to generate the training data. For each specific query, we randomly select 3 query and positive document pair to form the in-context examples. During the data generation process, we found that a significant portion of the responses duplicated the query provided. So we add "*Don't repeat the query, just give the response.*" in the system prompts.

H Case Study

In Table 8, we demonstrate an example of the generated thought and the ground-truth document

System Prompt
You are a helpful assistant. Your anwer should be follow the task description. Do not ask the user for further clarification. Don't repeat the query, just give the response.
User Prompt
Task: Think about a plausible response to address the query Examples: Query: love ranch movie cast Response: Love Ranch Love Ranch. Love Ranch is ...<omit> Query: what does physical medicine do Response:
Response
Physical medicine, also known as physiatry, is a branch of medicine that focuses on helping patients with physical disabilities or impairments to achieve maximum function and independence. Physiatrists, or physical medicine doctors, use a multidisciplinary approach to diagnose and treat a wide range of conditions, ...<omit>

Table 7: Example of the Prompt and the Response of the teacher model for data generation. For presentation purposes, we use <omit> to omit the long documentation in the example.

Query: what age was martin luther king when he was admitted
Thought: Martin Luther King Jr. was admitted to **Morehouse College in Atlanta**, Georgia **at the age of 15**. He attended the college from **1944** to 1948, where he earned a Bachelor of Arts degree in sociology.
Positive Doc: Dr. Martin L. King, Jr. and His Mentors: A Message for America Today If it was not for Benjamin Mays... Benjamin Mays was the president of **Morehouse College in Atlanta** when he met Martin Luther King, Jr. In **1944**, Martin Luther King was admitted to the college **at age 15**. ...

Table 8: Examples of the original query, thinking content and the positive document. The similar patterns between gnenrated content and the groundtruth are marked in **green**.

Query: Hayden is a singer-songwriter from Canada, but where does Buck-Tick hail from?
Thought: **Buck-Tick** is a **Japanese** rock band, and its members are from various parts of **Japan** and their music is a unique blend of alternative rock, gothic rock, and visual kei styles.
Positive Doc: Buck-Tick **Buck-Tick** (stylized as BUCK-TICK) is a **Japanese** rock band, formed in Fujioka, Gunma in 1983. The group has consisted of ...

Table 9: Another Example from HotPotQA (zero-shot).

to a complex multi-hop query. In this case, the query asks about Martin Luther King's age upon his admission to college. Our thought generated effectively uncovers useful contextual information, highlighting that King was admitted to Morehouse College at the age of 15 in 1944. This generated content not only answers the query directly but also enriches the context by providing additional details about the college and the timeframe of his attendance. **By generating such useful patterns, the embedding model can obtain crucial information related to the query, which results in a more precise retrieval result**.

To better understand why thought can better help with retrieval, we further analyzed the attention scores of the model for the simple example in Table 9. We calculate the attention weight of the <emb> token with each token t_j in the thought:

$$emb\_att(t_j) = \sum_i \frac{\exp\left(q^i_{<emb>} \cdot k^i_j\right)}{\sum_{l=1}^{|L|} \exp\left(q^i_{<emb>} \cdot k^i_l\right)}$$

where $q^i_{<emb>}$ and k^i_j are the "query" vector and the jth token's "key" vector of the ith attention head. $|L|$ is the total length of input sequence.

To enhance the clarity of our findings, we present the top 20 attention scores from the "<emb>" token in the generated thought, illustrated in Figure 4. The results reveal that the model assigns high scores to similar patterns found between the thought and the positive document, indicating its effectiveness in recognizing and focusing on relevant generated patterns. Notably, the word "Japan" emerges with the highest attention score, despite not appearing in the original query. This highlights the critical role of the generated thought in the

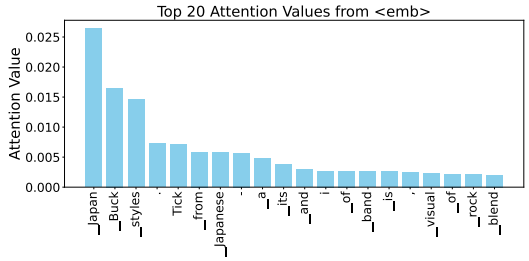

Figure 4: Top 20 Attention score from <emb> token in the thought

retrieval process, as it enables the model to supplement essential information that may be absent from the query.

In addition, we present another case from CosQA dataset demonstrating the capabilities of O1 embedder in Table 10, to effectively generate and retrieve relevant code snippet. The query posed, "python check if a directory is writable" illustrates a common programming challenge. The O1 embedder responds by generating a comprehensive thought that not only provides a direct solution using the os.access() function but also includes an example code snippet. This case highlights the model's ability to synthesize information and present it in an accessible format, thereby aiding more accurate query representation. Additionally, the thought generated by the model incorporates alternative methods, such as verifying if the path is indeed a directory before checking write permissions. This demonstrates the O1 embedder's depth of understanding by providing multiple approaches to the problem. Through this case, we illustrate that the O1 embedder is capable of producing coherent, contextually relevant outputs that significantly improve the utility of code retrieval systems.

Query: python check if a directory is writable

Thought:

To check if a directory is writable in Python, you can use the os.access() function. This function takes two arguments: the path to the directory and the mode you want to check (e.g. os.W_OK for writing). If the directory is writable, the function returns True, otherwise it returns False. Here's an example:

```
import os
path = '/path/to/directory'
if os.access(path, os.W_OK):
    print('The directory is writable')
else:
    print('The directory is not writable')
```

Alternatively, you can also use the os.path.isdir() function to check if the path is a directory, and then use the os.access() function to check the write permissions:

```
import os
path = '/path/to/directory'
if os.path.isdir(path) and os.access(path, os.W_OK):
    print('The directory is writable')
else:
    print('The directory is not writable')
```

Positive Doc:

```
def _writable_dir(path):
    """
    Whether `path` is a directory, to which the user has write access.
    """
    return os.path.isdir(path) and os.access(path, os.W_OK)
```

Table 10: An example of O1 Embedder solving a complex code retrieval problem.