Published: 02 March 2026
Online AM: 19 January 2026
Accepted: 26 December 2025
Revised: 16 September 2025
Received: 22 April 2025

RESEARCH-ARTICLE

DemoRank: Selecting Effective Demonstrations for Large Language Models in Ranking Task

WENHAN LIU, Gaoling School of Artificial Intelligence, Beijing, China

YUTAO ZHU, Gaoling School of Artificial Intelligence, Beijing, China

ZHICHENG DOU, Gaoling School of Artificial Intelligence, Beijing, China

YUJIA ZHOU, Tsinghua University, Beijing, China

Open Access Support provided by:

Gaoling School of Artificial Intelligence

Tsinghua University

DemoRank: Selecting Effective Demonstrations for Large Language Models in Ranking Task

WENHAN LIU, YUTAO ZHU, and ZHICHENG DOU, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
YUJIA ZHOU, Tsinghua University, Beijing, China

Large Language Models (LLMs) have been proven to have strong zero-shot passage ranking capabilities. In-context learning effectively enhances LLM performance by providing few-shot demonstrations, opening avenues for further improving LLM's ranking ability. However, existing studies usually retrieve the most similar demonstrations to the input, ignoring the demonstration dependencies and diversity, which is insufficient to inspire the LLM for assessing the current query-passage relevance. In this article, we propose a framework named DemoRank, which selects few-shot demonstrations by performing a novel dependency-aware reranking of the retrieved demonstrations. Considering the dependency, combining top-ranked demonstrations yields better results. Nevertheless, generating the training samples for such a dependency-aware demonstration reranker faces two challenges: (1) the traditional demonstration ranked list assumes demonstration independence, which cannot be used to train our reranker, and (2) obtaining the optimal demonstration ranked list from the retrieved set is NP-hard and inefficient. To overcome these challenges, we propose an approach to construct a kind of dependency-aware training samples efficiently and design a list-pairwise training approach for the optimization of the demonstration reranker. We conduct extensive experiments on a series of passage ranking datasets, and the results demonstrate the superior performance of our proposed DemoRank framework under various scenarios. Our code is publicly available at https://github.com/8421bcd/demorank.

CCS Concepts: • **Information systems** → **Learning to rank**;

Additional Key Words and Phrases: Large Language Models for Passage Ranking, In-context Learning, Few-shot Learning

ACM Reference format:
Wenhan Liu, Yutao Zhu, Zhicheng Dou, and Yujia Zhou. 2026. DemoRank: Selecting Effective Demonstrations for Large Language Models in Ranking Task. *ACM Trans. Inf. Syst.* 44, 3, Article 61 (March 2026), 25 pages.
https://doi.org/10.1145/3788867

This work was supported by the National Natural Science Foundation of China No. 62272467. The work was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.
Authors' Contact Information: Wenhan Liu (corresponding author), Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; e-mail: lwh@ruc.edu.cn; Yutao Zhu, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; e-mail:yutaozhu94@gmail.com; Zhicheng Dou, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; e-mail: dou@ruc.edu.cn; Yujia Zhou, Tsinghua University, Beijing, China; e-mail: zhouyujia@mail.tsinghua.edu.cn.
*For correspondence, please contact Zhicheng Dou.

ACM 1558-2868/2026/3-ART61
https://doi.org/10.1145/3788867

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable performance across a spectrum of **Natural Language Processing (NLP)** tasks. Recently, there has been significant interest in using LLMs for passage ranking tasks [29, 36, 48], which play a critical role in research of **Information Retrieval (IR)**. A typical approach is relevance generation, which judges the relevance of a query–passage pair in a pointwise manner. In this approach, LLMs are instructed to assess the relevance of a passage to a query by generating straightforward responses, typically in the form of "Yes" or "No." These responses are then used to compute the relevance score by calculating the log-likelihood of these generated responses. Previous studies [15, 48] have extensively demonstrated the effectiveness of this approach and highlighted its potential to improve the ranking quality of search engine systems.

In-Context Learning (ICL) has emerged as a powerful capability of LLMs, allowing them to adapt to specific tasks by leveraging a series of task demonstrations (i.e., input–output examples) [40]. This mechanism enhances the LLMs' understanding of both the task objective and the expected output format, thereby improving their overall performance. Many studies have investigated the strategy of demonstration selection for various NLP tasks [13, 23, 32, 41, 45], highlighting the importance of tailored demonstrations in achieving higher performance. Despite these advancements, the application of ICL to passage ranking tasks remains relatively unexplored. Given the inherent complexity of relevance assessment in passage ranking, ICL presents a challenging yet promising opportunity to enhance LLMs' ranking performance. Therefore, this study aims to develop effective demonstration selection strategies to optimize the performance of ICL in passage ranking.

Demonstration retrieval is a widely used technique for selecting demonstrations, showing promising results in the field of NLP. It retrieves demonstrations most similar to the test input for few-shot ICL. Despite its effectiveness, directly applying it to the passage ranking task may result in suboptimal performance. This is due to the complex nature of the query-passage relevance, which may require *a combination of multiple demonstrations* to provide diverse information for assessing the relevance. The example in Figure 1 illustrates such a problem. When selecting a 2-shot demonstration for the current input (a relevant query–passage pair), existing methods [32] directly choose the top-2 ranked demonstrations (z_1 and z_2) returned by the retriever. However, we deem that combining z_1 and z_5 is more suitable for this case. This is because z_1 and z_5 have more distinct queries and different relevance outputs. They provide LLM with *richer and more diverse query-passage relationships*, thus contributing more to the relevance assessment. Besides, since ICL is very sensitive to the order of demonstrations [23], it is also necessary to set an appropriate order for z_1 and z_5 (e.g., $[z_5, z_1]$). Thus, it is insufficient to select few-shot demonstrations based solely on similarity or relevance without considering their dependencies. In this article, we define the dependency between demonstrations in terms of their order and diversity.

In this article, we introduce a novel **Demonstration Reranker (DReranker)** that selects few-shot demonstrations through *dependency-aware demonstration reranking*. Different from existing **demonstration retrievers (DRetrievers)** [13, 32, 39] that directly choose the most similar demonstrations, our DReranker reranks the retrieved demonstrations by considering demonstration dependencies, making the combination of selected demonstrations more effective for the few-shot ranking tasks.

Existing methods for training a demonstration ranking model typically rely on LLMs' feedback [13, 32], which serves as the supervised training signals. They utilize an LLM to score demonstrations based on the LLM's likelihood of producing the correct output given each demonstration, and subsequently use the top-scored demonstrations as training samples to train the ranker. Following this technique line, we use the LLM's feedback to train our DReranker. Nevertheless, we face two

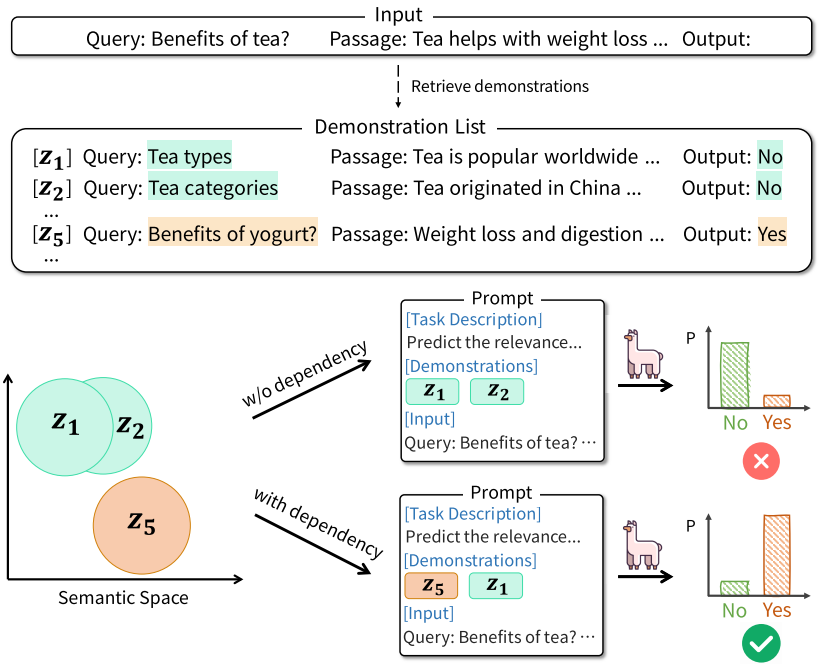

Fig. 1. Compared with choosing top-2 demonstrations (z_1 and z_2), the selection of z_5 and z_1 considers the demonstration dependencies (more diverse query-passage relationships and more appropriate demonstration order), thus yielding better relevance assessment.

significant challenges in generating its training samples: (1) The demonstration independence assumption: existing methods [32] assume the demonstration candidates are independent of each other and utilize the LLM's feedback to evaluate each individual candidate, which does not apply to training our dependency-aware DReranker. (2) High complexity: obtaining the optimal dependency-aware demonstration list from the retrieved demonstration set based on LLM's feedback is an NP-hard problem, which is highly time-consuming and of significant complexity.

To overcome these challenges, we propose an efficient approach to construct a kind of dependency-aware training samples. Our approach approximates the optimal demonstration list by iteratively selecting demonstrations from the retrieved set, which significantly reduces the search space and improves efficiency. In each iteration, the selection aims to maximize the LLM's feedback on the combination of the previously selected demonstrations and the current one, thus taking the demonstration dependencies into account. With these constructed training samples, we further design a list-pairwise training method for the DReranker's optimization, which teaches the reranker to iteratively select the next demonstration given a previous sequence.

To this end, we propose DemoRank, a few-shot *Demo*nstration selection framework for passage *Rank*ing. Specifically, we first design and train a ranking-task-specific DRetriever to obtain a list of high-quality demonstration candidates. Then, we introduce our dependency-aware DReranker, which iteratively selects few-shot demonstrations by reranking the demonstration candidate list. To overcome the challenges mentioned above, we propose an efficient approach to construct the training samples and design a list-pairwise training method for DReranker's optimization. Experiments on a series of ranking datasets demonstrate the effectiveness of DemoRank. Further analysis also shows the contribution of each proposed component and DemoRank's strong ability

under various scenarios, including low-resource, generalization on unseen datasets, transferability to different LLMs, and so on.

The main contributions of our article are summarized as follows:

(1) We propose a novel framework, namely DemoRank, that optimizes the selection of few-shot demonstrations for passage ranking tasks through dependency-aware demonstration reranking.
(2) To train our DReranker, we propose an approach to construct a kind of dependency-aware training samples in a time-efficient manner.
(3) We design a list-pairwise training method for the optimization of the DReranker.
(4) Extensive experiments on a series of ranking datasets demonstrate the strong ability of our DReranker under various scenarios.

2 Related Work

2.1 LLM For Passage Ranking

Passage ranking [15, 21, 22, 25, 36] is a crucial stage in IR that involves rearranging a list of candidate passages (usually obtained by a retriever) to better align with the relevance to a given query. It ensures that the relevant passages are displayed at the top of the list, better meeting the user's information needs. With the advancement of LLMs in IR [46], many studies have explored their use in passage ranking, typically categorized into three different types: pointwise ranking, pairwise ranking, and listwise ranking. In the following sections, we will elaborate on these three ranking methods, respectively.

2.1.1 Pointwise Ranking. The pointwise method assesses the relevance between a query and a single passage. A typical approach is relevance generation [15], which provides LLM with a query–passage pair and instructs it to output "Yes" if the passage is relevant to the query or "No" if not. The relevance score can be calculated based on the generation probability of the token "Yes." Zhuang et al. [48] propose a variant of relevance generation which incorporates multi-level relevance labels ("Highly Relevant," "Somewhat Relevant," and "Not Relevant") into the prompt for LLM passage rerankers. Another approach of pointwise methods is query generation [33, 49], which calculates a relevance score based on the log-likelihood of generating the query based on the passage.

2.1.2 Pairwise Ranking. In pairwise ranking [29], LLMs are given a prompt that consists of a query and a pair of passages and are instructed to identify the more relevant passage. To rank all passages, existing approaches typically employ aggregation methods such as AllPairs [29]. This method operates by first enumerating all possible passage pairs and producing discrete preference judgments for each pair (e.g., selecting Document 1 over Document 2), then aggregating these pairwise comparisons into final relevance scores. To optimize the ranking efficiency, specialized sorting algorithms, including heap sort and bubble sort are implemented. These algorithms leverage optimized data structures to perform selective passage pair comparisons, effectively promoting the most relevant documents to higher positions in the ranked list, which is particularly useful in top-k ranking.

2.1.3 Listwise Ranking. Listwise methods [3, 17, 19–21, 28, 36, 43, 44] take the query and a list of passages as the input and directly output the identifiers of the reranked passages (such as "[3] > [1] > [2]"). Due to the limited input length constraint of LLMs, directly incorporating all candidate passages into the prompt is infeasible. To address this limitation, existing methods adopt a sliding window strategy for incremental reranking. Specifically, passages are processed in subsets by sliding

a fixed-size window backward through the ranked list. At each step, only the passages within the current window are reranked, enabling efficient partial refinement while maintaining tractable computational costs. Listwise ranking has already been proven to have superior performance in some large-scale closed-source models (e.g., ChatGPT and GPT-4) [21, 36].

Despite the promising results of these methods, few studies have specifically focused on how to select effective few-shot demonstrations for ranking tasks, which is the primary focus of this article. Difficulty-based selection [9] is suboptimal because it applies the same few-shot demonstrations for all queries, which ignores that the optimal few-shot demonstrations vary across different queries. Previous studies [47] have revealed that the relevance generation of the pointwise method is the most suitable method for passage ranking on *open-source LLMs* when compared with other ranking methods. Thus, we intend to utilize the relevance generation approach for passage ranking in this article.

2.2 ICL

ICL has emerged as a powerful paradigm for adapting LLMs to downstream tasks without modifying their parameters. Unlike traditional fine-tuning, which requires computationally expensive updates to the model weights, ICL relies on a few input-output demonstrations provided at inference time to guide the model's behavior. This approach not only preserves the generality of pretrained LLMs but also mitigates common challenges such as overfitting and resource-intensive training. The effectiveness of ICL hinges on the quality and relevance of the demonstrations, prompting extensive research into methods for selecting optimal examples dynamically. As a result, ICL has become a cornerstone of modern LLM applications, offering a flexible and efficient alternative to traditional adaptation techniques.

A widely used demonstration selection approach is demonstration retrieval, where demonstrations are dynamically retrieved based on the input query rather than relying on static or handcrafted sets. Prior studies have explored different kinds of DRetrievers, which broadly fall into two categories: off-the-shelf DRetrievers and task-specific DRetrievers.

2.2.1 Off-the-Shelf DRetrievers. Many studies attempt to use the off-the-shelf retrievers that select demonstrations based on term-based similarity or sentence embedding similarity. For example, Agrawal et al. [1] propose to use BM25 for demonstration retrieval in machine translation. Due to BM25's reliance on term frequency and passage length, it ignores the semantic meaning and sentence structure, which may lead to sub-optimal performance in certain instances. Li and Qiu [14] investigate the effectiveness of SBERT embeddings for demonstration retrieval, and the results demonstrate the boost in performance compared to zero-shot or random demonstration selection. Liu et al. [16] propose to use a kNN-based retriever to retrieve demonstrations semantically similar to the test input. Instead of solely relying on term-match as in BM25, the last two studies apply the dense embeddings which can better capture semantic similarity (e.g., synonyms and related topics).

2.2.2 Task-Specific DRetrievers. While off-the-shelf retrievers have demonstrated potential in selecting demonstrations for LLMs, their generic nature often fails to capture the nature and solution patterns required by individual tasks, which may lead to sub-optimal performance. To mitigate this issue, many researchers have attempted to train a DRetrievers using task-specific signals. A key goal in designing a good DRetrievers is to encourage it to rank demonstrations higher if they are useful to the LLM as demonstrations. This allows training to rely directly on query and output pairs from the task itself, without needing human annotations. For instance, Rubin et al. [32] employ LLM feedback to distill information into a dense retriever, EPR, specifically for semantic parsing tasks, optimizing retrieval by focusing on the task's semantic requirements. Similarly, Das et al. [8] utilize F1 scores of logic forms as a training signal to improve retrieval for knowledge-based question–answering tasks, ensuring that retrieved demonstrations align closely with task-specific

evaluation metrics. Poesia et al. [27] finetune a DRetrievers for code generation, using the edit distance of abstract syntax trees as the basis. Cheng et al. [4] focus on training a DRetrievers that improves LLMs' performance in the cross-task and cross-model scenarios. Wang et al. [39] propose to train the retriever iteratively based on a reward model on various NLP tasks. In addition to DRetrievers, Shi et al. [34] propose to train a DReranker. They utilize LLM feedback to score each individual demonstration candidate, thereby generating a ranking list of candidates as training samples.

Although these methods have shown promising results, a common issue with them is that they treat each demonstration independently without considering the inherent dependencies among them. This oversight can result in suboptimal demonstration selection, as existing studies [12, 23] have shown that the order and diversity of demonstrations—referred to as demonstration dependencies in this article—significantly affect the effectiveness of few-shot ICL. For example, Levy et al. [12] reveal that choosing diverse demonstrations could substantially improve the performance of few-shot ICL in compositional generalization semantic parsing tasks. In this article, we introduce a novel framework that first retrieves a set of demonstrations and then reranks them in a dependency-aware manner. By considering demonstration dependencies, this framework aims to select the most effective few-shot demonstrations for passage ranking tasks, thereby improving overall performance.

3 Preliminaries

3.1 Relevance Generation for Ranking Task

Passage ranking constitutes a core component of modern IR systems, aiming to effectively order a list of retrieved passages based on their relevance to a given query. Formally, given a query q and a passage list $[p_1, \ldots, p_n]$, our objective is to compute a fine-grained score $Rs(q, p_i)$ for each passage that accurately reflects its relevance to the query. Recent advances in LLM-based relevance generation methods [15, 48] have demonstrated superior performance by leveraging the semantic understanding capabilities of LLMs. In these approaches, an LLM is provided with a prompt consisting of a query and a passage, and instructed to output a binary label "Yes" or "No" to indicate whether the passage is relevant to the query or not. Then a softmax function is applied to the logits of tokens "Yes" and "No," and the probability of the token "Yes" is used as the relevance score:

$$Rs(q, p_i) = \Pr(\text{"Yes"}|T, q, p_i), \tag{1}$$

where T is the task description, which is used in ICL to help LLMs understand the task [13, 47]. $\Pr(\text{"Yes''}|\cdot)$ gets the probability of generating token "Yes." Finally, the passages are ranked according to the relevance score $Rs(q, p_i)$ in descending order.

3.2 ICL in Relevance Generation

ICL has emerged as a powerful technique for adapting LLMs to downstream tasks without parameter updates, particularly effective in low-resource scenarios. In relevance generation task, given k in-context demonstrations $\{z_i\}_{i=1}^{k}$, where $z_i = (\hat{q}, \hat{p}, \hat{y})$ is a triple consisting of a query, a passage and a binary output ("Yes" or "No") indicating the relevance label (with "Yes" representing the relevance and "No" representing the irrelevance), the relevance score $Rs(q, p_i)$ is calculated by:

$$Rs(q, p_i) = \Pr(\text{"Yes"}|T, \{z_i\}_{i=1}^{k}, q, p_i). \tag{2}$$

The detailed task descriptions and the format of the demonstrations are shown in Tables 2 and 3, respectively. The main notations used in this article are listed in Table 1.

Table 1. The Notations Used in Our Framework

Notation	Explanation
q	User query.
p_i	The ith passage in the passage list.
$Rs(q, p_i)$	The relevance score between q and p_i.
z_i	The in-context demonstration.
$\mathcal{P}$	The demonstration pool for retrieval.
I	One piece of training input which consists of a query q and a passage p.
Z	The set of LLM-scored demonstration candidates for training DRetriever.
Z^r	The demonstration candidates retrieved by the trained DRetriever for training DReranker.
Z^s	The selected demonstrations when constructing dependency-aware training samples for DReranker.
Z^u	The unselected demonstrations when constructing dependency-aware training samples for DReranker.
K	The maximum iteration number of the construction of dependency-aware training samples.
I^{test}	One piece of test input which consists of a query q^{test} and a passage p^{test}.

Table 2. The Instructions Used for Different Datasets

Dataset	Instruction
FEVER	Given an article and a claim, predict whether the article is relevant to the claim by outputting either Yes or No. If the article is relevant to the claim, output Yes; otherwise, output No.
NQ	Given a passage and a question, predict whether the passage is relevant to the question by outputting either Yes or No. If the passage is relevant to the question, output Yes; otherwise, output No.
HotpotQA	Given a passage and a question, predict whether the passage is relevant to the question by outputting either Yes or No. If the passage is relevant to the question, output Yes; otherwise, output No.
TREC DL19	Given a passage and a query, predict whether the passage is relevant to the query by outputting either Yes or No. If the passage is relevant to the query, output Yes; otherwise, output No.
TREC DL20	Given a passage and a query, predict whether the passage is relevant to the query by outputting either Yes or No. If the passage is relevant to the query, output Yes; otherwise, output No.
MS MARCO	Given a passage and a query, predict whether the passage is relevant to the query by outputting either Yes or No. If the passage is relevant to the query, output Yes; otherwise, output No.

Table 3. The Demonstration and Input Format Used for Different Datasets

Dataset	Demonstration Format
FEVER	Article: #{ARTICLE}\nClaim: #{CLAIM}\nIs the Article relevant to the Claim?\nOutput:
NQ	Passage: #{PASSAGE}\nQuestion: #{QUESTION}\nIs the Passage relevant to the Question?\nOutput:
HotpotQA	Passage: #{PASSAGE}\nQuestion: #{QUESTION}\nIs the Passage relevant to the Question?\nOutput:
TREC DL19	Passage: #{PASSAGE}\nQuery: #{QUERY}\nOutput:
TREC DL20	Passage: #{PASSAGE}\nQuery: #{QUERY}\nOutput:
MS MARCO	Passage: #{PASSAGE}\nQuery: #{QUERY}\nOutput:

4 The DemoRank Framework

As mentioned in Section 1, directly using the most similar demonstrations retrieved is often suboptimal for the performance of ICL. In this article, we propose a novel approach by reranking the retrieved demonstrations based on their dependencies, enabling the combination of top-ranked ones to achieve significantly better ICL performance. Our DemoRank framework, illustrated in Figure 2, comprises a DRetrievers and a dependency-aware reranker (DReranker). The DRetriever is trained to retrieve high-quality demonstration candidates, while the DReranker constructs more

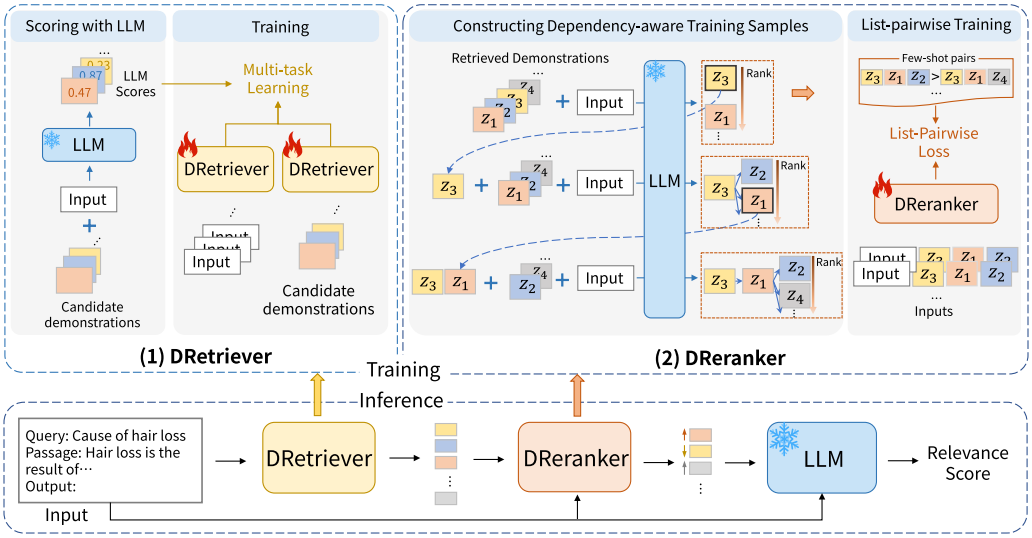

Fig. 2. An overview of our proposed framework DemoRank. It comprises two main components: DRetriever and DReranker. We first train the DRetriever using demonstration candidates individually scored by the LLM. Then, we construct dependency-aware training samples (i.e., few-shot pairs) through iterative selection and use a list-pairwise training approach to optimize the DReranker. During inference, we first use DRetriever to obtain a list of demonstrations and then apply DReranker to iteratively select top-k demonstrations for ICL.

effective few-shot demonstrations through demonstration reranking. In this section, we will provide a detailed introduction to our proposed DemoRank framework, including the construction of a demonstration pool, and the training method of DRetriever and DReranker.

4.1 Demonstration Pool Construction

Given a passage ranking dataset (e.g., MS MARCO [24]), we use its training set to construct our demonstration pool $\mathcal{P}$. For each query in the training set, we construct positive and negative demonstrations by pairing the query with its relevant and irrelevant passages, respectively. To maintain the output label balance in the demonstration pool $\mathcal{P}$, the number of negative demonstrations of each query is set equal to its positive demonstrations.

4.2 DRetrievers

The effectiveness of few-shot demonstrations depends on the quality of each demonstration. In this section, we train a ranking-task-specific DRetrievers to identify high-quality candidates for subsequent reranking. We apply an LLM to score a set of demonstration candidates to obtain supervised signals and use them to fine-tune our DRetriever based on a multi-task learning strategy.

4.2.1 Scoring with LLM. For a training input $I = (q, p)$ which contains a query–passage pair, we select a set of demonstrations from the demonstration pool $\mathcal{P}$ as training candidates. Following previous studies [39], we employ the BM25 algorithm to retrieve top-b demonstrations. Due to the complex nature of passage ranking, the utility of a demonstration is not directly related to its similarity to the input [9]. To include more potentially useful demonstrations for training, we also randomly sample another group of b demonstrations from $\mathcal{P}$. The total number of training candidates is annotated as N ($N = 2 * b$).

After that, we apply a frozen LLM scorer to score each demonstration z_i for the training input I based on the following equation:

$$f(z_i, I) = \frac{\Pr(y|T, z_i, I)}{\sum_{y' \in Y} \Pr(y'|T, z_i, I)}, \tag{3}$$

where y is the relevance label for the query–passage pair in I, $Y = \{\text{"Yes"}, \text{"No"}\}$ is the label space and T is the task description. In this article, the scorer uses the same LLM as the passage ranker. Nevertheless, we also explored the transferability of the LLM scorer on different LLM passage rankers in Section 5.8.

4.2.2 Training. Our DRetriever is based on a prevailing bi-encoder architecture [38]. Following existing DRetrievers [13, 39], the two encoders share the same parameters and encode the input and demonstrations separately, and then calculate their similarity. Given the current training input $I = (q, p)$ and its training candidates, we use encoder E_I and demonstration encoder E_z to encode them, respectively, and calculate the similarity score as:

$$S(I, z_i) = E_I(I)^\top E_z(z_i), \tag{4}$$

where the two encoders E_I and E_z share the same parameters and encode with average pooling.

Contrastive Loss. After that, we apply a contrastive loss L_c to maximize the score between the training input I and positive demonstration z^+ and minimize it for negative demonstration z_i^-. Here, z^+ is the demonstration with the highest LLM score, and z_i^- are the remaining ones. The contrastive loss L_c is calculated as:

$$L_c = -\log \frac{e^{S(I,z^+)}}{\sum_{z' \in Z} e^{S(I,z')}}, \tag{5}$$

where $Z = \{z^+, z_1^-, \ldots, z_{N-1}^-\}$.

Ranking Loss. To make use of the fine-grained supervision of LLM's feedback, we also use a ranking loss RankNet [2] to inject the ranking signal of candidates into training:

$$L_r = \sum_{i,j}^{|Z|} \mathbb{1}_{r_i < r_j} \log(1 + e^{S(I,z_j) - S(I,z_i)}), \tag{6}$$

where r_i is the rank of z_i in Z when sorted in descending order by the LLM score.

The final loss function L is defined as:

$$L = \lambda L_c + L_r, \tag{7}$$

where λ is a pre-defined hyper-parameter.

4.3 DReranker

During the training of DRetriever, each demonstration is evaluated independently, which overlooks the potential dependencies among them. This oversight may lead to a suboptimal combination of the top-retrieved demonstrations. To address this limitation, we present our novel DReranker, which reranks the retrieved demonstrations in a dependency-aware manner to obtain more effective few-shot demonstrations.

To overcome the challenges in constructing DReranker's training samples (i.e., the demonstration independence assumption and the issue of high complexity mentioned in Section 1), we propose a dependency-aware and time-efficient method for constructing training samples. Specifically, based on LLM's feedback, we greedily select demonstrations from the retrieved set to approximate the optimal few-shot demonstrations. The greedy selection significantly reduces the search space and time cost. And the LLM scorer evaluates the combination of multiple demonstrations at once, thus

taking the demonstration dependencies into account. In each selection, we construct a series of demonstration sequence pairs (with only the last demonstration being different). Building on these pairs, we design a list-pairwise training method that teaches the DReranker to select demonstrations based on a given sequence. During inference, our DReranker sequentially selects demonstrations to construct few-shot demonstrations. In the following section, we will provide a detailed introduction to our method for constructing dependency-aware training samples and our proposed list-pairwise training method.

4.3.1 Constructing Dependency-aware Training Samples. Given a training input I, we first use our trained DRetriever to retrieve top-M demonstrations Z^{r} as the training candidates. Then, we iteratively select demonstrations to estimate the optimal few-shot demonstrations. For the first iteration, we use the LLM scorer to score and rank these demonstrations based on Equation (3). Subsequently, we design a sampling strategy to select one demonstration from the unselected demonstration set Z^{u} ($Z^{\mathrm{u}} = Z^{\mathrm{r}}$ for the first iteration) based on its rank. The selection probability $p(\cdot)$ of each demonstration is computed by the following equation:

$$p(z_i) = \frac{g(z_i)}{\sum_{z_j \in Z^{\mathrm{u}}} g(z_j)}, \tag{8}$$

$$g(z_i) = \exp(-r_i), \tag{9}$$

where r_i is the rank of demonstration z_i. This strategy ensures that higher-scoring demonstrations are more likely to be selected for constructing the optimal demonstration sequence. Meanwhile, it retains the possibility of selecting lower-scoring demonstrations, which helps DReranker select the next demonstration when the previous sequence is not good enough. For the kth iteration, we append each unselected demonstration to the end of the previously selected demonstration sequence (obtaining k-shot demonstrations) and use the LLM scorer to score the k-shot demonstrations, respectively. Then, we select a demonstration from these unselected ones based on the same sampling strategy. After LLM scoring, we also obtain a ranking of k-shot demonstrations, based on which we obtain a set L_k which contains LLM-scored k-shot pairs with relative order (e.g., 3-shot pair $([z_3, z_1, z_2] > [z_3, z_1, z_4])$ in Figure 2). Each pair acts as a dependency-aware training sample and will be used by our list-pairwise training approach in the next section. In this way, the dependency between each demonstration and previous demonstrations is considered.

The ranking of few-shot demonstrations is regenerated in each iteration. The prompt used by the LLM scorer for scoring few-shot demonstrations is created by concatenating the task instruction, few-shot demonstrations, and input. The format of the instruction is shown in Table 2, and the format of demonstrations and input is shown in Table 3. Note that as the number of iterations increases, the computational cost associated with LLM inference also rises significantly. Due to our limited computational resources, we set a maximum iteration number K. The entire construction process is detailed in Algorithm 1.

4.3.2 List-Pairwise Training. According to Liu et al. [18], learning to rank methods can be divided into three categories: pointwise, pairwise, and listwise. Demonstration reranking is naturally a listwise problem because the rank of each demonstration depends on the previous demonstration sequence. In this part, we propose a list-pairwise training method to optimize our DReranker. We call it list-pairwise because the loss is calculated by comparing a pair of few-shot demonstration lists (l_1, l_2) where l_1 and l_2 differ only in the last demonstration. With the dependency-aware training samples from 1-shot to K-shot constructed, we generate all pairs of demonstration lists for each

Algorithm 1: Constructing Dependency-Aware Training Samples

Input: Training input I, maximum iteration K.
Output: Dependency-aware training samples O.

```
Retrieve top-M demonstrations Z^r
O ← {}, selected demonstrations Z^s ← [], unselected demonstrations Z^u ← Z^r
for i = 1 to K do
    Score and rank each z_j ∈ Z^u using f(Z^s ⊕ z_j, I) based on Equation (3)
    Append (I, Z^s, Z^u) and the ranking of scored Z^u to O
    Select a demonstration z* ∈ Z^u based on Equation (8)
    Append z* to Z^s, remove z* from Z^u
end for
return O
```

shot (for 1-shot, the list length is 1). Then, we calculate our list-pairwise loss L_{lp} as follows:

$$L_{\text{lp}} = \sum_{k=1}^{K} \sum_{l_i, l_j}^{L_k} J(l_i, l_j) \log(1 + e^{\text{Sc}(I, l_j) - \text{Sc}(I, l_i)}), \tag{10}$$

where L_k is the set of k-shot pairs and $J(l_i, l_j)$ is a indicator function. If the LLM score of l_i is bigger than l_j, $J(l_i, l_j) = 1$, else $J(l_i, l_j) = 0$. Our DReranker is based on a cross-encoder model, which calculates $\text{Sc}(I, l_i)$ by taking the concatenation of the training input I and demonstration list l_i as input, and outputs a prediction score using the representation of "[CLS]" token. Observing demonstration lists of different shots helps the DReranker learn to select the next demonstration given the selected sequence.

4.4 Inference

During inference, we first encode the entire demonstration pool $\mathcal{P}$ using our trained DRetriever and build the index. Then, given a test input $I^{\text{test}} = (q^{\text{test}}, p_i^{\text{test}})$, we first retrieve top-$D$ demonstrations using DRetriever. After that, we perform dependency-aware reranking of these retrieved demonstrations through iterative selection. Specifically, for each selection, we choose the demonstration from the unselected ones that, when concatenated with the selected demonstration sequence, results in the highest LLM score. The process is repeated until K demonstrations are selected. Finally, these K few-shot demonstrations will be concatenated with the test input to calculate the relevance score. We perform this process for all retrieved passages of the test query q^{test} and rank passages based on their relevance scores.

5 Experiments

5.1 Setting

5.1.1 Datasets. To comprehensively evaluate our approach, we conducted extensive experiments on six benchmark datasets spanning three distinct ranking scenarios: question-answering, fact-checking, and passage retrieval.

Question-Answering Scenario. This scenario focuses on retrieving relevant passages that directly answer user queries. We evaluate two widely used QA datasets:

— HotpotQA [42]: A comprehensive multi-hop question-answering dataset containing 113k Wikipedia-based question–answer pairs, requiring multi-hop reasoning across multiple passages to arrive at the correct answers. The test set consists of 7,405 carefully curated questions.
— **Natural Questions (NQ)** [11]: A large-scale open-domain QA dataset based on real user queries sourced from Google search, containing 307k training examples with Wikipedia

articles as context. The test set includes 3,452 questions for the evaluation of the model's performance.

Fact-Checking Scenario. This scenario assesses the ability to verify factual claims using supporting evidence:

—FEVER [37]: A fact verification dataset containing 185k claims extracted from Wikipedia. The test set comprises 6,666 claims, requiring models to determine whether each claim is supported, refuted, or not enough information is available. This dataset is important for developing models that can accurately perform fact-checking and verify information across various topics.

Passage Retrieval Scenario. This scenario evaluates general-purpose passage retrieval capabilities:

—MS MARCO [24]: A large-scale web search dataset with 8.8M passages and 532k training queries from Bing search logs. We use its development set containing 6,980 queries for evaluation.
—DL19 (TREC Deep Learning Track 2019) [7]: A standard retrieval benchmark containing 43 queries with detailed relevance judgments, commonly used for official TREC evaluations.
—DL20 (TREC Deep Learning Track 2020) [6]: The subsequent TREC benchmark with 54 queries, featuring more challenging and diverse queries than its predecessor.

For HotpotQA, NQ, and FEVER, models were trained on each dataset's training set and evaluated on its corresponding test set. For MS MARCO, models were trained on its comprehensive training set and tested on three evaluation sets: the MS MARCO development set, DL19, and DL20.

5.1.2 Implementation Details. We adopt FLAN-T5-XL [5] as the LLM scorer and passage ranker (unless otherwise specified) due to its strong performance in pointwise ranking tasks, as demonstrated in prior work [35]. For the construction of training input, we pair each query in the training set with one relevant passage and one irrelevant passage, respectively, thus generating two training inputs. The maximum length of query and passage is set as 64 and 100, respectively. Next, we will introduce the training details of our models. As for the DRetriever, we set the number of demonstration candidates N as 50 and hyper-parameter λ for multi-task learning as 0.2. Following the previous study [39], we use e5-base-v2 [38] to initialize DRetriever and train the DRetriever with a learning rate of 3e-5. As for DReranker, the number M of retrieved demonstrations for scoring is 50, and the maximum iteration number K in Section 4.3 is set as 3. We apply DeBERTa-v3-base [10] for model initialization for its efficient yet powerful sequence modeling capabilities and optimize the model with a learning rate of 1e-5. Both the DRetriever and DReranker are trained for 2 epochs. The choices of learning rates above follow the pretraining setups of their corresponding backbone models, while the 2-epoch training schedule ensures convergence without overfitting. During inference, the number of retrieved demonstrations D for reranking is set as 30 to balance effectiveness and efficiency, achieving near-optimal performance with significantly lower computational cost.

The prompt we used in this article consists of the instruction, demonstrations (one or more), and test input. For zero-shot, no demonstrations are included. The instructions and demonstrations we used are listed in Tables 2 and 3, respectively. The instructions are used only for each test query–passage pair and the LLM scoring process. The test inputs have the same format as the demonstration.

5.1.3 Baselines. We compare DemoRank with a series of baselines, including Initial Order, 0-shot, rule-based demonstration selection (Random, K-means, and DBS [9]), and retrieval-based selection (BM25 [31], SBERT [30], E5 [38], EPR [32]). The baseline details are shown as follows:

— *Initial Order*: Following previous studies [36, 48], we use the top-100 passages retrieved by BM25 as the initial passages order, which serves as the foundation for all subsequent reranking operations. This baseline reflects the raw retrieval performance before any passage reranking is applied.
— *0-shot*: The passage ranking approach based on pure relevance generation without any demonstration, representing the LLM's intrinsic ranking capability without ICL. This baseline is used to compare with demonstration selection baselines.
— *Random*: We randomly sample demonstrations from the demonstration pool $\mathcal{P}$ for each test input, serving as a lower-bound demonstration selection baseline.
— *K-means*: This method first clusters all the demonstrations in the demonstration pool into k clusters using a dense embedding model, then selects the k demonstrations closest to each cluster center. The approach aims to maximize demonstration diversity by covering different semantic clusters in the embedding space. Following Wang et al. [39], we use E5 [38] to encode all the demonstrations.
— *DBS* [9]: DBS is a rule-based approach based on demonstration difficulty. It selects the demonstrations that are the most difficult for the LLM to predict. In this article, we implemented the algorithm based on the relevance generation approach for a fair comparison.
— *BM25* [31]: BM25 is a widely used sparse retriever. We apply it to retrieve demonstrations most similar to the test input.
— *SBERT* [30]: We use Sentence-BERT as the off-the-shelf DRetriever following [32].[1] We first use Sentence-BERT to build the index for all the demonstrations. Then, we compute cosine similarity between test inputs and demonstrations in the sentence embedding space, selecting the most semantically similar demonstrations.
— *E5* [38]: E5 is a state-of-the-art text embedding model trained contrastively on diverse text pairs, enabling robust zero-shot retrieval and semantic understanding. Following Wang et al. [39], we utilize e5-base-v2 checkpoint[2] as our off-the-shelf DRetriever and apply the same retrieval method as we do with SBERT.
— *EPR* [32]: EPR is a competitive dense DRetriever fine-tuned with task-specific training data. It first applies a frozen LLM scorer to label positive and negative demonstrations based on the LLM's feedback. Then, it uses contrastive learning with in-batch negatives for retriever optimization. We trained EPR for demonstration retrieval in the passage ranking task using the same training data as DemoRank.
— *LLM-R* [39]: LLM-R first uses LLM feedback to train a reward model to assess the quality of demonstrations and then distil the knowledge of the reward model to a DRetriever.

5.2 Main Results

We compare DemoRank with baselines in 3-shot ICL, and the results are shown in Table 4. From analyzing the results, we draw the following observations:

(1) Our framework DemoRank shows the best performance on all datasets and surpasses all the baseline models with p-value < 0.05 (with Bonferroni correction), which indicates the significant improvements and robustness of DemoRank on different ranking scenarios and query types. Besides, compared with SOTA DRetriever LLM-R, DemoRank has a significant improvement (e.g., about 2.5 NDCG@10 points on HotpotQA and 3 NDCG@10 points on FEVER, respectively). This indicates that DemoRank can capture the dependencies between

[1]The checkpoint is from https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2.
[2]https://huggingface.co/intfloat/e5-base-v2 .

Table 4. Main Results on Different Datasets

Method	HotpotQA		NQ		FEVER		DL19		DL20		MS MARCO		Average	
	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10
Initial Order	61.10	63.30	26.74	30.55	62.94	65.13	52.78	50.58	50.67	47.96	19.74	22.84	45.66	46.73
0-shot	57.76	60.65	45.71	48.62	36.29	38.92	67.48	66.13	68.43	65.57	30.18	33.24	50.98	52.19
Random	56.45	59.42	45.96	48.61	35.94	38.61	67.66	66.57	66.27	64.84	30.48	33.70	50.46	51.96
K-means	56.34	59.27	46.09	48.71	35.81	38.33	67.29	66.30	69.08	66.22	30.65	33.73	50.88	52.09
DBS	57.29	60.15	45.73	48.62	36.34	39.00	68.15	66.40	68.54	65.21	30.39	33.61	51.07	52.17
BM25	60.64	63.18	47.17	49.78	37.39	40.19	67.73	66.08	68.21	65.85	31.02	34.03	52.03	53.19
SBERT	55.24	58.38	46.30	49.23	33.95	36.80	67.69	66.67	67.46	65.07	30.48	33.71	50.19	51.64
E5	60.90	63.42	46.98	49.60	37.02	39.71	69.27	66.40	67.60	65.33	31.13	34.07	52.15	53.09
EPR	62.04	64.59	47.49	50.07	42.70	45.23	69.60	66.12	68.55	65.34	31.28	34.29	53.61	54.27
LLM-R	61.89	64.54	48.34	50.86	43.42	45.85	69.53	65.98	68.45	65.79	30.99	33.98	53.77	54.50
DemoRank	**64.84**†	**67.03**†	**49.77**†	**52.31**†	**46.08**†	**48.43**†	**70.15**†	**67.76**†	**69.62**†	**66.76**†	**32.27**†	**35.31**†	**55.46**	**56.27**

Due to space constraints, we abbreviate NDCG@5 and NDCG@10 as "@5" and "@10," respectively. The best results are marked in bold, and the column average represents the average performance of all datasets. "†" indicates the model outperforms the best baseline significantly with a paired t-test at p-value < 0.05 level (with Bonferroni Correction). The best result is marked in bold.

demonstrations, thereby constructing more effective few-shot demonstrations for passage ranking.

(2) The similarity-based demonstration selection methods, such as BM25 and E5, exhibit a certain degree of improvement when compared to the 0-shot baseline. This observation suggests that the similarity between the demonstration and the input contributes to selecting useful demonstrations. However, these similarity-based methods remain significantly inferior to both LLM-R and our proposed DemoRank. This performance gap underscores the critical importance of incorporating task-specific optimization through LLM feedback, rather than relying solely on general-purpose similarity metrics (such as term overlap and semantic similarity).

(3) Rule-based methods (such as random selection and K-means) struggle to achieve performance improvements and even degrade ranking results below the 0-shot baseline on certain datasets like HotpotQA. This is quite different from the phenomenon observed in many NLP tasks [13, 39]. This significant difference highlights the unique challenges inherent in few-shot demonstration selection for passage ranking tasks, which require sophisticated approaches to achieve performance improvement.

(4) The improvement from DemoRank compared to 0-shot varies greatly across datasets. For instance, DemoRank yields a 2-point gain on MSMARCO but nearly a 7-point boost on HotpotQA. We speculate that this is due to the varying difficulty levels of different ranking datasets. HotpotQA is a multi-hop QA dataset, which requires more complex query understanding and reasoning than the MSMARCO passage ranking dataset. Therefore, introducing demonstrations can better help the LLM understand the task, leading to greater improvements.

5.3 Different Variants of DemoRank

To understand the effectiveness of each component in our proposed DemoRank framework, we evaluate different variants of DemoRank on three datasets—FEVER, NQ, and DL19—each of which represents distinct ranking scenarios. We employ 3-shot demonstrations for each variant and present the results in Table 5.

One key variant involves removing the DReranker from DemoRank, leaving only the DRetriever. This variant is referred to as "DemoRank w/o DReranker." We can see that the absence of the

Table 5. Results (NDCG@10) of Different Variants

Method	FEVER	NQ	DL19	Average
Ablation study				
– DRetriever w/o L_r	43.65	50.65	66.06	53.45
– DemoRank w/o DReranker	44.40	50.98	66.49	53.95
– DemoRank w/o Dependency	46.64	51.22	66.89	54.91
DemoRank	**48.43**†	**52.31**†	**67.76**†	**56.17**
Using E5 as DRetriever				
E5	39.71	49.60	66.40	51.90
E5+DReranker	**47.29**	**51.76**	**68.19**	**55.75**
Using EPR as DRetriever				
EPR	45.23	50.07	66.12	53.80
EPR+DReranker	**48.00**	**51.60**	**67.81**	**55.80**

"†" indicates the model outperforms all the variants significantly with a paired t-test at p-value < 0.05 level. The best result is marked in bold.

DReranker leads to an average drop of 2.2 points, demonstrating its effectiveness in the DemoRank framework. Another variant, designed to test the impact of dependency-aware reranking, is "DemoRank w/o Dependency." This variant uses LLM to score each demonstration candidate independently, following the same scoring strategy as DRetriever, and applies RankNet for optimization. Ignoring the demonstration dependencies, this variant causes significant drops across all datasets. Notably, it drops the performance by about 2 points on FEVER, confirming the effectiveness of our dependency-aware demonstration reranking. We also investigate the effectiveness of ranking loss L_r on the training of the DRetriever. Note that "DemoRank w/o DReranker" represents only applying DRetriever for demonstration selection. Removing L_r (denoted as DRetriever w/o L_r) causes a performance drop, such as about 1 point on FEVER, which indicates that incorporating ranking signals of demonstration candidates is beneficial for retrieving more effective demonstrations.

Lastly, we assess the training effectiveness of our DReranker by integrating it with different DRetrievers. Specifically, we replace our DRetriever with E5 and EPR, respectively, creating two variants referred to as "E5+DReranker" and "EPR+DReranker." The results demonstrate that the incorporation of DReranker significantly enhances the ICL performance. For example. "E5+DReranker" outperforms E5 for about 4 points on average and "EPR+DReranker" outperforms EPR by 2 points on average. These experimental results indicate that our DemoRank is flexible and not restricted by specific DRetrievers.

5.4 Comparison with Supervised Rankers

The training of DemoRank is primarily based on queries in the training set, which can also be used to directly fine-tune a supervised passage ranker. In this part, we conduct systematic comparisons against three representative supervised passage rerankers: (1) monoBERT (340 M) [25], a BERT-based pointwise ranker that formulates ranking as a binary classification task; (2) monoT5 (220 M parameters) [26], which treats relevance prediction as a text generation task by outputting "true" or "false" token; and (3) our implemented monoFLAN-T5, a FLAN-T5-XL based ranker specifically developed for this study to ensure a fair comparison with DemoRank in LLM backbone. The monoFLAN-T5 adopts the identical generative training strategy as monoT5, processing query–passage pairs through sequence-to-sequence modeling while optimizing the same binary classification objective. Our experimental design evaluates two distinct training scenarios: a high-resource setting

Table 6. Results (NDCG@10) on MS MARCO, DL19 and DL20

QNum	Method	MS MARCO	DL19	DL20	Avg.
0	0-shot	33.24	66.13	65.57	54.98
500K	monoBERT	39.97	**70.72**	67.28	**59.32**
	monoT5	**40.05**	70.58	**67.33**	**59.32**
	monoFLAN-T5	36.22	**70.72**	66.26	57.73
	DemoRank	35.31	67.76	66.76	56.61
100K	monoBERT	34.24	66.04	63.10	54.46
	monoT5	34.25	67.79	59.31	53.78
	monoFLAN-T5	**37.35**	**70.84**	65.12	**57.77**
	DemoRank	34.90	67.83	**67.43**†	56.72
20K	monoBERT	30.69	63.61	59.32	51.21
	monoT5	29.79	61.16	52.72	47.89
	monoFLAN-T5	32.90	64.23	64.90	54.01
	DemoRank	**34.97**†	**67.33**†	**67.53**†	**56.61**
1K	monoBERT	24.33	58.57	47.73	43.54
	monoT5	25.69	61.10	51.22	46.00
	monoFLAN-T5	32.51	65.26	65.14	54.30
	DemoRank	**34.03**†	**68.08**†	**66.01**†	**56.04**

QNum represents the number of queries used in the MS MARCO training set. All models rerank the top-100 passages retrieved by BM25. "†" indicates demorank outperforms all the baselines significantly with a paired t-test at p-value < 0.05 level. The best result is marked in bold.

with 500K (full training set) and 100K training queries, and a low-resource setting with only 20K and 1K queries, using MS MARCO dataset as our training set. All passage rerankers rerank the top-100 passages retrieved by BM25. Performance is measured using the NDCG@10 metric, with zero-shot performance included as an important reference. The complete results are presented in Table 6.

From the results, we can see that the traditional supervised rerankers demonstrate a performance advantage in the high-resource scenario (500K and 100K queries). However, in the low-resource setting (20K queries), DemoRank significantly outperforms these supervised rerankers on three datasets and also shows a stable improvement over the 0-shot baseline. This suggests that when training data is limited, DemoRank is more effective than supervised models, highlighting the few-shot ability of DemoRank in low-resource scenarios.

5.5 Different Demonstration Numbers

The number of demonstrations is a critical factor that significantly influences the performance of ICL. In this section, we analyze how varying the number of few-shot demonstrations affects the performance of our models. Specifically, we evaluate the performance of DemoRank in comparison to the EPR baseline on two datasets: FEVER and HotpotQA, with NDCG@10 as the evaluation metric. To highlight the contribution of our DReranker, we also include a comparison with DRetriever.

As illustrated in Figure 3, both DRetriever and DemoRank consistently outperform the EPR baseline across different numbers of demonstrations. This consistent performance advantage

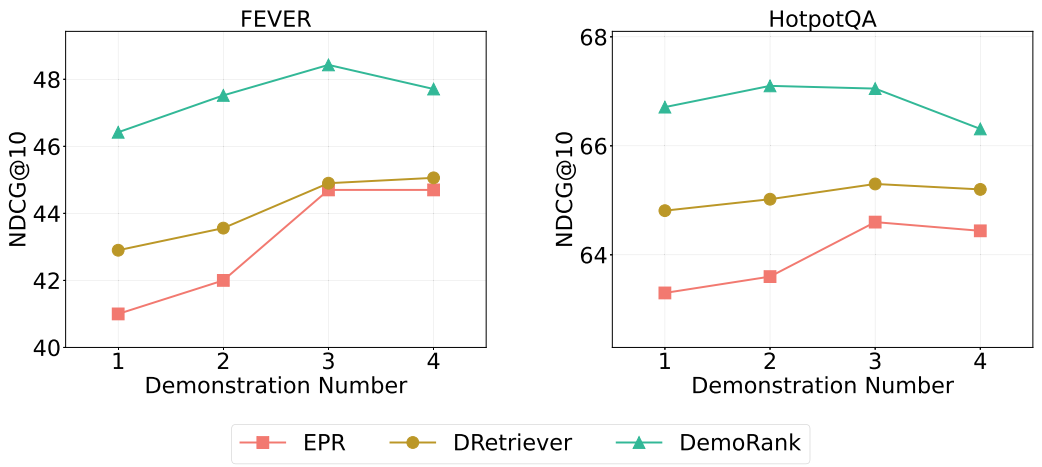

Fig. 3. The impact of different demonstration number (1, 2, 3, and 4) on model performance on FEVER and HotpotQA datasets. We use NDCG@10 as the evaluation metric.

underscores the effectiveness of our training approach. Furthermore, our trained DReranker shows consistent improvements over DRetriever, showcasing its effectiveness in selecting more effective few-shot demonstrations and enhancing overall ICL performance. Besides, we observe that when expanding from 3-shot to 4-shot, the performance of DemoRank decreases on two datasets (e.g., 0.7-point drop on FEVER and 0.8-point drop on HotpotQA). We speculate that this is because 3-shot demonstrations already provide sufficient knowledge, and adding more demonstrations may introduce unnecessary noise and hinder the ICL performance.

5.6 The Impact of Retrieved Demonstration Number D

Our DReranker is designed to select effective few-shot demonstrations from top-D demonstrations retrieved by DRetriever. Thus, the performance of DReranker highly relies on the quality of these retrieved D demonstrations. In this part, we intend to explore the impact of different retrieval numbers D on the performance of the DReranker. We choose different retrieval numbers D (10, 20, 30, 40, and 50) to conduct demonstration reranking using our trained DReranker and subsequently evaluate the performance of 3-shot ICL on two in-domain datasets: FEVER and HotpotQA. We also report the performance of DRetriever as a reference. The results are shown in Figure 4. From the results, we observe that increasing the retrieval number D consistently enhances the DemoRank's performance on both datasets. This finding indicates that a larger pool of retrieved demonstrations allows the DReranker to select more effective few-shot demonstrations. Besides, we observe that the results reveal a performance decline when increasing D from 40 to 50. This may be because the expansion of the retrieval number introduces some low-quality demonstrations, reducing the effect of few-shot ICL. Considering the marginal improvement brought by increasing the retrieval number from 30 to 40, we set the retrieval number to 30 in our main experiment to achieve a tradeoff between effect and efficiency.

5.7 Generalization across Different Datasets

One of the application scenarios of DemoRank is its generalization to unseen datasets. To validate this capability, we conduct an evaluation of DemoRank across a series of BEIR datasets. We choose the 0-shot and DRetriever EPR for comparison. Both DemoRank and EPR are trained on the MS MARCO dataset. We also incorporate five widely used passage rankers as our baselines:

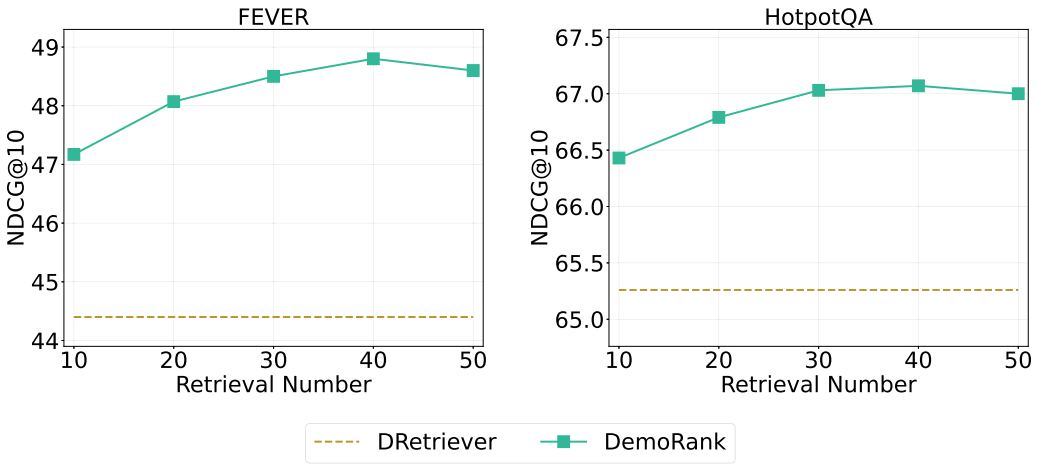

Fig. 4. The impact of different retrieved demonstration number D (10, 20, 30, and 40) on the performance of DemoRank on FEVER and HotpotQA datasets. NDCG@10 is used as the evaluation metric.

Table 7. Results (NDCG@10) on BEIR

Method	Robust04	SCIDOCS	DBPedia	NEWS	FiQA	Quora	NFCorpus	Average
Initial Order	40.70	14.90	31.80	39.52	23.61	78.86	33.75	37.59
MonoBERT	44.18	15.99	41.70	44.62	32.06	74.65	34.97	41.17
monoELECTRA	42.97	16.31	40.39	47.56	32.18	81.67	34.52	42.23
MonoT5	44.19	16.49	42.42	46.83	33.24	82.17	35.42	42.97
RankGPT	45.61	16.22	40.66	44.40	26.89	74.09	34.68	40.36
RankZephyr	48.65	18.19	42.95	**48.62**	31.88	83.29	37.48	44.44
Using Flan-T5-XL (3B) as LLM passage reranker								
0-shot	47.90	16.33	36.22	45.01	35.30	83.42	35.89	42.87
EPR	47.37	16.65	37.61	46.23	34.75	83.87	35.67	43.16
DemoRank	48.94	17.09	39.67	46.66	36.18	84.04	36.05	44.09
Using Flan-T5-XXL (11B) as LLM passage reranker								
DemoRank	**53.68**	**18.20**	**45.43**	46.33	**37.55**	**86.16**	**37.62**	**46.42**

Best results are marked in bold. We use MS MARCO's demonstration pool for retrieval and 3-shot ICL for E5 and DemoRank. Note that while using Flan-T5-XXL (11B) as LLM passage reranker, we still utilize the DRetriever and DReranker trained with the demonstration scorer Flan-T5-XL. The best result is marked in bold.

MonoBERT, monoT5, monoELECTRA, RankGPT,[3] and RankZephyr. We use the demonstration pool from MS MARCO due to the lack of training sets in the BEIR datasets. Furthermore, given that our DemoRank's LLM passage reranker is not limited to a specific LLM (e.g., Flan-T5-XL), we explore the use of a more powerful LLM, Flan-T5-XXL, as the passage reranker, while still utilizing the DRetriever and DReranker trained with the demonstration scorer Flan-T5-XL. The results are shown in Table 7. From the results, we have the following observations:

(1) The results in Table 7 show that, when using Flan-T5-XXL as LLM passage reranker, our DemoRank achieves an average performance of 46.42, outperforming all baselines and

[3]To ensure a fair comparison in model size, we use Mistral-7B-Instruct-v0.3 as RankGPT's backbone instead of GPT-4.

exceeding the best-performing baseline RankZephyr by approximately two points on average. This demonstrates the strong generalization ability of DemoRank.

(2) When using Flan-T5-XL (3B) as the passage reranker, our DemoRank also surpasses nearly all baselines and achieves average performance comparable to RankZephyr. Note that RankZephyr relies on a larger 7B backbone model and is distilled from a more powerful teacher model GPT-4. Such a fine-tuning approach can compromise the inherent zero-shot capabilities of LLM on tasks beyond ranking. In contrast, our DemoRank (3B) achieves comparable results without such fine-tuning, thereby preserving the model's versatility across various tasks. This demonstrates the efficiency and adaptability of DemoRank, effectively leveraging a smaller 3B model while maintaining broader applicability.

(3) Despite utilizing demonstrations from an out-of-domain dataset MS MARCO, DemoRank consistently improves upon the 0-shot baseline (when using Flan-T5-XL (3B) as LLM passage reranker) across all BEIR datasets. This indicates not only the potential of cross-dataset demonstrations but also highlights the adaptability and effectiveness of DemoRank in diverse scenarios.

5.8 Transferability across Different LLM Rankers

In previous experiments, we utilized the same LLM (Flan-T5-XL) as both the demonstration scorer and passage ranker within the DemoRank framework. However, it remains uncertain whether the passage ranker can be replaced with other LLMs during the inference stage while keeping its ranking performance. To investigate this, we evaluate DemoRank's transferability across different LLM-based rankers and compare its effectiveness against multiple baselines, including the 0-shot baseline and three DRetrievers: BM25, E5, and EPR. We select two distinct LLMs as alternative passage rankers: Flan-T5-XXL[4] and Mistral-7B-Instruct-v0.3.[5] The two LLMs differ from Flan-T5-XL either in scale (Flan-T5-XXL being significantly larger) or in architectural design (Mistral-7B being a decoder-only model, unlike the encoder–decoder Flan-T5 family). The results are presented in Table 8.

From the results, we can see that DemoRank, based on two distinct LLM rankers, surpasses all the baselines on the Average metric. It also successfully passes the t-test with a p-value < 0.05 on most datasets, demonstrating its strong transferability across different LLM passage rankers. Moreover, we observe that when using Flan-T5-XXL as LLM Ranker, DemoRank achieves higher performance on datasets such as FEVER, DL19, and MS MARCO (with NDCG@10 of 50.89, 69.23, and 36.05, respectively) compared with Flan-T5-XL (with scores of 48.43, 67.76, and 35.31, as shown in Table 4). This illustrates DemoRank's potential to enhance passage ranking when larger-scale LLM rankers are utilized.

5.9 The Tradeoff between Effectiveness and Efficiency

In this work, we introduce a novel DReranker as a key component of the DemoRank framework. While this step significantly improves passage ranking performance by selecting higher-quality few-shot demonstrations, it inevitably introduces additional computational overhead. To quantify this tradeoff, we conduct an evaluation measuring both the ranking performance gains (NDCG@10) and the latency increase (seconds per query). We experiment on two different datasets FEVER and HotpotQA, with two different ICL settings 2-shot and 3-shot, respectively. We choose FLan-T5-XXL as the passage ranker and conduct the experiment on 2 NVIDIA A800 80GB GPUs. The experimental results are shown in Table 9. The variant "DemoRank w/o DReranker" represents only

[4]https://huggingface.co/google/flan-t5-xxl .
[5]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3 .

Table 8. The Performance (NDCG@10) of Different LLM Passage Rankers

Method	HotpotQA	NQ	FEVER	DL19	DL20	MS MARCO	Average
Initial Order	63.30	30.55	65.13	50.58	47.96	22.84	46.73
Mistral-7B-Instruct-v0.3							
0-shot	53.69	35.59	34.98	55.23	49.95	24.63	42.35
BM25	58.10	43.25	35.48	64.82	**61.00**	26.57	48.20
E5	57.69	43.69	34.89	62.15	54.75	26.55	46.62
EPR	62.24	41.24	47.42	**66.98**	57.63	26.71	50.37
DemoRank	**65.90**†	**43.79**	**58.09**†	64.85	58.29	**28.02**†	**53.16**
Flan-T5-XXL							
0-shot	56.64	47.61	37.38	66.22	64.30	34.29	51.07
BM25	60.66	50.47	43.89	66.82	65.67	35.15	53.78
E5	60.74	50.14	43.86	66.45	65.44	34.84	53.58
EPR	59.19	50.45	45.84	68.12	64.12	34.30	53.67
DemoRank	**62.44**†	**51.68**†	**50.89**†	**69.23**†	**65.71**	**36.05**†	**56.00**

We apply 3-shot ICL for DRetrievers (BM25 and E5), as well as our DemoRank framework. The "†" symbol indicates that the model outperforms all the baselines significantly ($p < 0.05$, Paired t-Test). The best result is marked in bold.

Table 9. The Performance (NDCG@10) Improvement and Latency (Seconds) per Query Increased by DReranker

Shot Number	Method	FEVER		HotpotQA	
		NDCG@10	Latency	NDCG@10	Latency
2-shot	DemoRank w/o DReranker	46.53	14.33	54.97	17.71
	DemoRank	**50.02**†	15.46	**61.29**†	19.21
3-shot	DemoRank w/o DReranker	47.62	18.69	57.48	23.64
	DemoRank	**50.89**†	20.64	**62.44**†	25.93

We rerank the top-100 passages retrieved by BM25. "†" represents the significant improvement of demorank with paired t-test at p-value < 0.05 level, compared with "DemoRank w/o DReranker."

using DRetriever to retrieve in-context demonstrations for passage ranking. The results indicate that DemoRank achieves a significant improvement of about 3–6 points in NDCG@10 (with paired t-test at p-value < 0.05 level) with only approximately a 7–10% increase in latency. This demonstrates that our lightweight DReranker effectively enhances the quality of demonstration selection while maintaining a good balance between performance enhancement and efficiency.

In this work, we introduce a novel DReranker as a key component of the DemoRank framework. While this step significantly improves passage ranking performance by selecting higher-quality few-shot demonstrations, it inevitably introduces additional computational overhead. To quantify this tradeoff, we conduct an evaluation measuring both the ranking performance gains (NDCG@10) and the latency increase (seconds per query) on the FEVER dataset. For our experiment, we choose FLan-T5-XXL as the passage ranker with 3-shot in-context demonstrations and conduct the experiment on 8 NVIDIA A800 80GB GPUs. The experimental results are shown in Table 9. The variant "DemoRank w/o DReranker" represents only using DRetriever to retrieve top-3 in-context

Table 10. A Case Study for Judging the Relevance of a *Relevant Query–Passage Pair* in DL19 Dataset

Test Input	*Passage*: William Booth. William Booth (April 10, 1829–August 20,1912) was the founder and first General (1878–1912) of The Salvation Army. Originally a Methodist lay preacher, William Booth saw tremendous need not being fulfilled by mainstream churches in Victorian England. In his honor... *Query*: *When* was the *salvation army* founded? *Output*:
Demonstrations of EPR	*Passage*: THE SALVATION ARMY. The *Salvation Army* is an integral part of the Christian Church, although distinctive in government and practice. The Army's doctrine follows the mainstream of Christian belief and its articles of faith emphasize God's saving purposes. *Query*: Army article failure to follow lawful order *Output*: No *Passage*: Goods to The *Salvation Army*, those items are then sold at our Family Stores. And the proceeds are used to fund our Adult Rehabilitation Centers, where those in the grip of addiction find help, hope, and a second chance at life. *Query*: Where to donate used items to salvation army? *Output*: No *Passage*: Glassdoor is your resource for information about the Health Insurance benefits at Salvation Army. Learn about Salvation Army Health Insurance, including a description from the employer, and comments and ratings provided anonymously by current and former... *Query*: What do the *salvation army* do? *Output*: No
Relevance score	0.32
Demonstrations of DemoRank	*Passage*: THE SALVATION ARMY. The *Salvation Army* is an integral part of the Christian Church, although distinctive in government and practice. The Army's doctrine follows the mainstream of Christian belief and its articles of faith emphasise God's saving purposes. *Query*: army article failure to follow lawful order *Output*: No *Passage*: OPM was originally founded as the United States Civil Service Commission by the Pendleton Civil Service Reform Act of 1883. The commission was abolished and replaced by OPM on January 1, 1979, following the passage of the Civil Service Reform Act of 1978 and... *Query*: *When* was opm established? **Output**: Yes *Passage*: The Little Prince first published in 1943, is a novella, the most famous work of French aristocrat, writer, poet, and pioneering aviator Antoine de Saint-Exup. *Query*: *When* was the little prince book published? *Output*: Yes
Relevance score	0.95

Note that the higher predicted relevance score indicates the demonstrations are more effective.

demonstrations for passage ranking. The results indicate that DemoRank achieves a significant improvement of about 3 points in NDCG@10 (with paired t-test at p-value < 0.05 level) with only approximately a 7% increase in latency. This demonstrates that our lightweight DReranker effectively enhances the quality of demonstration selection while maintaining a good balance between performance enhancement and efficiency.

5.10 Case Study

To more intuitively illustrate how DemoRank can select more effective few-shot demonstrations by considering dependencies among them, we select a relevant query-passage pair from the DL19 dataset and use both EPR and DemoRank to choose 3-shot demonstrations. As shown in Table 10,

for the test query "When was the Salvation Army founded?," EPR only retrieves demonstrations most relevant to "Salvation Army," while our DemoRank could select more diverse demonstrations that not only involved "Salvation Army" but also included queries starting with "when." It not only tells the LLM the background of the Salvation Army but also helps the LLM assess relevance by determining whether the passage contains time-related answers. Additionally, the labels of the demonstrations selected by DemoRank cover both "Yes" and "No," which better informs the LLM about what is relevant and what is not. Thus, the demonstrations of DemoRank could help LLM yield a higher relevance score, contributing to the final passage ranking.

6 Conclusion

In this article, we discuss the challenges of demonstration selection in ranking tasks and propose the DemoRank framework, which consists of a DRetriever and a dependency-aware reranker. Different from the DRetriever's training, which models demonstrations independently, the DReranker is trained using dependency-aware training samples, which are constructed using an efficient method. We also design a novel list-pairwise training approach for reranker optimization, which compares a pair of demonstration lists that differ only in the last demonstration. Experiments on various ranking datasets prove the effectiveness of DemoRank. Further analysis shows the effectiveness of each proposed component, the advantages compared to supervised models, performance on different demonstration numbers, generalization on unseen datasets, etc. In the future, we will conduct experiments with larger LLMs, such as those with 30B or even 70B parameters. We also intend to incorporate other passage ranking methods, such as listwise and pairwise approaches, to validate the generalizability of our few-shot demonstrations selection method comprehensively.

Acknowledgments

The work was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

References

[1] Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context examples selection for machine translation. In *Proceedings of the Findings of the Association for Computational Linguistics (ACL '23)*. Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.), Association for Computational Linguistics, 8857–8873. DOI: https://doi.org/10.18653/V1/2023.FINDINGS-ACL.564

[2] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference (ICML '05)*. Luc De Raedt and Stefan Wrobel (Eds.), ACM, Vol. 119. 89–96. DOI: https://doi.org/10.1145/1102351.1102363

[3] Yiqun Chen, Qi Liu, Yi Zhang, Weiwei Sun, Daiting Shi, Jiaxin Mao, and Dawei Yin. 2024. TourRank: Utilizing large language models for documents ranking with a tournament-inspired strategy. arXiv:2406.11678. Retrieved from http://arxiv.org/abs/2406.11678

[4] Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Weiwei Deng, and Qi Zhang. 2023. UPRISE: Universal prompt retrieval for improving Zero-Shot evaluation. In *EMNLP*. Association for Computational Linguistics, 12318–12337.

[5] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. arXiv:2210.11416. Retrieved from http://arxiv.org/abs/2210.11416

[6] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 deep learning track. arXiv:2102.07662. Retrieved from http://arxiv.org/abs/2102.07662

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees.2020. Overview of the TREC 2019 deep learning track. arXiv:2003.07820. Retrieved from http://arxiv.org/abs/2003.07820

[8] Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP '21)*. Marie-Francine Moens,

Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.), Association for Computational Linguistics, 9594–9611. DOI: https://doi.org/10.18653/V1/2021.EMNLP-MAIN.755

[9] Andrew Drozdov, Honglei Zhuang, Zhuyun Dai, Zhen Qin, Razieh Rahimi, Xuanhui Wang, Dana Alon, Mohit Iyyer, Andrew McCallum, Donald Metzler, et al. 2023. PaRaDe: Passage ranking using demonstrations with LLMs. In *Proceedings of the Findings of the Association for Computational Linguistics (EMNLP '23)*. Houda Bouamor, Juan Pino, and Kalika Bali (Eds.), Association for Computational Linguistics, 14242–14252. DOI: https://doi.org/10.18653/v1/2023.findings-emnlp.950

[10] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net

[11] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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 (2019), 452–466.

[12] Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL '23)*, Vol. 1. Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.), Association for Computational Linguistics, 1401–1422. DOI: https://doi.org/10.18653/V1/2023.ACL-LONG.78

[13] Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 4644–4668.

[14] Xiaonan Li and Xipeng Qiu. 2023. MoT: Pre-thinking and recalling enable chatGPT to self-improve with memory-of-thoughts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

[15] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. arXiv:2211.09110. Retrieved from http://arxiv.org/abs/2211.09110

[16] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of the Deep Learning Inside Out*. Association for Computational Linguistics, 100–114.

[17] Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2024. Leveraging passage embeddings for efficient listwise reranking with large language models. arXiv:2406.14848. Retrieved from http://arxiv.org/abs/2406.14848

[18] Tie-Yan Liu. 2009. *Learning to Rank for Information Retrieval*, Vol. 3. Foundations and Trends® in Information Retrieval, 225–331.

[19] Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. 2025. ReasonRank: Empowering passage ranking with strong reasoning ability. arXiv:2508.07050. Retrieved from http://arxiv.org/abs/2508.07050

[20] Wenhan Liu, Xinyu Ma, Yutao Zhu, Lixin Su, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025. CoRanking: Collaborative ranking with small and large ranking agents. arXiv:2503.23427. Retrieved from http://arxiv.org/abs/2503.23427

[21] Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2024. Sliding windows are not the end: Exploring full ranking with long-context large language models. arXiv:2412.14574. Retrieved from http://arxiv.org/abs/2412.14574

[22] Wenhan Liu, Yujia Zhou, Yutao Zhu, and Zhicheng Dou. 2024. How to personalize and whether to personalize? Candidate documents decide. *Knowledge and Information Systems* 66, 9 (2024), 5581–5604.

[23] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 8086–8098.

[24] 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. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches 2016 Co-Located with the 30th Annual Conference on Neural Information Processing Systems (NIPS '16)*, Vol. 1773. Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne (Eds.), Retrieved from CEUR-WS.org. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[25] Rodrigo Frassetto Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. arXiv:1901.04085. Retrieved from http://arxiv.org/abs/1901.04085

[26] Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Proceedings of the Findings of the Association for Computational Linguistics (EMNLP '20)*, Trevor Cohn, Yulan He, and Yang Liu (Eds.), Association for Computational Linguistics, 708–718. DOI: https://doi.org/10.18653/v1/2020.findings-emnlp.63

[27] Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR '22)*. OpenReview.net. Retrieved from https://openreview.net/forum?id=KmtVD97J43e

[28] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankVicuna: Zero-shot listwise document reranking with open-source large language models. arXiv:2309.15088. Retrieved from https://arxiv.org/abs/2309.15088

[29] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. arXiv:2306.17563. Retrieved from https://arxiv.org/abs/2306.17563

[30] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. arXiv:1908.10084. Retrieved from https://arxiv.org/abs/1908.10084

[31] Stephen E. Robertson and Hugo Zaragoza. 2009. *The Probabilistic Relevance Framework: BM25 and Beyond*, Vol. 3. Foundations Trends in Information Retrieval, 333–389.

[32] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the NAACL-HLT*. Association for Computational Linguistics, 2655–2671.

[33] Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP '22)*. Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.), Association for Computational Linguistics, 3781–3797. DOI: https://doi.org/10.18653/v1/2022.emnlp-main.249

[34] Jimmy Bai, Peng Shi, Rui Zhang, HeLin. 2022. XRICL: Cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-SQL semantic parsing. In *Proceedings of the Findings of the Association for Computational Linguistics (EMNLP '22)*. Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.), Association for Computational Linguistics, 5248–5259. DOI: https://doi.org/10.18653/V1/2022.FINDINGS-EMNLP.384

[35] Weiwei Sun, Zheng Chen, Xinyu Ma, Lingyong Yan, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren.2023. Instruction distillation makes large language models efficient zero-shot rankers. arXiv:2311.01555. Retrieved from https://arxiv.org/abs/2311.01555

[36] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren.2023. Is ChatGPT good at search? Investigating large language models as Re-Ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP '23)*. Houda Bouamor, Juan Pino, and Kalika Bali (Eds.), Association for Computational Linguistics, 14918–14937. Retrieved from https://aclanthology.org/2023.emnlp-main.923

[37] 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 (NAACL-HLT '18)*, Vol. 1. Marilyn A. Walker, Heng Ji, and Amanda Stent (Eds.), Association for Computational Linguistics, 809–819. DOI: https://doi.org/10.18653/V1/N18-1074

[38] 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:2212.03533. Retrieved from https://arxiv.org/abs/2212.03533

[39] Liang Wang, Nan Yang, and Furu Wei. 2023. Learning to retrieve in-context examples for large language models. arXiv:2307.07164. Retrieved from https://arxiv.org/abs/2307.07164

[40] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. In *Proceedings of the Transactions on Machine Learning Research*. Retrieved from https://openreview.net/forum?id=yzkSU5zdwD

[41] Zhe Xu, Daoyuan Chen, Jiayi Kuang, Zihao Yi, Yaliang Li, and Ying Shen. 2024. Dynamic demonstration retrieval and cognitive understanding for emotional support conversation. arXiv:2404.02505. Retrieved from https://arxiv.org/abs/2404.02505

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

[43] Soyoung Yoon, Gyuwan Kim, Gyu Hwung Cho, and Seung-Won Hwang.2025. AcuRank: Uncertainty-aware adaptive computation for listwise reranking. arXiv:2505.18512. Retrieved from https://arxiv.org/abs/2505.18512

[44] Soyoung Yoon, Jongho Kim, Daeyong Kwon, Avishek Anand, and Seung-Won Hwang.2025. 2025. On listwise reranking for corpus feedback. arXiv:2510.00887. Retrieved from https://arxiv.org/abs/2510.00887

[45] Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active example selection for in-context learning. In *Proceedings of the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 9134–9148.

[46] 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:2308.07107. Retrieved from https://arxiv.org/abs/2308.07107

[47] Yutao Zhu, Peitian Zhang, Chenghao Zhang, Yifei Chen, Binyu Xie, Zhicheng Dou, Zheng Liu, and Ji-Rong Wen. 2024. INTERS: Unlocking the power of large language models in search with instruction tuning. arXiv:2401.06532. Retrieved from https://arxiv.org/abs/2401.06532

[48] Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky.2023. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. arXiv:2310.14122. Retrieved from https://arxiv.org/abs/2310.14122

[49] Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon.2023. Open-source large language models are strong zero-shot query likelihood models for document ranking. In *Proceedings of the Findings of the Association for Computational Linguistics (EMNLP '23)*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.), Association for Computational Linguistics, 8807–8817. Retrieved from https://aclanthology.org/2023.findings-emnlp.590

Received 22 April 2025; revised 16 September 2025; accepted 26 December 2025