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

Anonymous ACL submission

Abstract

Recently, there has been increasing interest in applying large language models (LLMs) as zero-shot passage rankers. However, few studies have explored how to select appropriate in-context demonstrations for the passage ranking task, which is the focus of this paper. Previous studies mainly apply a demonstration retriever to retrieve demonstrations and use top-k demonstrations for in-context learning (ICL). Although effective, this approach overlooks the dependencies between demonstrations, leading to inferior performance of few-shot ICL in the passage ranking task. In this paper, we formulate the demonstration selection as a *retrieve-then-rerank* process and introduce the DemoRank framework. In this framework, we first use LLM feedback to train a demonstration retriever and construct a novel dependency-aware training samples to train a demonstration reranker to improve few-shot ICL. The construction of such training samples not only considers demonstration dependencies but also performs in an efficient way. Extensive experiments demonstrate DemoRank's effectiveness in in-domain scenarios and strong generalization to out-of-domain scenarios.

1 Introduction

Large language models (LLM) 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 (Zhuang et al., 2023a; Sun et al., 2023; Qin et al., 2023). A typical approach is relevance generation, which judges the relevance of a query-passage pair in a pointwise manner. This method prompts LLMs to assess the relevance of a passage to a query by generating responses such as "Yes" or "No". The relevance score is then computed based on the log-likelihood of these responses. This approach has been demonstrated to be effective in previous studies (Zhuang et al., 2023a; Liang et al., 2022).

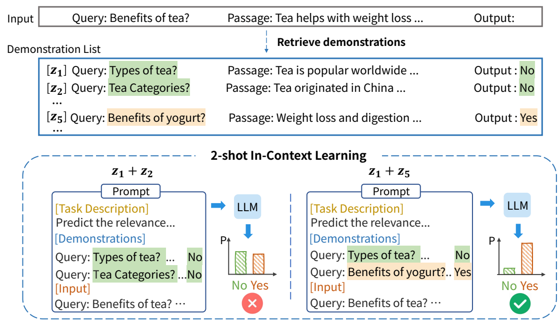

Figure 1: Compared with choosing top-2 demonstrations (z_1 and z_2), the combination of z_1 and z_5 provides richer and more diverse query-passage relationships, thus yielding better relevance assessment.

In-context learning (ICL) has been proved as an emergent ability of LLMs (Wei et al., 2022), enabling them to adapt to specific tasks through several task demonstrations (*i.e.*, input-output examples). Many studies have investigated the optimal selection of demonstrations for NLP tasks (Lu et al., 2022; Zhang et al., 2022; Li et al., 2023; Wang et al., 2023; Xu et al., 2024), highlighting the importance of tailored demonstrations in achieving high performance. However, the application of ICL to passage ranking tasks has not been extensively studied. Given the complex nature of passage ranking, ICL presents a challenging yet promising opportunity to enhance LLMs' performance. Consequently, this study aims to develop effective demonstration selection strategies to optimize the application of ICL in passage ranking.

A widely-used and effective approach for demonstration selection is training a demonstration retriever using LLM' feedback (Wang et al., 2023; Rubin et al., 2022; Li et al., 2023; Cheng et al., 2023; Scarlatos and Lan, 2023; Luo et al., 2023). This approach first utilizes an LLM to score some demonstration candidates based on LLM's likelihood of producing the correct output given each candidate and the input, and choose positive and negative candidates based on scores for retriever training. Following this technique line, we propose to train a demonstration retriever based on LLM's feedback tailored for passage ranking task.

In the inference stage, a common practice (Wang et al., 2023) is to use the trained retriever to obtain a list of demonstrations and concatenate the top-retrieved ones together in the prompt for ICL. Despite its effectiveness in NLP tasks, directly extending it into the passage ranking task may result in sub-optimal performance. The main challenge lies in the complex nature of the query-passage relationship in passage ranking, which may require a *combination* of multiple demonstrations to provide effective information for understanding such a relationship. Figure 1 shows an example of such a problem. When selecting 2-shot demonstrations for the current input (a relevant query-passage pair), existing methods (Wang et al., 2023; Rubin et al., 2022) will choose the top-2 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 opposite outputs (relevance label), which provide LLM with *richer and more diverse query-passage relationship signals*, thus contributing more to the relevance assessment. This example shows the insufficiency of pure relevance-based demonstration selection in the few-shot LLM-based passage ranking task. In this paper, we transform the problem of selecting the optimal k-shot demonstrations from initially retrieved n demonstrations into a demonstration ranking problem and propose to use LLM's feedback to train a novel dependency-aware demonstration reranker, making the top-ranked ones more suitable in the few-shot ICL for passage ranking.

Nevertheless, training such a reranker is a very challenging task. As previously mentioned, it is unreasonable to use LLM's feedback on each individual demonstration for training a reranker designed for k-shot selection, because demonstrations can influence each other. Additionally, constructing the ground truth ranking of a reranker tailored for k-shot selection requires finding the optimal k-shot permutation from the retrieved n demonstrations. Theoretically, this requires using LLM to score total $\frac{n!}{(n-k)!}$ demonstration permutations, which is highly time-consuming and impractical. To overcome these challenges, we propose to construct a kind of dependency-aware training samples (a list of demonstrations with ranking labels) for reranker training. Specifically, given a retrieved demonstration set, we greedily select demonstrations from the set and annotate them with different ranking labels (from highest to lowest). Each time, the demonstration that maximizes the LLM's feedback when concatenated with the already selected ones is chosen. This process not only considers the dependencies between current demonstration and previously selected ones, but also greatly reduces the number of LLM inferences.

To this end, we propose DemoRank, a **Demo**nstration selection framework for passage **Rank**ing, using a two-stage "retrieve-then-rerank" strategy. In this framework, we first train a demonstration retriever DRetriever based on LLM's feedback for the ranking task. Then, we introduce a dependency-aware demonstration reranker DReranker to rerank the retrieved demonstrations. To address the challenges of its training, we propose a method to construct dependency-aware training samples that not only incorporates demonstration dependency but is also time-efficient.

Experiments on a series of ranking datasets prove the effectiveness of DemoRank, especially in few-shot ICL. Further analysis also demonstrates the contribution of each proposed component and DemoRank's strong ability under different scenarios, including limited training data, different demonstration numbers, unseen datasets, etc.

The main contributions of our paper are summarized as follows: (1) To the best of our knowledge, we are the first to comprehensively discuss effective demonstration selection in passage ranking and propose DemoRank framework. (2) We propose a novel dependency-aware demonstration reranker and design a rational and efficient method for constructing its training data. (3) Besides in-domain performance, further experiments also demonstrate DemoRank's generalization on unseen datasets.

2 Related Work

2.1 LLM for Passage Ranking

With the development of large language models (LLMs) in information retrieval (Zhu et al., 2023), there have been many studies exploring how to utilize LLMs for the passage ranking task. In general, these studies can be divided into three categories: pointwise (Liang et al., 2022; Sachan et al., 2022), pairwise (Qin et al., 2023), and listwise methods (Sun et al., 2023; Ma et al., 2023). Pointwise methods assess the relevance between a query and a single passage. A typical approach is relevance generation (Liang et al., 2022; Zhuang et al., 2023a), 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". Another approach of pointwise methods is query generation (Sachan et al., 2022; Zhuang et al., 2023b), which calculates relevance score based on the log-likelihood of generating the query based on the passage. Pairwise methods compare two passages at a time and determine their relative relevance to a query, and listwise methods directly rank a passage list.

Despite promising results, these studies only focus on the zero-shot scenarios, with less emphasis on how to select effective demonstrations in few-shot scenarios. Manually written or rule-based selection (Drozdov et al., 2023) is inflexible for ranking tasks. In this paper, we explore more effective demonstration selection approaches for ranking tasks. Previous studies (Zhu et al., 2024) have revealed that relevance generation of the pointwise method is the most suitable method for passage ranking on open-source LLMs compared with other methods. Thus, we intend to use the relevance generation approach for passage ranking in this paper.

2.2 Demonstration Retrieval

A widely used demonstration selection approach is demonstration retrieval. Prior studies have explored using different retrievers for demonstration retrieval, which can be divided into two categories. One is utilizing off-the-shelf retrievers such as BM25 (Agrawal et al., 2023) or dense retriever (Liu et al., 2022). The other is to train a demonstration retriever using task-specific signals. For example, Rubin et al. (2022) propose to distill the LLM's feedback to a dense retriever EPR for the semantic parsing task. Li et al. (2023) and Wang et al. (2023) propose to train the retriever iteratively on various NLP tasks. However, a common issue with these methods is that they directly choose the top-retrieved demonstrations, which may include redundant information and contribute little to the LLM's understanding of relevance. In this paper, we take the demonstration dependencies into account and introduce a framework that first retrieves a list of demonstrations and then reranks in a dependency-aware manner, better aligning with the few-shot ICL in the ranking task.

3 Preliminaries

3.1 Relevance Generation for Ranking Task

Passage ranking aims to rank a list of retrieved passages based on their relevance to a query. Formally, given a query q and a passage list $[p_1, \ldots, p_n]$, our task is to compute a relevance score $S(q, p_i)$ for each passage. In the LLM-based relevance generation methods (Liang et al., 2022; Zhuang et al., 2023a), 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:

$$Rs(q, p_i) = \text{Pr}(\text{“Yes”}|T, q, p_i), \qquad (1)$$

where T is the task description. Finally, the passages are ranked based on the relevance score $S(q, p_i)$ in descending order.

3.2 In-context Learning in Ranking Task

In-context learning is a technique that inserts a few demonstrations into the prompt to help LLMs perform a task without updating parameters. 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, the relevance score $Rs(q, p_i)$ could be calculated by:

$$Rs(q, p_i) = \text{Pr}(\text{“Yes”}|T, \{z_i\}_{i=1}^k, q, p_i), \qquad (2)$$

where T is the task description, which is used in ICL to help LLMs understand the task (Zhu et al., 2024; Li et al., 2023).

4 The DemoRank Framework

As shown in Figure 2, our DemoRank framework follows a process of demonstration retrieval followed by dependency-aware reranking. The demonstration retriever DRetriever is trained using the demonstration candidates scored by LLM and the demonstration reranker DReranker is trained based on our constructed dependency-aware training samples. In this section, we elaborate on our demonstration pool construction, the pipeline of training, and inference.

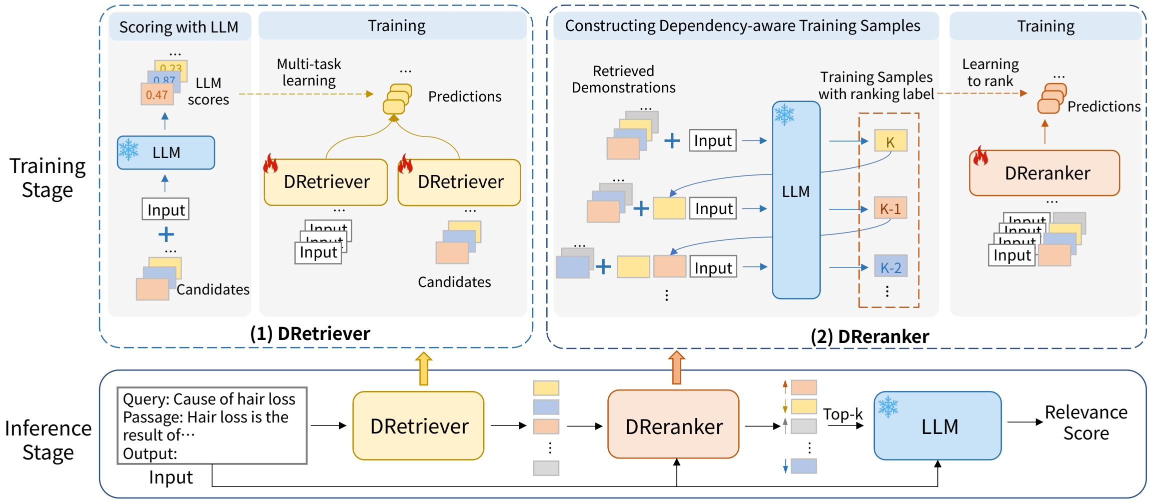

Figure 2: An overview of our proposed framework DemoRank. DemoRank comprises two main components: DRetriever and DReranker. We train the DRetriever using demonstration candidates scored by LLM and construct a kind of dependency-aware training samples to train the DReranker. During inference, a retrieve-then-rerank pipeline is performed and the top-k reranked ones are used for ICL.

4.1 Demonstration Pool Construction.

Given a passage ranking dataset (*e.g.*, MS MARCO (Nguyen et al., 2016)), 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 Demonstration Retriever DRetriever

In this part, we train DRetriever to retrieve potentially useful demonstrations for subsequent demonstration reranking. We apply an LLM to score a set of demonstration candidates to obtain supervised signals and use them to train the retriever through a multi-task learning strategy.

Scoring with LLM For a training input $I = (q, p)$ which contains a query-passage pair, we select a set of demonstrations from demonstration pool $\mathcal{P}$ as training candidates. Following previous studies (Wang et al., 2023), 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 (Drozdov et al., 2023). To include more potential useful demonstrations for training, we also randomly sample another 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 using the following equation:

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

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 paper, the LLM scorer uses the same model as the LLM passage ranker. Nevertheless, we also explored the transferability of LLM scorer on different LLM passage rankers (see Appendix C).

Training Our DRetriever is based on bi-encoder architecture. Given the current training input $I = (q, p)$ and a candidate z_i, we use encoder E_I and demonstration encoder E_z to encode them respectively and calculate the similarity score as:

$$S(I, z_i) = E_I(I)^\top E_z(z_i), \quad (4)$$

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

Then 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:

$$L_{\mathrm{c}} = -\log \frac{e^{S(I,z^+)}}{\sum_{z' \in Z} e^{S(I,z')}}, \quad (5)$$

where $Z = \{z^+, z_1^-, \dots, z_{N-1}^-\}$. Here we choose not to use in-batch negatives. The reasons are discussed in appendix D.

To make use of the fine-grained supervision of LLM's feedback, we also consider a ranking loss RankNet (Burges et al., 2005) to inject the ranking signal of candidates into training:

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

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 the weighted sum of L_c and L_r:

$$L = \lambda L_c + L_r, \quad (7)$$

where λ is a pre-defined hyper-parameter.

4.3 Demonstration Reranker DReranker

Previous studies (Wang et al., 2023; Rubin et al., 2022; Li et al., 2023) mainly use the top-k retrieved demonstrations for ICL which ignores the demonstration dependencies and could be sub-optimal for ranking tasks. To mitigate this issue, we formulate the selection of the optimal k-shot permutation from retrieved demonstrations into a demonstration reranking problem and construct a novel dependency-aware training samples in an efficient way for the reranker's training.

Constructing Dependency-aware Training Samples. To align with the aim of our DReranker, we propose constructing a dependency-aware training samples for training. Specifically, given a training input I, we use our trained DRetriever to retrieve top-M demonstrations Z^{r} from the demonstration pool. Then, we iteratively select demonstrations from Z^{r} and annotate each of them with a ranking label, as Figure 2 shows. In each iteration, we select, from the unselected demonstrations in Z^{r}, the one that maximizes the LLM's feedback when concatenated with already selected ones. Once a demonstration is selected, we append it to the training samples. This process considers previous demonstration sequence when selecting the current demonstration and approximates the optimal k-shot demonstration permutation incrementally, which is time-efficient and aligns with the few-shot setting. Note that as the number of iterations increases, the computational cost of LLM inference also increases. Due to limited computational resources, we set a maximum iteration number K. After the K-th iteration is completed, we annotate a ranking label from K to 1 to each demonstration in the training sample according to their selection order and annotate 0 to the unselected demonstrations in Z^{r}. Algorithm 1 shows this procedure.

Algorithm 1 Constructing dependency-aware training samples

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

```
Y ← {}, selected demonstrations S ← [].
Retrieve top-M demonstrations Z^r
for y = K to 1 do
    // y is the current ranking label.
    z* = argmax_{z_j ∈ Z^r \ S} f([S, z_j], I), using Equation (3)
    S ← [S, z*], Y ← Y ∪ {(z*, y)}
end for
for z_j in Z^r \ S do
    Y ← Y ∪ {(z_j, 0)}
end for
return Y
```

Training After constructing the dependency-aware training sample, we obtain a ranking label for each demonstration candidate in Z^{r}. We employ a cross-encoder model to train our DReranker. The model takes as input the concatenation of training input I and one candidate z_i with a "[SEP]" token and outputs a prediction score s_i using the representation of "[CLS]" token. Then we apply the RankNet loss function to optimize the reranker model, similar to Equation (6):

$$L_{\mathrm{r}} = \sum_{i,j}^{|Z^{\mathrm{r}}|} \mathbb{1}_{y_i > y_j} * \log(1 + e^{s_j - s_i}), \quad (8)$$

where y_i represents the ranking label of z_i. Note that our DReranker only receives an input and a single demonstration, without including dependent demonstrations, which may not fully capture the dependency-aware ranking labels. Nonetheless, this design saves inference time, making our DReranker more efficient. We plan to explore architectures that can model multiple dependent demonstrations efficiently in the future.

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 retrieve top-$M$ demonstrations using DRetriever and rerank them using our trained DReranker. Finally, we choose top-k reranked demonstrations as the in-context demonstrations and concatenate them with the test input to calculate the relevance score. We perform this process for all retrieved passages of q^{test} and rank these passages based on their relevance scores.

5 Experiments

5.1 Setting

Datasets In our experiments, we train and evaluate our DemoRank on diverse ranking datasets, including HotpotQA (Yang et al., 2018), NQ (Kwiatkowski et al., 2019), FEVER (Thorne et al., 2018) and MS MARCO (Nguyen et al., 2016). We use their training set to train our models respectively and evaluate the models on the corresponding test set (for MS MARCO, the evaluation is conducted on its development set as well as two in-domain datasets, TREC DL19 (Craswell et al., 2020b) and TREC DL20 (Craswell et al., 2020a)).

Implementation Details We use FLAN-T5-XL (Chung et al., 2022) as the frozen LLM for demonstration scoring and passage ranking unless otherwise specified. During the training stage, the number of demonstration candidates for retriever and reranker (N and M respectively) are both set as 50. And the maximum iteration number K in Section 4.3 is set as 4. During training, we apply e5-base-v2 (Wang et al., 2022) and DeBERTa-v3-base (He et al., 2023) to initialize our demonstration retriever and reranker respectively. Following previous studies (Sun et al., 2023; Zhuang et al., 2023a), we use the top-100 passages retrieved by BM25 as the passages to rerank. Due to the limited space, more implementation details on model training and inference are listed in Appendix A.

Baselines We compare our demonstration selection method with a series of baselines:

• **Random**: We randomly sample demonstrations from the demonstration pool $\mathcal{P}$ for each test input.

• **DBS** (Drozdov et al., 2023): DBS is a rule-based selection approach based on query generation in passage ranking. It selects the demonstrations which are the most difficult for the LLM to predict. In this paper, we implemented the algorithm based on the relevance generation approach. We define a score for each demonstration as the probability of the LLM generating the corresponding relevance label given a query and passage. The demonstrations with the lowest scores are applied.

• **K-means**: K-means is another static demonstration selection approach. This method clusters all the demonstrations in the pool into k clusters and then selects k demonstrations closest to each cluster center for ICL. We use the E5 (Wang et al., 2022) model to obtain the demonstration embeddings for clustering.

• **BM25** (Robertson and Zaragoza, 2009): BM25 is a widely-used sparse retriever. We apply BM25 to retrieve demonstrations that are most similar to the test query.

• **SBERT** (Reimers and Gurevych, 2019): We use Sentence-BERT as the off-the-shelf demonstration retriever following (Rubin et al., 2022)[1]. We use SBERT to encode all the demonstrations in the pool and retrieve the most similar demonstrations.

• **E5** (Wang et al., 2022): E5 is another off-the-shelf dense retriever. Following Wang et al. (2023), we use the same retrieval method as SBERT based on e5-base-v2 checkpoint[2].

5.2 Main Results

We compare DemoRank with baselines in 1-shot and 3-shot ICL respectively. Note that although DemoRank mainly focuses on few-shot scenarios, it can also work in 1-shot ICL, so we provide the performance of the 1-shot ICL as a reference. Table 1 shows the main results of our experiments. From the results, we draw the following observations: (1) Our framework DemoRank outperforms all the baselines significantly across all datasets. For example, in 3-shot ICL, DemoRank outperforms the second-best model E5 on HotpotQA by 3 points, and the second-best model BM25 on FEVER by about 7 points. It shows the DemoRank's powerful ability to select demonstrations. (2) When expanding from 1-shot to 3-shot, DemoRank shows greater improvement on Avg metric compared to other baselines, indicating that our DemoRank can better enhance the ICL performance in few-shot scenario. (3) The similarity-based demonstration selection baselines (*e.g.*, E5) outperform Random,

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

[2]https://huggingface.co/intfloat/e5-base-v2

	Method	HotpotQA	NQ	FEVER	DL19	DL20	MS MARCO	Avg
	Initial Order	63.30	30.55	65.13	50.58	47.96	22.84	46.73
	0-shot	60.65	48.62	38.92	66.13	65.57	33.24	52.19
1-shot	Random	59.71	48.69	38.41	66.76	65.35	33.53	52.08
	K-means	59.62	48.68	37.96	66.45	65.30	33.59	51.93
	DBS	60.34	49.05	38.96	66.83	65.79	33.54	52.42
	BM25	61.46	49.53	40.43	65.08	65.86	33.73	52.68
	SBERT	58.41	49.49	36.25	66.63	64.18	33.98	51.49
	E5	61.70	49.49	39.96	66.48	65.20	33.79	52.77
	DemoRank	**65.64**	**52.11**	**44.16**	**68.64**	**67.38**	**35.03**	**55.49**
3-shot	Random	59.42	48.61	38.61	66.57	64.84	33.70	51.96
	K-means	59.27	48.71	38.33	66.30	66.22	33.73	52.09
	DBS	60.15	48.62	39.00	66.40	65.21	33.61	52.17
	BM25	63.18	49.78	40.19	66.08	65.85	34.03	53.19
	SBERT	58.38	49.23	36.80	66.67	65.07	33.71	51.64
	E5	63.42	49.60	39.71	66.40	65.33	34.07	53.09
	DemoRank	**66.39**	**52.52**	**46.90**	**68.28**	**67.66**	**35.12**	**56.15**

Table 1: Main results (NDCG@10) on different datasets. The best results are marked in bold and the column Avg represents the average performance of all datasets. The Initial Order represents the order of the top-100 passages retrieved by BM25.

Method	NQ	DL19	FEVER	Avg
Ablation study				
- w/o DReranker	51.69	68.44	44.40	54.84
- w/o DTS	52.09	67.12	46.64	55.28
DemoRank	52.52	68.28	46.90	**55.90**
Using E5 as demonstration retriever				
E5	49.60	66.40	39.71	51.90
$\text{DemoRank}_{\text{E5}}$	50.74	67.37	41.76	**53.29**

Table 2: Results (NDCG@10) of different variants.

K-means, and DBS baselines, but still lags far behind DemoRank, which proves the effectiveness of task-specific finetuning based on LLM's feedback.

5.3 Analysis

In this section, we discuss different variants of DemoRank, compare DemoRank with supervised models, evaluate its performance on different demonstration numbers, and generalization on unseen datasets.

5.3.1 Different Variants of DemoRank

To understand the effectiveness of each component in DemoRank, we further evaluate different variants of DemoRank. We conduct the experiments on DL19, NQ sssand FEVER with 3-shot ICL, shown in Table 2. First, we remove our demonstration reranker DRanker and only use demonstrations retrieved by our demonstration retriever DRetriever, denoted as "- w/o DRanker". We can see that removing DRanker causes about 1 point drop, which indicates that the reranked demonstrations are more useful for ICL. Secondly, to further validate the effectiveness of our dependency-aware training samples DTS in few-shot ICL, we introduce another variant that score each retrieved demonstration in Z^r independently based on LLM, denoted as "- w/o DTS". Without considering the demonstration dependency, this variant lags behind DemoRank by 0.62 points, which proves that the dependency-aware training samples align more with the few-shot ICL. Thirdly, we also replace our trained DRetriever with E5 in our framework to validate the training effectiveness of our DReranker on different demonstration retrievers, denoted as $\text{DemoRank}_{\text{E5}}$. From the results, we can see that $\text{DemoRank}_{\text{E5}}$ significantly improves E5, which proves that our DReranker's training is flexible and not restricted by specific demonstration retriever. In addition, we also discuss the effectiveness of the ranking loss L_r and in-batch negatives during DRetriever's training in Appendix D.

5.3.2 Comparison with Supervised Reranker

The training of DemoRank is primarily based on queries in the training set, which can also be used to finetune a supervised model. In this part, we compare DemoRank with two supervised passage ranking models (monoBERT (Nogueira and Cho, 2019) and monoT5 (Nogueira et al., 2020)) under different quantities of training queries. Training details of monoBERT and monoT5 are provided in Appendix B. We choose MS MARCO as the training set and NDCG@10 as the metric. We also report the 0-shot performance as a reference. The results are shown in Table 4. We can see that when provided with 500K queries, although DemoRank

Method	Robust04	SCIDOCS	DBPedia	NEWS	FiQA	Quora	NFCorpus	Avg
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
0-shot	47.90	16.33	36.22	45.01	35.30	83.42	35.89	42.87
E5	46.49	16.78	37.72	45.40	35.38	**84.13**	35.44	43.05
DemoRank	**48.14**	**16.90**	39.76	**46.47**	**35.93**	83.96	**36.14**	**43.90**

Table 3: Results (NDCG@10) on BEIR. Best results are marked in bold. We use MS MARCO's demonstration pool for retrieval and 3-shot ICL for E5 and DemoRank.

QNum	Method	MS MARCO	DL19	DL20
0	0-shot	33.24	66.13	65.57
500K	monoBERT	39.97	**70.72**	67.28
	monoT5	**40.05**	70.58	67.33
	DemoRank	35.12	68.28	**67.66**
20K	monoBERT	30.69	63.61	59.32
	monoT5	29.79	61.16	52.72
	DemoRank	**34.63**	**67.25**	**66.67**

Table 4: Results (NDCG@10) on MS MARCO, DL19 and DL20. QNum represents the number of queries used in the MS MARCO training set.

slightly outperforms monoBERT and monoT5 on DL20, it still lags behind them on DL19 and MS MARCO, indicating the advantages of supervised models when abundant training data is available. However, when the number of queries is limited to 20K, DemoRank significantly outperforms the two supervised models on three datasets and also shows a significant improvement over 0-shot baseline. This suggests that when training data is limited, DemoRank is more effective than supervised models, highlighting the potential of DemoRank in low-resource scenarios.

5.3.3 Different Demonstration Numbers

Demonstration number is often considered a key factor affecting ICL. In this part, we discuss the performance of our models under different demonstration numbers. We compare DemoRank with E5 baseline on FEVER and NQ datasets, using NDCG@10 as the metric. we also compare with our DRetriever to better understand the performance of our DReranker. The results are shown in Figure 3. We can see that both DRetriever and DemoRank outperform E5 consistently across different demonstration numbers, proving the effectiveness and robustness of our models. Besides, we can observe that as the demonstration number increases, the gap between DemoRank and DRetriever becomes more obvious (especially on FEVER), proving the effectiveness of dependency-aware demonstration reranking in few-shot ICL.

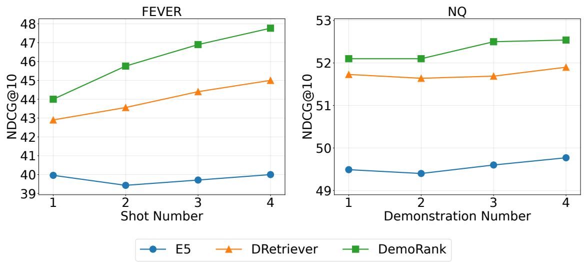

Figure 3: The impact of demonstration number.

5.3.4 Generalization on Unseen Datasets

One of the application scenarios of DemoRank is its generalization on unseen datasets. To prove this, we evaluate DemoRank trained on MS MARCO dataset on a series of BEIR datasets. We choose 0-shot, E5 demonstration retriever, and a supervised passage ranker MonoBERT (Nogueira and Cho, 2019), which is also trained on the MS MARCO dataset, for comparison. We use the demonstration pool from MS MARCO due to the lack of training sets in most BEIR datasets. As shown in Table 3, DemoRank outperforms the second-best model E5, by an average of about 1 point, proving its generalization ability. Furthermore, we also draw an interesting observation: despite using demonstrations from MS MARCO, DemoRank improves the 0-shot baseline across all datasets, indicating the potential of cross-dataset demonstrations in ICL.

6 Conclusion

In this paper, we explore how to select demonstrations for passage ranking task and propose DemoRank. We first trains a demonstration retriever with multi-task learning based on LLM's feedback. Then, an reasonable and efficient method is propose to construct dependency-aware training samples, serving as the training data of the demonstration reranker. 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, and generalization on BEIR, etc.

Limitations

In this paper, we introduce a novel demonstration selection framework DemoRank for passage ranking task. We acknowledge several limitations in this paper that present opportunities for future work. First, due to limited computational resources, we can not conduct experiments with larger LLMs, such as those with 30B or even 70B parameters. Second, our framework is limited to pointwise passage ranking and lacks discussion on how demonstrations can be selected in pairwise and listwise passage ranking, which can be a promising direction to explore.

References

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context examples selection for machine translation. In *ACL (Findings)*, pages 8857–8873. Association for Computational Linguistics.

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 *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 89–96. ACM.

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*, pages 12318–12337. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020a. Overview of the TREC 2020 deep learning track. In *TREC*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020b. Overview of the TREC 2019 deep learning track. *CoRR*, abs/2003.07820.

Andrew Drozdov, Honglei Zhuang, Zhuyun Dai, Zhen Qin, Razieh Rahimi, Xuanhui Wang, Dana Alon, Mohit Iyyer, Andrew McCallum, Donald Metzler, and Kai Hui. 2023. PaRaDe: Passage ranking using demonstrations with LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14242–14252, Singapore. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *ICLR*. OpenReview.net.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781. Association for Computational Linguistics.

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

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 *ACL (1)*, pages 4644–4668. Association for Computational Linguistics.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. Holistic evaluation of language models. *CoRR*, abs/2211.09110.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *DeeLIO@ACL*, pages 100–114. Association for Computational Linguistics.

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 *ACL (1)*, pages 8086–8098. Association for Computational Linguistics.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Seyed Mehran Kazemi, Chitta Baral, Vaiva

Imbrasaite, and Vincent Y. Zhao. 2023. Dr.icl: Demonstration-retrieved in-context learning. *CoRR*, abs/2305.14128.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *CoRR*, abs/2305.02156.

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 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rodrigo Frassetto Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *CoRR*, abs/1901.04085.

Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 708–718. Association for Computational Linguistics.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large language models are effective text rankers with pairwise ranking prompting. *CoRR*, abs/2306.17563.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

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

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 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3781–3797. Association for Computational Linguistics.

Alexander Scarlatos and Andrew S. Lan. 2023. Reticl: Sequential retrieval of in-context examples with reinforcement learning. *CoRR*, abs/2305.14502.

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 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, pages 809–819. Association for Computational Linguistics.

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

Liang Wang, Nan Yang, and Furu Wei. 2023. Learning to retrieve in-context examples for large language models. *CoRR*, abs/2307.07164.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022.

Zhe Xu, Daoyuan Chen, Jiayi Kuang, Zihao Yi, Yaliang Li, and Ying Shen. 2024. Dynamic demonstration retrieval and cognitive understanding for emotional support conversation. *CoRR*, abs/2404.02505.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380. Association for Computational Linguistics.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active example selection for in-context learning. In *EMNLP*, pages 9134–9148. Association for Computational Linguistics.

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

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. *CoRR*, abs/2401.06532.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2023a. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. *CoRR*, abs/2310.14122.

Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon. 2023b. Open-source large language models are strong zero-shot query likelihood models for document ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 8807–8817. Association for Computational Linguistics.

	Retriever Model	Reranker Model
Initialization	e5-base-v2	DeBERTa-v3-base
Optimizer	AdamW	AdamW
Learning Rate	3e-5	1e-5
Batch Size	8	8
Warmup Steps	400	400
Train Epochs	2	2
λ	0.2	-

Table 5: Hyperparameters for training the demonstration retriever and reranker model.

A Implementation Details of DemoRank

The hyper-parameters for training the demonstration retriever and reranker are shown in Table 5. For the construction of training input, we pair each query with one relevant passage and one irrelevant passage respectively, thus generating two training inputs. The passages labeled with 1 in the training set are used as relevant passages and the irrelevant ones are sampled from the top-100 passages retrieved by BM25. The number of queries used in each dataset is listed in Table 6. The maximum length of the queries and passages is set to 100 and 64, respectively.

During inference, for each test query-passage pair, we first use our DRetriever to retrieve top-50 demonstrations and then rerank them using our DReranker. The top-ranked demonstrations are used for ICL. The prompt we used 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 Table 7 and Table 8 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.

B Training Details of Supervised Models

For fair comparison with DemoRank, we construct the training data by pairing each query with one relevant passage and one irrelevant passage respectively. As for monoBERT (Nogueira and Cho, 2019), we start training from a bert-large-uncased model and use a binary classification loss to optimize the model. As for monoT5 (Nogueira et al., 2020), we initialize the model with T5-base model and finetune the model using generative loss. The training parameters are the same as the original paper.

	Query Number
FEVER	150K
NQ	150K
HotpotQA	150K
MS MARCO	200K

Table 6: The number of training queries for our framework DemoRank.

C Transferability across different LLM Ranker

In previous experiments, we used the same LLM (Flan-T5-XL) as the demonstration scorer and passage ranker. It is unknown whether the passage ranker could be replaced with other LLMs in the inference stage. In this section, we evaluate DemoRank's transferability across different LLM rankers on several datasets and compare with several baselines, including 0-shot, Random, K-means, BM25, and E5. We experiment with Flan-T5-XXL[3] (larger model size) and Llama-3-8B-Instruct[4] (different model architecture) and the results are shown in Table 9. From the results, we can draw the following observations: (1) DemoRank outperforms all the baselines on Avg metric when using both Flan-T5-XXL and Llama-3-8B-Instruct as the LLM rankers, proving its strong transferability across different LLM rankers. (2) We observe that when using Flan-T5-XXL as LLM Ranker, DemoRank yields higher performance on FEVER, DL19, and MS MARCO (49.56, 68.74 and 35.90 respectively), compared with Flan-T5-XL (46.90, 68.28 and 35.12 respectively in Table 1). This shows DemoRank's potential ability to improve passage ranking with larger-scale LLM rankers. (3) Comparing the overall 0-shot performance between Flan-T5-XL (see Table 1), Flan-T5-XXL and Llama-3-8B-Instruct, it is obvious that FlanT5 models perform better on average. This indicates that FlanT5 models are more suitable for passage ranking tasks, similar to findings from previous research (Zhuang et al., 2023b).

D Discussion on DRetriever's Training

In this part, we conducted experiments to verify the rationale of Dretriever's training. Firstly, we remove the ranking loss L_r (Equation (6)) from training (denoted as "- w/o L_r") and find a signifi-

[3]https://huggingface.co/google/flan-t5-xxl
[4]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

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 7: The instructions 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:

Table 8: The demonstration format used for different datasets.

	HotpotQA	NQ	FEVER	DL19	DL20	MS MARCO	Avg
Initial Order	63.30	30.55	65.13	50.58	47.96	22.84	46.73
Flan-T5-XXL							
0-shot	56.64	47.61	37.38	66.22	64.30	34.29	51.07
Random	58.31	48.51	39.56	67.47	65.31	35.15	52.39
K-means	58.75	48.86	39.37	67.40	65.47	35.24	52.52
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
DemoRank	**62.25**	**51.68**	**49.56**	**68.74**	**65.90**	**35.90**	**55.67**
Llama-3-8B-Instruct							
0-shot	55.93	36.24	27.53	58.47	55.10	28.09	43.56
Random	49.45	36.34	29.03	59.18	56.26	28.11	43.06
K-means	57.01	35.28	34.73	57.63	53.30	26.51	44.08
BM25	60.18	35.31	29.22	59.24	**57.28**	26.71	44.66
E5	60.09	**36.76**	28.64	57.36	53.09	26.98	43.82
DemoRank	**60.89**	35.47	**45.24**	**60.36**	56.45	**28.54**	**47.83**

Table 9: Results (NDCG@10) of different LLM ranker.

Method	NQ	DL19	FEVER	Avg
- w/o L_r	50.60	67.65	43.65	53.97
- w/ IBN	51.68	67.14	44.43	54.42
DRetriever	51.69	68.44	44.40	**54.84**

Table 10: The results (NDCG@10) of different training variants of DRetriever. We apply 3-shot ICL for each model.

cant performance degradation on the three datasets. This indicates that the ranking signal in demonstration candidates is useful for demonstration retriever training. Besides, as we mentioned in Section 4.2, we do not apply in-batch negatives when calculating the contrastive loss L_c, which is different from previous studies (Wang et al., 2023; Li et al., 2023; Karpukhin et al., 2020). To verify its rationale, we incorporate the in-batch negatives into the calculation of contrastive loss, denoted as "- w/ IBN". From the results, we can see that the in-batch negatives do not bring significant improvement and even harm the retriever's performance on DL19. This is because the utility of demonstrations in ranking tasks is not directly related to their similarity with the training input and the randomly sampled in-batch demonstrations may still contain valuable information and act as positive candidates, which is different from the assumption in passage retrieval (Karpukhin et al., 2020). Thus, directly using in-batch negatives may introduce additional noise into the training process.