Syntriever: How to Train Your Retriever with Synthetic Data from LLMs

Minsang Kim
Department of Computer Science and Engineering
Korea University
SK Telecom
South Korea
kmswin1@korea.ac.kr

Seungjun Baek*
Department of Computer Science and Engineering
Korea University
South Korea
sjbaek@korea.ac.kr

Abstract

LLMs have boosted progress in many AI applications. Recently, there were attempts to distill the vast knowledge of LLMs into information retrieval systems. Those distillation methods mostly use output probabilities of LLMs which are unavailable in the latest black-box LLMs. We propose Syntriever, a training framework for retrievers using synthetic data from black-box LLMs. Syntriever consists of two stages. Firstly in the distillation stage, we synthesize relevant and plausibly irrelevant passages and augmented queries using chain-of-thoughts for the given queries. LLM is asked to self-verify the synthetic data for possible hallucinations, after which retrievers are trained with a loss designed to cluster the embeddings of relevant passages. Secondly in the alignment stage, we align the retriever with the preferences of LLMs. We propose a preference modeling called partial Plackett-Luce ranking to learn LLM preferences with regularization which prevents the model from deviating excessively from that trained in the distillation stage. Experiments show that Syntriever achieves state-of-the-art performances on benchmark datasets from various domains in nDCG@K. The code is available at https://github.com/kmswin1/Syntriever.

1 Introduction

Large Language Models (LLMs) have become a core technology in various NLP applications such as chatbots (Achiam et al., 2023; Team et al., 2023) and coding assistants (Roziere et al., 2023; Guo et al., 2024). It is essential that the knowledge of LLMs is complemented by up-to-date information from external sources. To this end, retrieval-augmented generations (RAG) have been proposed and actively explored for various knowledge-intensive NLP tasks (Lewis et al., 2020; Guu et al., 2020; Lazaridou et al., 2022). RAG enhances the LLM performance without fine-tuning by incorporating external knowledge into LLMs through search and alleviates problems such as hallucination (Welleck et al., 2020), i.e., plausible but non-factual information generated by LLMs.

The retrieval of documents relevant to a given query is a key task of the RAG system. Dense retrieval methods (Karpukhin et al., 2020; Gao and Callan, 2022) are widely used to capture semantic relationships between queries and documents, in which text encoders are trained to learn dense embeddings of queries and passages for their semantic matching. The encoders can be pre-trained in an unsupervised manner by using large-scale text pairs sampled from sentences and their contexts (Lee et al., 2019; Izacard et al., 2021), and then be fine-tuned on the annotated datasets for retrieval tasks (Wang et al., 2022; Chen et al., 2024). Meanwhile, recent LLMs have exhibited remarkable generalization abilities in many NLP tasks, including information retrieval. In this paper, we explore how the vast knowledge of LLMs can be effectively utilized in training retrievers. Recently, RePlug (Shi et al., 2024) has been proposed for distilling the LLMs' knowledge into small retrievers. RePlug calculates the relevance scores of k retrieved passages given a query, from which a likelihood over k passages is computed. The retriever is trained to minimize the KL divergence between this likelihood and the LLM's likelihood over passages based on its probability of predicting the ground truth answer. However, prediction probabilities are mostly unavailable as the output in the latest *black-box* LLMs (Achiam et al., 2023; Team et al., 2023). Thus, we consider the distillation of LLM's knowledge into retrievers when only the synthetically generated texts are available as the output from LLMs.

Contribution. We propose Syntriever, a framework to train/fine-tune retriever models based on synthetic data so as to distill the knowledge of

*Corresponding Author

black-box LLMs into retrievers effectively. We propose a two-stage framework: in the first stage, called *distillation stage*, we fine-tune the retriever with LLM-generated synthetic data; in the second stage, called *alignment stage*, we align the retriever with the preference of LLMs. In the distillation stage, Syntriever exploits synthetically augmented queries using chain-of-thoughts (Wei et al., 2022), synthetic positive and hard-negative passages, as well as self-verification to deal with hallucination. The retriever is then trained by modified Soft Nearest-Neighbor loss (Frosst et al., 2019) to cluster multiple relevant passages together in the embedding space. In the alignment stage, we continually fine-tune the retriever trained from the distillation stage, where the goal is to align the retriever with LLM preferences. The retriever fetches top-K passages from which a set of passage pairs is sampled and provided to LLMs for preference feedback. In particular, we propose a preference modeling called *partial Plackett-Luce ranking* to learn LLM preferences with a regularization effect such that the aligned model does not deviate excessively from the distilled model. We evaluated the performance of Syntriever in various domains of benchmark datasets for the retrieval tasks from BeIR (Thakur et al., 2021). Syntriever achieves superior performances on all benchmark datasets, by up to 18.6% in nDCG@10, compared to the prior state-of-the-art. Moreover, we show that the Syntriever framework can be combined with diverse base retrievers and LLMs, leading to a significant increase in retrieval accuracy.

2 Training Retrievers through Passage Synthesis

2.1 Problem Statement and Notation

Neural retrieval is a task of searching for top-K relevant passages $\mathcal{C}$ given query q using encoder E from knowledge source $\mathcal{Z}$:

$$\mathcal{C} = \text{Retrieval}(q, \mathcal{Z}, K, E) \tag{1}$$

The retrieval system (retriever in short) is used for retrieval-augmented generations (RAG) (Lewis et al., 2020; Guu et al., 2020). Our goal is to train/fine-tune a (pre-trained) text encoder E which outputs embeddings for semantic representations of queries and passages. The semantic similarity of query q and passage p is measured by

$$s_\tau(q, p) := \frac{\text{sim}(E(q), E(p))}{\tau}$$

where $\text{sim}(a, b)$ stands for the cosine similarity of vectors a and b, and τ is the temperature hyperparameter which controls the concentration of (normalized) embeddings on the unit hypersphere.

In the training dataset, each query q_i is paired with passage p_i manually labeled as relevant or the answer to q_i. We will denote a batch of samples during training by B, where B is a set of indices of batch samples. A typical method to train a retriever is metric learning with *contrastive loss* such as InfoNCE (Oord et al., 2018; Izacard et al., 2021):

$$\mathcal{L}_{\text{InfoNCE}} = -\log \left[\frac{\exp(s_\tau(q_i, p_i))}{\sum_{j \in B} \exp(s_\tau(q_i, p_j))} \right]$$

That is, manually labeled p_i is regarded as a *positive* passage for q_i, and the embeddings of q_i and p_i are pulled closer in the embedding space. The other passages in batch B are considered irrelevant to q_i, and as *negative* passages whose embeddings are pushed away from that of q_i.

Next, we outline the proposed method, dubbed *Syntriever*, which consists of two stages. In Stage 1 (Sec. 2.2), we use LLM-generated synthetic data to distill their parametric knowledge into the retriever. In Stage 2 (Sec. 2.3), we align the retriever with LLM preferences. The two-stage process of Syntriever is analogous to the training of LLMs, i.e., supervised fine-tuning (SFT) followed by alignment with human preferences (Ouyang et al., 2022). An overview of Syntriever is depicted in Fig. 1.

2.2 Stage-1. Distillation of LLM's knowledge through Synthesis

Given query q, our goal is to assimilate q to a set of positive documents, and to disassimilate q from negative documents. We synthesize a variety of positive and negative passages so as to distill the vast knowledge of LLMs into the retriever.

Decomposing query to easier sub-queries. Neural retrievers struggle with challenging queries (Li et al., 2024), e.g., if a query requires multi-step reasoning, or is too complex to understand. LLMs are capable of decomposing a complex query into multiple easier sub-queries which contain fine-grained planning to answer the query. We leverage the decomposition capability by applying the original query with prompts generating chain-of-thoughts (CoT) (Wei et al., 2022), e.g., "`Let's think step-by-step`" proposed by (Kojima et al., 2022). Specifically, given query q_i, we generate

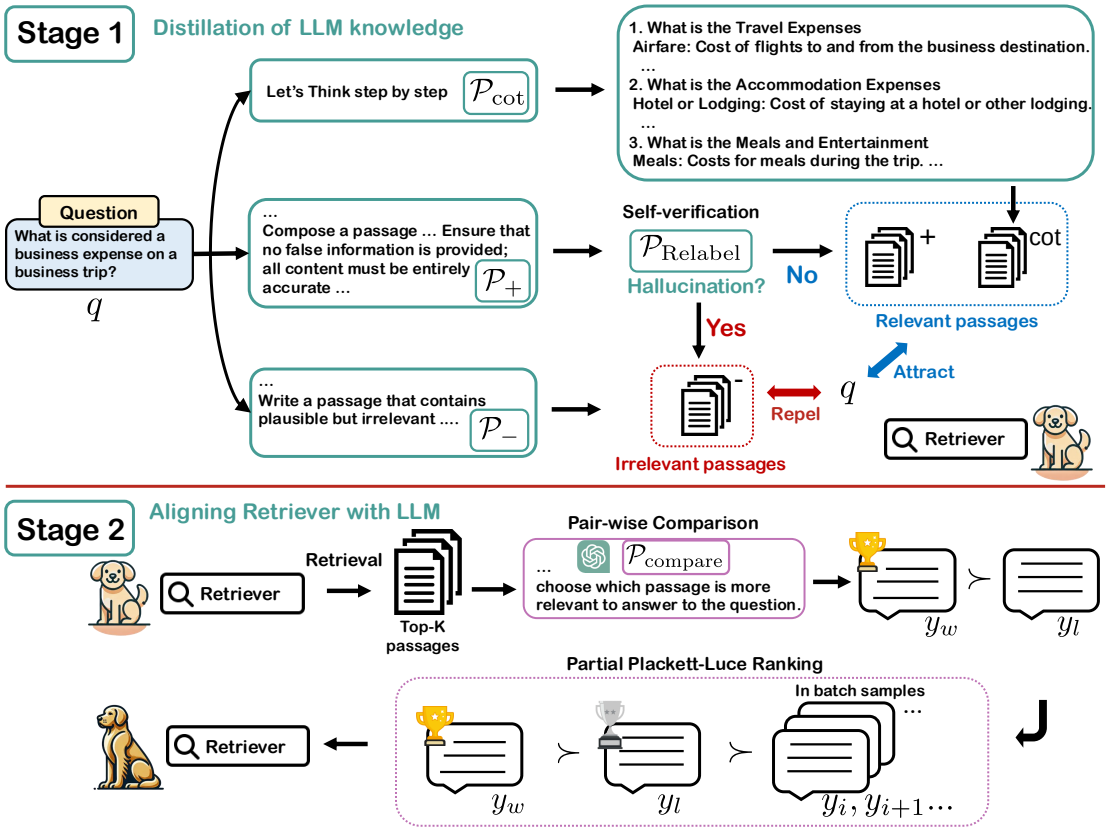

Figure 1: Overview of Syntriever. **Stage-1 (Distillation Stage).** Given a query, Syntriever uses LLMs to synthesize (i) related sub-queries (prompt $\mathcal{P}_{\text{cot}}$), (ii) relevant passages ($\mathcal{P}_{+}$) which are self-verified for hallucination ($\mathcal{P}_{\text{Relabel}}$), (iii) plausibly irrelevant passages ($\mathcal{P}_{-}$). The retriever is trained with the synthetic positive and negative passages. **Stage-2 (Alignment Stage).** The retriever is aligned with the LLM preferences. LLM compares passage pairs from top-K retrieved passages. If LLM prefers y_w over y_l, we write $y_w \succ y_l$. We propose *partial Plackett-Luce ranking* to combine preference modeling and contrastive learning for the retriever to learn $y_w \succ y_l \succ \{\text{in-batch negatives}\}$.

augmented query q_i^{cot} given by

$$q_i^{\text{cot}} = \mathcal{M}(\mathcal{P}_{\text{cot}}(q_i)) \tag{2}$$

where $\mathcal{M}(\cdot)$ denotes the LLM operation, and $\mathcal{P}_{\text{cot}}$ denotes the prompt operator to generate CoT (see Appendix C for prompt details). q_i^{cot} contains sub-queries relevant to the original query, which are carefully planned out with clarification and details necessary to retrieve relevant documents. We will use q_i^{cot} as a positive document for q_i. This helps the retriever with understanding diverse contexts associated with related queries in the future.

Synthesizing positive and hard-negative passages. We generate synthetic *positive* and *hard-negative* passages from query q. Although there exist positive passages manually labeled for q in the dataset, the synthesis of positive passages can distill a broader range of knowledgeable contexts from LLM to the retriever, and provide different perspectives on the query, which prevents overfitting to specific keywords or contexts. We generate synthetic positive passage p_i^+ related to query q_i with prompt $\mathcal{P}_{+}$:

$$p_i^+ = \mathcal{M}(\mathcal{P}_{+}(q_i)) \tag{3}$$

In addition, contrastive learning can be made more robust using *hard-negatives* (Robinson et al., 2021) where hard-negatives are samples that are difficult to distinguish from positive samples. For the retriever, hard negatives are plausible but irrelevant answers to query q. We synthesize hard-negative passage p_i^- with prompt $\mathcal{P}_{-}$ given by

$$p_i^- = \mathcal{M}(\mathcal{P}_{-}(q_i)) \tag{4}$$

Hallucination as Hard-negatives. We take a step to verify whether synthetic positive passages are indeed relevant to the given query. Using LLMs to generate answers runs a risk of *hallucinations*. Hallucination is a non-factual but seemingly plausible passage. The synthetic positive p_i^+ can potentially

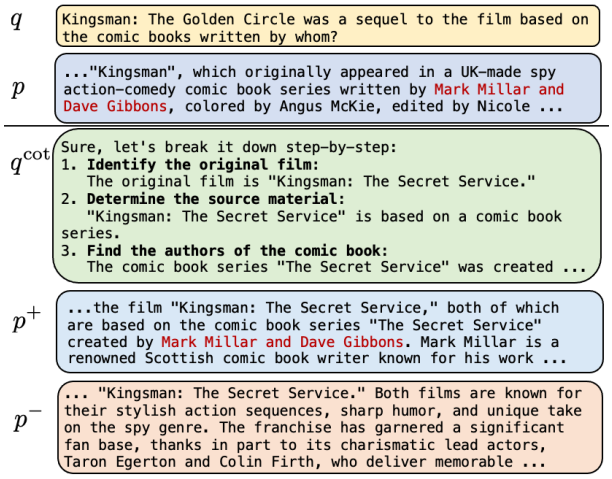

Figure 2: Example of LLM synthesis. The correct answer to the query is shown in red font.

be a hallucination. However, *the plausible irrelevance of hallucination fits the definition of hard-negatives.* Thus, we re-use hallucination as hard negative passages, which differs from prior works (Weng et al., 2023; Madaan et al., 2024) which simply discards hallucination outputs.

To that end, once positive passage p_i^+ is synthesized, LLM checks the passage for hallucination. LLMs are known to have self-verification ability (Weng et al., 2023), i.e., they can re-verify the inferred answer. If p_i^+ is decided as hallucination, we label p_i^+ as a hard-negative, which we call *Relabeling* step. Specifically,

$$\hat{p}_i = \mathcal{M}(\mathcal{P}_{\text{Relabel}}(q_i, p_i^+)) \tag{5}$$

where $\mathcal{P}_{\text{Relabel}}$ denote the prompting for relabeling. If p_i^+ is relabeled as a hard-negative, query q_i will have two hard-negatives (synthetic and relabeled) and two positives (manually labeled and CoT).

In summary, given query q_i, positive passages are manually labeled passage p_i, CoT q_i^{cot}, and synthetic positive p_i^+ (if not relabeled as negative). The negative passages are p_i^- (and relabeled passages, if any) and in-batch samples. An example of synthesized passages is shown in Fig. 2.

Putting positives together: modified Soft-Nearest Neighbor Loss. Next, we train the retriever with synthesized passages. Considering that there are multiple positives for a given query, we propose to use a loss inspired by soft-nearest neighbor (SNN) loss (Frosst et al., 2019). SNN loss is used in metric learning for supervised classification as follows. Consider batch B from a labeled dataset and a sample x_i in B with label y_i. The "nearest" neighbor (NN) to x_i is selected from B in a randomized fashion: the probability of x_j being selected as the NN is $\propto \exp(-d_T(x_i, x_j))$ where d_T is the distance metric with temperature parameter T. SNN loss is the negative logarithm of the probability that the NN of x_i is in the same class as x_i:

$$\mathcal{L}_{\text{SNN}}(x_i) = -\log\left(\frac{\sum_{j\in B: y_j = y_i} \exp(-d_T(x_i, x_j))}{\sum_{j\in B} \exp(-d_T(x_i, x_j))}\right)$$

The goal of loss $\mathcal{L}_{\text{SNN}}$ is *entanglement* (Frosst et al., 2019) which is to closely cluster the sample embeddings from the same class.

We consider a loss inspired by SNN loss. In our case, the set of points we want to cluster is a group of 4 samples (q_i, p_i, p_i^+, q_i^{cot}). Although these groups do not represent individual classes as in SNN loss, we still want the group to be "entangled". Thus, similar to SNN loss, we propose a loss $\mathcal{L}_{\text{distill}}(q_i)$ for query q_i given by

$$\mathcal{L}_{\text{distill}}(q_i) = -\log\Bigg(\frac{e^{s_\tau(q_i,p_i)} + e^{s_\tau(q_i,p_i^+)} + e^{s_\tau(q_i,q_i^{\text{cot}})}}{\sum\limits_{j\in B} e^{s_\tau(q_i,p_j)} + e^{s_\tau(q_i,p_j^+)} + e^{s_\tau(q_i,q_j^{\text{cot}})} + e^{s_\tau(q_i,p_j^-)}}\Bigg)$$

where the similarity metric (s_τ) is used instead of the negative distance ($-d_T$). Another difference between $\mathcal{L}_{\text{distill}}(q_i)$ and SNN loss is that there is no attraction term for synthetic hard-negatives (p_i^-) in $\mathcal{L}_{\text{distill}}(q_i)$, i.e., they are used only for repulsion from other samples.

2.3 Stage-2. Retriever Alignment from LLM Feedback

Alignment is a process of aligning language models with human preferences (Ouyang et al., 2022; Rafailov et al., 2024). Alignment provides LMs with a pair of answer candidates for a question, where the preference between the pair is labeled by humans. We propose to align the retriever with LLM preferences as follows. Given a query, the retriever trained in the distillation stage is asked to retrieve top-K passages. Next, a pair of passages is sampled from top-K passages, and LLM is asked to provide the preference between the pair. Since K passages are top passages from a retriever trained through the distillation stage, deciding the preference between the pair is likely to be challenging (for moderately small K, e.g., $K = 5$). The

retriever is continually trained based on the preference feedback from LLMs. The details of the alignment process are outlined as follows.

Step 1: Retrieve top-K passages. Given query q_i, we retrieve top-K passages using encoder $\hat{E}$ trained through the distillation stage:

$$\hat{\mathcal{C}}_i = \text{Retrieval}(q_i, \mathcal{Z}, K, \hat{E}) = \{c_{i,1}, c_{i,2}, ..., c_{i,K}\}$$

Step 2: Pair-wise Comparison. A pair of passages, $c_{i,j}$ and $c_{i,k}$, is sampled from $\hat{\mathcal{C}}_i$. We probe LLM to decide which passage is more relevant to answer query q_i using prompt $\mathcal{P}_{\text{Compare}}$:

$$(q_i, c_i^+, c_i^-) = \mathcal{M}(\mathcal{P}_{\text{Compare}}(q_i, c_{i,j}, c_{i,k})) \quad (6)$$

where LLM labels the more (resp. less) preferred passage as c_i^+ (resp. c_i^-). We compute the pairwise preferences of N distinct passage pairs sampled from $\hat{\mathcal{C}}_i$ where $N \leq \binom{K}{2}$ is a hyperparameter.

Step 3: Partial Plackett-Luce ranking. Consider batch B of triples (q_i, c_i^+, c_i^-) obtained in **Step 2** where q_i's in the batch are distinct. Encoder $\hat{E}$ is fine-tuned with the following loss function:

$$\mathcal{L}_{\text{align}}(q_i) = -\log \left[\frac{e^{s_\tau(q_i, c_i^+)}}{\sum_{j \in B} \left(e^{s_\tau(q_i, c_j^+)} + e^{s_\tau(q_i, c_j^-)} \right)} \times \frac{e^{s_\tau(q_i, c_i^-)}}{e^{s_\tau(q_i, c_i^-)} + \sum_{j \in B, j \neq i} \left(e^{s_\tau(q_i, c_j^+)} + e^{s_\tau(q_i, c_j^-)} \right)} \right] \quad (7)$$

We refer to the training under loss (7) as *partial Plackett-Luce ranking*. The method is explained in detail as follows.

From Bradely-Terry to Plackett-Luce model. Preference modeling has been used for aligning language models with human preferences (Ouyang et al., 2022; Rafailov et al., 2024). Bradley-Terry (BT) model (Bradley and Terry, 1952) is widely adopted for modeling preference over two choices. Consider a pair of answer passages y_w and y_l given query q. If y_w is preferred over y_l by a human annotator, the preference relation is denoted as $y_w \succ y_l \,|\, q$. In preference modeling, it is typically assumed that there exists some (implicit) reward function $r(q, y)$ for query q and answer y. Given query q and two answers y_1 and y_2, BT model is defined by the distribution

$$p(y_1 \succ y_2 \,|\, q) = \frac{e^{r(q,y_1)}}{e^{r(q,y_1)} + e^{r(q,y_2)}} \quad (8)$$

The fitting of BT model involves either explicitly formulating and optimizing reward $r(\cdot, \cdot)$ (Ziegler et al., 2019; Ouyang et al., 2022), or implicitly doing so by policy optimization through parameterization (Rafailov et al., 2024).

Plackett-Luce (PL) model (Plackett, 1975; Luce, 1959) generalizes BT model to ranking $M \geq 2$ choices. Suppose $\pi : [M] \to [M]$ is a permutation. Given query q and answers $y_1, ..., y_M$, we define the notation:

$$p(\pi \,|\, q) := p(y_{\pi(1)} \succ y_{\pi(2)} \succ ... \succ y_{\pi(M)} \,|\, q).$$

The PL model defines distribution $p(\pi \,|\, q)$ as

$$p(\pi \,|\, q) = \prod_{m=1}^{M} \left(\frac{e^{r(q, y_{\pi(m)})}}{\sum_{j=m}^{M} e^{r(q, y_{\pi(j)})}} \right) \quad (9)$$

where the m-th term in the product of (9) is the soft-max probability of the reward for the choice of rank m, $r(q, y_{\pi(m)})$, along with the rewards of choices of lower preferences.

Partial Ranking through Marginalization. The key idea of our method is to include in-batch samples in preference modeling. Consider triple (q_i, c_i^+, c_i^-) from batch B. Our goal is to model the following preference relation:

$$c_i^+ \succ c_i^- \succ \{\text{in-batch samples}\} \,|\, q_i \quad (10)$$

where the preference ordering of in-batch samples can be arbitrary or "don't care". Relation (10) is explained as follows. Firstly, c_i^+ is preferred over c_i^- by LLM given q_i. Secondly, since c_i^+ and c_i^- are in top-K passages obtained from a retriever trained through the distillation stage, it is highly likely that *both c_i^+ and c_i^- are preferred over irrelevant samples in the batch.* We call this relation *partial ranking*, since the ranking of the samples is incompletely specified.

The preference relation in (10) can be modeled by *marginalization* of Plackett-Luce distribution given by (9) as follows. Suppose we want to model the preference relation

$$y_{\pi(1)} \succ y_{\pi(2)} \succ \{y_{\pi(3)}, \ldots, y_{\pi(M)}\} \,|\, q \quad (11)$$

where the top-two choices ($\pi(1)$ and $\pi(2)$) are preferred over the rest ($\pi(3), \ldots, \pi(M)$), and the ordering of the rest can be arbitrary. Since $p(\pi|q)$ is a distribution over π, the distribution modeling (11) can be obtained by marginalizing $p(\pi|q)$ over the

components of π except top-two choices, $\pi(1)$ and $\pi(2)$. Specifically, we have that

$$\sum_{\pi(3),\pi(4),\ldots,\pi(M)} p(\pi \mid q) = \frac{e^{r(q,y_{\pi(1)})}}{\sum_{j=1}^{M} e^{r(q,y_j)}} \times \frac{e^{r(q,y_{\pi(2)})}}{e^{r(q,y_{\pi(2)})} + \sum_{j \neq \pi(1),\pi(2)} e^{r(q,y_j)}} \quad (12)$$

The derivation of (12) is provided in Appendix B. Thus, if we set $q = q_i$, $y_{\pi(1)} = c_i^+$, $y_{\pi(2)} = c_i^-$ and the rest of y's as in-batch samples with $M = |B|$ and the reward $r(\cdot,\cdot)$ as the similarity metric $s_\tau(\cdot,\cdot)$, then (12) models the partial relation (10).

In conclusion, the proposed loss (7) is the negative log-likelihood of the marginalized PL model representing partial ranking given by (10), which makes our training a maximum likelihood estimation under distribution (12). The key question is: *why should we include in-batch samples in the preference modeling?*

Combining Preference Modeling and Contrastive Learning. Our observation is that, the training objective for preference modeling invariably takes the form of a *contrastive loss*. For example, the BT model is trained with the loss which is the negative log of (8). Suppose we use the BT model, in which case y_1 and y_2 in (8) are replaced by c_i^+ and c_i^- respectively. From a contrastive learning perspective, (8) attracts c_i^+ (y_1) to q, but repels c_i^- (y_2) from q. But this may unintentionally move c_i^+ and c_i^- closer to samples irrelevant to q_i. This is undesirable, because c_i^+ and c_i^- are among top-K documents retrieved by the model trained through the distillation stage, and thus should be regarded as relatively "positive" and kept away from irrelevant in-batch negatives. Conventional preference modeling, such as BT model, lacks perspective on learning with negative (irrelevant) samples.

The proposed loss directly addresses the problem: it not only captures the LLM's preferences but also maintains separation among irrelevant documents. Thus, *our loss combines preference modeling and contrastive learning*. It can be seen that (7) is simply a sum of two contrastive-type losses. By having a similar form of contrastive loss as that from the distillation stage, e.g., positive embeddings keeping distances from in-batch negatives, our alignment loss serves as *regularization*. That is, the model is prevented from excessively deviating from that trained in the distillation stage. Regularization is deemed important in the alignment of LLMs as well (Ouyang et al., 2022). In addition, it is reported that the larger number of negatives leads to better performance in contrastive learning (He et al., 2020). In the Experiments section, we show that partial PL ranking model achieves robust performances across datasets, whereas BT model occasionally suffers from poor alignment.

3 Experiment

3.1 Experimental Settings

Datasets. Experiments are conducted on retrieval benchmark datasets from various domains in BeIR (Thakur et al., 2021). We evaluate the performance of retrievers in two benchmark settings as follows.

- **Supervised Fine-Tuning.** The models are evaluated on BeIR benchmark datasets which contain the training datasets. For each benchmark dataset, every model is fine-tuned on its training dataset, and we report in-domain evaluation results on that benchmark dataset.

- **Zero-shot Transfer.** The models are evaluated on out-of-domain datasets from BeIR benckmark. The zero-shot setting is similar to previous work (Izacard et al., 2021; Wang et al., 2022): the models can be first fine-tuned on large retrieval datasets such as MSMARCO (Nguyen et al., 2016) and NQ (Kwiatkowski et al., 2019) for generic knowledge, and then are evaluated on unseen datasets.

We use Normalised Discounted Cumulative Gain (nDCG@K) as the default performance metric.

Baselines. We experiment with lexical retriever BM-25 (Robertson et al., 2009), semantic retrievers DPR (Karpukhin et al., 2020), SBERT (Reimers and Gurevych, 2019), CoCondenser (Gao and Callan, 2022), RocketQA (Ren et al., 2021), Contriever (Izacard et al., 2021), E5 (Wang et al., 2022), English model of BGE-M3-EN (Chen et al., 2024), Nomic-embed (Nussbaum et al., 2024).

Settings of Syntriever. Syntriever uses pre-trained E5 (Wang et al., 2022) as the base encoder E. In the settings of supervised fine-tuning, Syntriever is trained with synthetic data generated from each training dataset. In the settings of the zero-shot transfer, Syntriever is first trained on synthetic data based on training datasets of MSMARCO and NQ, and then is evaluated on out-of-domain datasets

Method	BM-25	DPR	CoCondenser	RocketQA	Contriever	E5	BGE-M3-EN	Nomic-Embed	Syntriever
MSMARCO	22.8	36.8	37.4	30.2	40.3	**42.4**	41.3	37.1	**50.3**
HotpotQA	60.1	48.9	56.3	53.3	61.2	63.4	64.2	**66.3**	**70.2**
FiQA	23.5	21.4	27.6	30.2	31.4	37.9	39.1	**40.9**	**41.9**
Fever	**75.3**	50.7	51.8	52.3	53.7	57.1	54.3	53.5	**60.3**
SciFact	66.5	63.3	48.7	56.8	64.9	73.7	75.4	**79.1**	**80.5**
NFCorpus	32.5	35.4	32.5	29.3	31.7	35.8	**38.2**	36.8	**43.3**
NQ	32.9	41.2	43.3	42.1	42.5	49.3	46.3	**50.1**	**52.1**

Table 1: Supervised fine-tuning results on seven BeIR benchmark with training datasets (nDCG@10). The best scores are highlighted in **bold with underline** and, the second best scores are emphasized in **bold**.

Method	BM-25	DPR	CoCondenser	RocketQA	Contriever	BGE-M3-EN	E5	Nomic-Embed	Syntriever
MSMARCO	22.8	17.7	24.3	23.2	40.7	35.2	**43.1**	26.4	**50.1**
Trec-covid	65.6	33.2	**71.2**	67.5	59.6	44.6	61.7	67.1	**75.3**
HotpotQA	60.3	39.1	56.3	35.6	63.8	**68.3**	52.4	**69.1**	60.2
FIQA	23.6	11.2	27.6	30.2	32.9	28.3	**37.9**	37.8	**39.5**
Arguana	31.5	17.5	29.9	45.1	44.6	**61.5**	51.4	**54.2**	38.8
Touche-2020	**36.7**	13.1	19.1	24.7	23.0	13.5	**28.3**	19.0	19.9
Quora	78.9	24.8	30.5	31.2	86.5	**88.7**	87.9	88.4	**88.9**
CQADupstack	29.9	15.3	17.2	19.3	34.5	40.2	28.3	**49.6**	**41.4**
DBPedia	31.3	26.3	36.3	35.6	41.3	19.0	33.8	**39.4**	**39.8**
Climate-Fever	21.3	14.8	14.4	18.0	**23.7**	18.3	15.4	**27.0**	13.1
SciDocs	15.8	7.7	13.7	13.1	16.5	9.6	19.0	**19.2**	**19.7**
SciFact	66.5	31.8	61.5	56.8	69.3	71.5	**73.1**	**71.8**	64.2
NFCorpus	32.5	18.9	32.5	29.3	32.8	32.7	35.1	**35.5**	**36.6**
Fever	**75.3**	56.2	49.5	67.6	**75.8**	64.3	58.2	60.3	60.2
NQ	32.9	47.4	48.7	59.5	49.8	29.8	**60.0**	51.2	**62.2**

Table 2: Zero-shot transfer results on BeIR benchmark datasets (nDCG@10). The best scores are highlighted in **bold with underline**, and the second best scores are emphasized in **bold**.

q^{cot}	$p^{+,-}$	$c^+ \succ c^-$	MSMARCO	HotpotQA	FiQA	SciFact	NFCorpus
✗	✗	✗	42.4	63.4	37.9	73.7	35.8
✔	✗	✗	44.6	64.3	38.5	76.7	40.2
✗	✔	✗	45.7	64.7	39.3	77.3	40.8
✔	✔	✗	46.2	65.3	40.2	78.9	41.4
✗	✗	✔	45.8	67.3	39.1	75.5	37.3
✔	✔	✔	**50.3**	**70.2**	**41.9**	**80.5**	**43.3**

Table 3: Ablation study (in nDCG@10). The first three columns represent the following components: synthesized query with CoT (q^{cot}), synthetic positives and hard-negatives ($p^{+,-}$), alignment ($c^+ \succ c^-$).

from BeIR benchmarks. This is a similar setting as Contriever (Izacard et al., 2021), E5 (Wang et al., 2022), etc. To minimize the effects of model sizes on performance, we set the size of Syntriever and all baseline models to (approximately) 125M. In alignment stage of Syntriever, we set $K = 5$ by default, and set $N = \binom{K}{2} = 10$. Detailed hyperparameters are in Appendix D.

4 Experimental Results

4.1 Main Results

We first present the supervised fine-tuning results on seven datasets for the retrieval task which are BeIR benchmarks with training datasets. The results are shown in Table 1. Compared to the second-best models, Syntriever improves the retrieval performances by: 18.6% on MSMARCO, 5.9% on HotpotQA, 2.5% on FiQA, 1.8% on SciFact, 8.3% on NFCorpus, and 4% on NQ. The base encoder for Syntriever is a pre-trained E5; still, Syntriever achieves performance gain over E5 by: 18.6% on MSMARCO, 10.7% on HotpotQA, 10.6% on FiQA, 9.2% on SciFact, 20.9% on NFCorpus, 5.6% on Fever and 5.7% on NQ. This shows that Syntriever can successfully distill LLMs' capability into small retrievers and improve their performance by a large margin. Overall, Syntriever shows robust performances on datasets both in generalized and specialized domains. Our results show that small LMs can efficiently learn from the teacher model through synthetic data and can be successfully aligned through feedback, even without access to the output probability of black-box LLMs.

Next, we present zero-shot transfer results. Table 2 shows that Syntriever achieves the best performances on 8, and the second best on 1, out of 15 datasets. Note that the performances of Syntriever on MSMARCO and NQ are in-domain results, whereas other baselines, e.g., Contriever (Izacard et al., 2021), E5 (Wang et al., 2022), etc., are re-

Dataset	Metric	Base encoder		
		ColBERT	SBERT	Contriever
HotpotQA	nDCG@1	65.3 (**+10.3**)	63.2 (**+10.3**)	73.3 (**+10.5**)
	nDCG@3	58.2 (**+9.7**)	57.0 (**+9.5**)	68.2 (**+9.5**)
	nDCG@5	60.7 (**+8.5**)	59.5 (**+9.8**)	70.4 (**+9.7**)
	nDCG@10	62.8 (**+9.1**)	61.8 (**+11.1**)	72.1 (**+10.9**)
FiQA	nDCG@1	30.1 (**+3.8**)	26.7 (**+5.2**)	32.1 (**+5.1**)
	nDCG@3	27.8 (**+3.2**)	25.1 (**+4.6**)	30.4 (**+4.7**)
	nDCG@5	30.5 (**+2.8**)	26.1 (**+4.5**)	31.9 (**+4.3**)
	nDCG@10	33.5 (**+2.9**)	25.8 (**+4.3**)	35.2 (**+4.6**)

Table 4: Performance gains of Syntriever with different base encoders.

ported also to be trained on MSMARCO and/or NQ. In particular, although Syntriever shares the same base model as E5, it improves the retrieval accuracy on 11 datasets. This is perhaps because LLM-generated synthetic data and alignment feedback improve the generalization capabilities of the retriever on unseen data.

4.2 Ablation study

We conduct an ablation study on Syntriever. We add or remove model components, and the effects on the performance are shown in Table 3. The results show that both synthesized query (q^{cot}) and passages (p^+, p^-) in the distillation stage improve the retrieval performances. Overall, the distillation stage achieves an average gain of 8.2% over the base retriever. Results show that the retriever successfully learns from the parametric knowledge of LLMs during the distillation stage. Also, the alignment component ($c^+ \succ c^-$) in Table 3 is shown to achieve performance gains of up to 8.8%. Our results show that the alignment component is significant for retrieval performance, considering that nDCG@K is sensitive to the fine-grained ranking of relevant passages.

4.3 Performances with different encoders

Syntriever is a framework for training encoders for retrieval, and thus can be combined with different sentence encoders. We experiment with various well-known encoders, e.g., ColBERT, SBERT, and Contriever, as the base encoders for Syntriever. Syntriever improves the performance by a large margin in all three retrieval models. The performance improvement is particularly high in nDCG@1 which concerns retrieving the exact passage relevant to the query. This is because the alignment stage in Syntriever helps the retriever with a fine-grained ranking of highly relevant passages. Overall, the results show that Syntriever is generally applicable to, and improves the performances of, various retrievers.

Method	HotpotQA	FiQA	SciFact
w/o Self-verification	67.4	40.8	79.7
w/ Self-verification	**70.2**	**41.9**	**80.5**

Table 5: Effectiveness of re-labeling hallucination passages. Results are in nDCG@10.

Dataset	Metric	GPT-4o mini	GPT-4o
SciFact	nDCG@1	65.7	**66.7**
	nDCG@3	73.0	**75.0**
	nDCG@5	76.7	**79.1**
	nDCG@10	76.3	**78.9**
NFCorpus	nDCG@1	**46.0**	45.0
	nDCG@3	**42.3**	41.2
	nDCG@5	**42.1**	41.1
	nDCG@10	**42.6**	41.4

Table 6: Comparison of models trained only up to **the distillation stage** using synthetic data from GPT-4o mini vs. GPT-4o.

Dataset	Metric	GPT-4o mini	GPT-4o
HotpotQA	nDCG@1	64.5	**68.3**
	nDCG@3	63.5	**71.2**
	nDCG@5	66.3	**71.4**
	nDCG@10	68.6	**70.2**
FiQA	nDCG@1	41.8	**42.1**
	nDCG@3	39.7	**40.4**
	nDCG@5	41.3	**41.5**
	nDCG@10	40.7	**41.9**

Table 7: Comparion of models trained by preference feedback from GPT-4o mini vs. GPT-4o. Both models are trained with GPT-4o in the distillation stage.

4.4 Effects of Re-labeling Hallucination Passages

Table 5 shows that LLM self-verification and re-labeling are effective for the synthetic training by Syntriever. The performance improvement of self-verification on HotpotQA is relatively greater than other datasets. We found that, approximately 15% of synthetic positive passages were relabeled as hallucinations in the case of HotpotQA, whereas the proportion was about a few percent in other datasets. This indicates that the performance improvement through relabeling is likely higher for HotpotQA. In conclusion, removing hallucinations (and even *re-using* them as hard-negatives as in Syntriever) through self-verification is important for data synthesis, which is the case for most tasks utilizing LLMs (Weng et al., 2023).

4.5 Weaker but Cheaper LLMs can be effective

We examine how the LLM capabilities affect distillation and alignment performances. We consider

Method	FiQA	NFCorpus	SciFact
Partial Plackett Luce	**41.9**	**43.3**	**80.5**
Bradley Terry	35.7	36.1	79.8

Table 8: Comparison of preference modeling. Results are in nDCG@10.

two LLMs: GPT-4o vs. GPT-4o-mini, where GPT-4o is the larger and more capable model. First, we compare the distillation capabilities of two LLMs. Table 6 shows the comparison, where Syntriever is trained only up to the distillation stage. Interestingly, the distillation performance of GPT-4o-mini is better than GPT-4o on NFCorpus. Considering the datasets concern different knowledge domains (SciFact: scientific, NFCorpus: medical), smaller models may be better at teaching than larger ones in certain domains. Our results interestingly coincide with recent findings that weaker models may be better at teaching than stronger models in domains like math problem solving (Bansal et al., 2024). Next, we examine the alignment capabilities of LLMs. For a fair comparison, two models are first trained by GPT-4o in the distillation stage, and then trained by different LLMs in the alignment stage. Table 7 shows that the larger model (GPT-4o) is better at alignment. It is challenging to rank top-K passages retrieved by a distilled retriever, requiring a deep understanding of various contexts, and thus larger models may be more favored for the task. Overall, smaller models appear to be quite competitive, i.e., the performance gap is small or even better in some domains. Thus, our prospect is that distillation/alignment through small models will become an increasingly good alternative, especially under a fixed compute budget (Bansal et al., 2024).

4.6 Comparison of Preference Modeling Methods

Table 8 compares the preference modeling methods for alignment: Bradley-Terry (BT) (Bradley and Terry, 1952) and partial Plackett-Luce (PL) ranking model. While BT and partial PL models achieve similar performances on SciFact, BT model shows poor performances on FiQA and NFCorpus. The following is a possible explanation. The search results on SciFact tend to be highly accurate, and most of top-K passages are likely to contain (partly) relevant context. By contrast, top-K passages on FiQA and NFCorpus which are more challenging datasets, will tend to be only marginally relevant to the given query. The partial PL performs preference ranking while keeping those marginally relevant passages away from highly irrelevant (in-batch) passages. Without such regularization of keeping marginally positive samples away from in-batch negatives, which was done during the distillation stage, BT model may cause the retriever to *forget* the knowledge learned during the distillation stage. This may cause performance drops on FiQA and NFCorpus as shown in Table 8. Thus, we conclude that the proposed partial ranking is crucial for the alignment performance.

Dataset	Metric	w/o Alignment	$K=3$	$K=5$	$K=10$
FiQA	nDCG@1	38.7	40.9	42.1	**42.3**
	nDCG@3	38.3	39.7	40.4	**41.5**
	nDCG@5	38.5	40.9	41.8	**42.7**
	nDCG@10	37.9	41.7	41.9	**42.5**
SciFact	nDCG@1	80.2	81.0	**81.3**	80.0
	nDCG@3	78.0	78.6	79.0	**79.8**
	nDCG@5	78.8	79.7	**79.8**	79.2
	nDCG@10	78.9	80.0	80.5	**80.6**

Table 9: Effect of varying K in top-K retrieved passages for preference alignment.

Dataset	Default	$K=3$	$K=5$	$K=10$
FiQA	41.9	41.7	41.8	**42.3**
SciFact	**80.5**	80.0	80.3	79.8
NFCorpus	43.3	42.4	42.6	**43.8**

Table 10: Effect of varying K in top-K retrieved passages and the number N of sampled pairs for comparison in alignment. We set $N = K$ for this experiment. By default, Syntriever uses $K = 5$ and $N = \frac{K(K-1)}{2} = 10$. The evaluation metric is nDCG@10.

4.7 Effects of the number of retrieved passages during alignment

We examine the effect of the number K in the top-K passage retrieved during the alignment process. Table 9 shows the results with varying K, where we sample all the possible pairs, or $N = \binom{K}{2}$, for comparison. The performance improves with increasing K, up to 12% in FiQA and 5.1% in SciFact. Also, results show that the larger K, the better the performance. In addition, using a larger number of passages is particularly effective when the overall retriever accuracy is low, since it is more likely to retrieve relevant context in top-K-ranked passages for large K. However, large K may incur high computational costs if $N = \binom{K}{2}$, and thus there is a trade-off between performance and computational overheads. In this paper, we chose $K = 5$ as a good trade-off point.

In addition, we experiment with the numbers of

Passages	Can derive answer	Cannot derive answer
Synthetic positive	88%	12%
Ground truth	84%	16%

Table 11: Results of GPT-4o about whether each passage can answer ground truth. We randomly select 1000 samples in each passage set.

passage pairs to be sampled for comparison (N) with varying K. Previously in Table 9, we set $N = \binom{K}{2} = K(K-1)/2$. Here we provide the experiments with a smaller N given by $N = K$. The results are shown in Table 10. Overall, if we compare Table 9 and 10, the performance seems to slightly degrade for smaller N. As previously, for challenging datasets such as FiQA and NFCorpus, the performance seems to gradually improve with increasing K, again because more retrieved passages lead to a higher chance of including relevant passages in top-K. At the same time, increasing K seems to exhibit diminishing returns on the performance. Overall, the default setting of Syntriever ($K = 5, N = 10$) appears to be a reasonable choice in terms of a balance between complexity and performance.

4.8 Quality of Synthetic Positives

In general, it is difficult to accurately quantify the ratio of hallucination in the synthetic passage. The passage may not have direct clues to the answers, but may contain partial information from which the answer can be deduced. How relevant a passage should be to the query so that the passage is classified as positive? This is very hard to quantify, and thus measuring the quality of synthetic passages is difficult as well.

We performed experiments to indirectly measure the quality of synthetic positives as follows. We asked GPT-4o that whether the true answer can be directly derived from synthetic positive passages (after self-verification). We asked the same question, but in this time whether the answer can be derived from the ground-truth passages provided by the dataset. The results are shown in the table below.

Interestingly, GPT-4o states that only 84% of the ground truth passages have direct clues to the true answer. This is because, a significant portion of the "ground truth" passages of the HotpotQA dataset do not contain direct clues to the true answer, but only indirect clues or partial information. By contrast, GPT-4o stated that 88% of synthetic positives after self-verification contain direct contexts to the true answer. Thus, we conclude that synthetic positives after self-verification are of fairly high quality.

5 Related Work

Neural Information Retrieval. Neural information retrieval is a key element of retrieval-augmented generation (RAG) (Lewis et al., 2020) which is a retrieve-and-read approach for open domain question answering tasks (Chen et al., 2017). Lexical retrieval methods such as BM-25 (Robertson et al., 2009) have been mostly used prior to neural retrievals, which however had difficulties with capturing semantic information at scale. Thus, dense passage retrievers using text encoders (Devlin, 2018) have been actively explored (Karpukhin et al., 2020; Gao and Callan, 2022; Xiong et al., 2021). RocketQA (Qu et al., 2021) is a multi-step training framework for a retrieval system consisting of a retriever and a re-ranker which typically is a cross-encoder to estimate the ranking among retrieved passages. RocketQA further utilizes the re-ranker to sample hard negatives from top-retrieved passages. Meanwhile, Syntriever does not use separate re-rankers, but continually trains the retriever for its alignment with the ranking preference of LLMs. Unsupervised learning for retrieval (Izacard et al., 2021; Wang et al., 2022) was proposed to train sentence encoders by contrastive learning using a large collection of text-pair datasets. Subsequently, a hybrid retrieval method which combines lexical, dense, and multi-vector retrievers has been proposed (Chen et al., 2024). RePlug (Shi et al., 2024) proposed a knowledge distillation for retrievers using KL divergence associated with the prediction probabilities of relevant documents from LLMs which, however, are available only from outdated APIs.

Training with Synthetic Data. Tiny-stories (Eldan and Li, 2023) first proposed training small language models using synthetic data generated by GPT-4 (Achiam et al., 2023). Motivated by (Eldan and Li, 2023), Phi (Gunasekar et al., 2023) proposed filtering of code data based on the *educational value* through the prompting of GPT-4. The next version of Phi-series (Li et al., 2023; Abdin et al., 2024) generated high-quality synthetic data from judiciously selected topics in order to distill GPT-4's knowledge into small LLMs. They demonstrated that distillation through synthetic data of

high educational value can boost the performances of small LLMs. (Wang et al., 2023) proposed to train a Mistral-7B model (Jiang et al., 2023) by synthetically generating query-document pairs by prompting GPT-4 for various text embedding tasks. (Yu et al., 2024) proposed distillation synthetic data where they fine-tune the student LLM using the output answers from teacher models based on rationales (Wei et al., 2022; Deng et al., 2023). The aforementioned methods have demonstrated that student models can efficiently learn from the synthetic data generated by teacher models.

6 Conclusion

We proposed Syntriever, a training framework for retrieval systems using LLM synthesis. In the distillation stage, Syntriever synthesizes various types of passages including augmented queries, relevant and plausibly irrelevant passages. Relevant passages are clustered in the embedding space using modified soft nearest-neighbor loss. In the alignment stage, the retriever is continually trained based on the preference feedback of LLMs on the retrieved passages. We propose a preference modeling called partial Plackett-Luce ranking to learn LLM preferences while maintaining the similarity relation among embeddings learned during the distillation stage. Experiments show that Syntriever achieves significant performance gains over baselines on benchmark datasets from various domains.

7 Limitations

Although Syntriever achieves performance gains compared to baseline retrievers on various benchmark datasets, it requires LLM inferences to generate synthetic data and alignment feedback. This may incur additional costs compared to other methods which only perform a fine-tuning of text encoders. However, the cost of proprietary black-box LLMs has become increasingly cheaper and affordable. Moreover, weaker but cheaper LLMs become increasingly capable of teaching student models (Bansal et al., 2024). Thus, we believe that the Syntriever framework is widely applicable to retrieval systems in practice.

8 Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2022-NR070834), and by the Institute of Information & Communications Technology Planning & Evaluation (IITP)-ICT Creative Consilience Program grant funded by the Korea government (MSIT) (IITP-2025-RS-2020-II201819).

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q Tran, and Mehran Kazemi. 2024. Smaller, weaker, yet better: Training llm reasoners via compute-optimal sampling. *arXiv preprint arXiv:2408.16737*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

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

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*.

Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. 2019. Analyzing and improving representations with the soft nearest neighbor loss. In *International conference on machine learning*, pages 2012–2020. PMLR.

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.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. *arXiv preprint arXiv:2407.16833*.

R Duncan Luce. 1959. *Individual choice behavior*, volume 4. Wiley New York.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Robin L Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2021. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. 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 8364–8377.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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*.

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, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

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

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text degeneration with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Reproducibility Statement

Source Code. We release the source code in the public repository.
Black-box LLMs. We experiment with `GPT-4o` and `GPT-4o-mini` for synthetic data generation. Those models are accessible by OpenAI API. We generate LLMs' responses using our prompt templates in Appendix C.
Training Implementation. We implement retrieval tasks based on BEIR (Thakur et al., 2021) which is implemented by sentence-transformers. All the sentence encoders used in our experiments are publicly accessible in HuggingFace.
Evaluation Datasets. The four evaluation datasets of HotpotQA, FiQA, SciFact, and NFCorpus are released to the public repository. All the experiments are conducted with the single A100 GPU with 80GB VRAM.

B Derivation of (12)

For notational simplicity, we define $z_k := \exp(r(q, y_{\pi(k)}))$. We have that

$$\sum_{\pi(3),\ldots,\pi(M)} p(\pi \mid q) = \sum_{\pi(3),\ldots,\pi(M)} \prod_{m=1}^{M} \left(\frac{z_m}{\sum_{j=m}^{M} z_j} \right) \tag{13}$$

$$= \sum_{\pi(3),\ldots,\pi(M)} \underbrace{\frac{z_1}{\sum_{j=1}^{M} z_j}}_{(a)} \underbrace{\frac{z_2}{\sum_{j=2}^{M} z_j}}_{(b)} \prod_{m=3}^{M} \left(\frac{z_m}{\sum_{j=m}^{M} z_j} \right) \tag{14}$$

$$= \underbrace{\frac{z_1}{\sum_{j=1}^{M} z_j}}_{(a)} \underbrace{\frac{z_2}{\sum_{j=2}^{M} z_j}}_{(b)} \underbrace{\sum_{\pi(3),\ldots,\pi(M)} \prod_{m=3}^{M} \left(\frac{z_m}{\sum_{j=m}^{M} z_j} \right)}_{(c)} \tag{15}$$

$$= \underbrace{\frac{z_1}{\sum_{j=1}^{M} z_j}}_{(a)} \underbrace{\frac{z_2}{z_2 + \sum_{j \neq 1,2} z_j}}_{(b)} \tag{16}$$

The derivation steps are explained as follows.

- (13) is the definition of $p(\pi|q)$ from Placket-Luce ranking model.
- In (14), we take two terms (a) and (b) out from the product in (13).
- In (15), (a) and (b) are invariant with respect to $\pi(3), \ldots, \pi(M)$, so they can be taken out of the summation with respect to $\pi(3), \ldots, \pi(M)$. Specifically, the denominators of (a) and (b) contains the sums over $\pi(3), \ldots, \pi(M)$, and the sum is a permutation-invariant operation. Also, (c) is the sum of Placket-Luce distribution over all possible permutations of $\pi(3), \ldots, \pi(M)$. Thus, (c) must sum up to 1.
- In (16), we simply re-write (b) in (15) as shown in (b) in (16).

By replacing z_i with $\exp(r(q, y_{\pi(i)}))$ in (16), we obtain the marginalization result in Eq. (12).

C Prompt templates

C.1 Prompt template of positive passage generation ($\mathcal{P}_+$).

> You are a subject matter expert in your field with substantial accumulated knowledge in a specific subject or topic, validated by academic degrees, certifications, and/or years of professional experience in that field.
>
> Question: {question}
>
> Write a passage that elaborates on the question. Ensure that no false information is provided; all content must be entirely accurate. Present everything you are aware of, offering a comprehensive and detailed explanation. Do not include any unverified or speculative information.

Figure 3: Prompt template design for generating synthetic positive passages.

C.2 Prompts for generating plausible but irrelevant passages ($\mathcal{P}_-$).

> You are a subject matter expert in your field with substantial accumulated knowledge in a specific subject or topic, validated by academic degrees, certifications, and/or years of professional experience in that field.
>
> Question: {question}
>
> Write a passage that contains plausible but irrelevant context given the question.

Figure 4: Prompt template design for generating plausible but irrelevant passages.

C.3 Prompt template of relabeling for synthetic passages ($\mathcal{P}_{\textbf{Relabel}}$).

> You are a subject matter expert in your field with substantial accumulated knowledge in a specific subject or topic, validated by academic degrees, certifications, and/or years of professional experience in that field.
>
> Question: {question}
> Passage: {passage}
> Is the above passage relevant to the aforementioned question?
> Answer with yes or no.

Figure 5: Prompt template design of relabeling for synthetic positive passages.

C.4 Prompts for pair-wise comparison of two passages ($\mathcal{P}_{\text{Compare}}$).

You are a subject matter expert in your field with substantial accumulated knowledge in a specific subject or topic, validated by academic degrees, certifications, and/or years of professional experience in that field.

Passage #1: {passage1}
Passage #2: {passage2}
Question: {question}

Based on your professional knowledge, choose which passage is more relevant to answer the given question.
Only answer as Passage #1 or Passage #2

Figure 6: Prompt template design for comparison of a passage pair.

D Hyperparameters

Hyperparameter	Value
Model size	125M
Learning rate	2e-5
Learning rate scheduler	Cosine
Optimizer	Adam (Kingma, 2014)
Warmup ratio	1000 steps
Weight decay	0.01
GPU Usage	Single A100 w/ 80GB VRAM
Batch size	60 (Stage 1), 100 (Stage 2)
τ (temperature)	0.05

Table 12: Detailed hyperparameters.

E Dataset Statistics

Dataset	Train	Validation	Test	Domain
MSMARCO	532,752	9,261	7,438	Search Engine
HotpotQA	170,001	10,895	14,811	Wikipedia
FiQA	14,167	1,239	1,707	Finance
SciFact	920	-	340	Science
NFCorpus	110,576	11,386	12,335	Medical
FEVER	140,086	8,080	7,938	Fact Verification
NQ	132,803	-	3,452	Search Engine

Table 13: Dataset statistics.