Passage-specific Prompt Tuning for Passage Reranking in Question Answering with Large Language Models

Xuyang Wu*
Santa Clara University
Santa Clara, USA
xwu5@scu.edu

Zhiyuan Peng*
Santa Clara University
Santa Clara, USA
zpeng@scu.edu

Sravanthi Rajanala
Walmart Global Tech
Sunnyvale, USA
sravanthi.rajanala@walmart.com

Hsin-Tai Wu
Docomo Innovations
Sunnyvale, USA
hwu@docomoinnovations.com

Yi Fang
Santa Clara University
Santa Clara, USA
yfang@scu.edu

ABSTRACT

Effective passage retrieval and reranking methods have been widely utilized to identify suitable candidates in open-domain question answering tasks, recent studies have resorted to LLMs for reranking the retrieved passages by the log-likelihood of the question conditioned on each passage. Although these methods have demonstrated promising results, the performance is notably sensitive to the human-written prompt (or hard prompt), and fine-tuning LLMs can be computationally intensive and time-consuming. Furthermore, this approach limits the leverage of question-passage relevance pairs and passage-specific knowledge to enhance the ranking capabilities of LLMs. In this paper, we propose passage-specific prompt tuning for reranking in open-domain question answering (PSPT[1]): a parameter-efficient method that fine-tunes learnable passage-specific soft prompts, incorporating passage-specific knowledge from a limited set of question-passage relevance pairs. The method involves ranking retrieved passages based on the log-likelihood of the model generating the question conditioned on each passage and the learned soft prompt. We conducted extensive experiments utilizing the Llama-2-chat-7B model across three publicly available open-domain question answering datasets and the results demonstrate the effectiveness of the proposed approach.

ACM Reference Format:
Xuyang Wu, Zhiyuan Peng, Sravanthi Rajanala, Hsin-Tai Wu, and Yi Fang. 2018. Passage-specific Prompt Tuning for Passage Reranking in Question Answering with Large Language Models. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.
[1]Our code can be found at https://github.com/elviswxy/Gen-IR_PSPT.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Open-domain question answering (QA) involves to answer questions from a vast collection of passages [41]. The existing works [5, 12, 44] have demonstrated that efficiently retrieving a small subset of passages, which contain the answer to the question, is a crucial part of enhancing the QA task. Typically, relevant passages can be retrieved using keyword matching methods such as TF-IDF or BM25 [31], or through dense latent representations [5]. The results can be refined further by reranking the top-k retrieved passages to ensure accuracy.

Generative text reranker (GTR) resort to the model's generation ability to rerank the retrieved passages. By leveraging the powerful generation capabilities of Large Language Models (LLMs), GTR has demonstrated cutting-edge reranking performances, even directly output the permutation of input documents (or passages) based on their relevances to the given query (or question) [35]. Current GTR methods either fine-tune the whole large language model on question-passage relevance pairs [6, 19, 48] or rely on prompt engineering to craft good prompts for producing desirable output [20, 26, 32, 35]. While it is possible to fine-tune the whole LLMs like T0-3B [34], it becomes prohibitively computationally intensive and time-consuming on larger and advanced LLMs such as Llama-2 [19, 40]. On the other hand, the prompt engineering approach to LLMs saves cost but the results are highly sensitive to both the quality of human-written prompt (hard prompt) and the generation ability of LLMs. Moreover, hard prompting cannot benefit from the available question-passage relevance pairs of passage-specific knowledge.

In this paper, we propose passage-specific prompt tuning for passage reranking in open-domain question answering (PSPT), a parameter-efficient method that fine-tunes learnable passage-specific soft prompts on a limited set of question-passage relevance pairs. Specifically, our method involves integrating a soft prompt with a passage-specific embedding layer to form a new learnable prompt module, then concatenated with the embeddings of the raw passage to serve as new input for the LLMs. We maximize the log-likelihood of a query conditioned on the relevant or positive passage and utilize a hinge loss to punish the model when the log-likelihood of a query conditioned on the positive passage is smaller than that conditioned on the negative passage. Only a small proportion of learnable parameters θ are updated during the training

while LLM's parameters Φ are fixed. While recent research has successfully applied LLMs such as ChatGPT to reranking [35], most of the existing work has been built on proprietary models hidden behind opaque API endpoints, which may produce non-reproducible or non-deterministic experimental results [26]. Instead, our work is based on a popular open-source large language model (LLM): Llama-2 [40]. Our main contributions can be summarized as follows:

- To the best of our knowledge, this is the first work to enhance soft prompt tuning with passage-specific knowledge for passage reranking in QA tasks with LLMs.
- Our parameter-efficient tuning approach based on an open-source LLM substantially enhances reranking performance in QA with minimal parameter increments while preserving reproducibility.
- A comprehensive set of experiments is conducted on three widely used open-domain QA datasets. The results demonstrate that integrating a soft prompt module with a passage-specific embedding layer significantly enhances reranking performance beyond that of baseline retrievers and recently published models based on LLMs.

2 RELATED WORK

2.1 Passage Retrieval and Reranking

The QA systems utilize passage retrieval and reranking to select candidates. The unsupervised retrievers, like BM25 [31], MSS [33], and Contriever [9], and supervised retrievers, such as DPR [12], and MSS-DPR [33], retrieve a subset of candidate passages, which are then re-ranked in subsequent stages. BM25 measures the relevant scores between questions and passages by exact term matching. MSS is a dense retriever that undergoes joint training with both retriever and reader components. Contriever utilizes a contrastive learning framework for information retrieval (IR) pre-training, exhibits competitive performance relative to BM25. DPR is a dense passage retriever that uses the bi-encoder architecture, trained on question-passage relevance pairs. MSS-DPR presents a notable enhancement in DPR's efficacy through pre-training the dense retriever with the MSS method and applying DPR-style supervised fine-tuning. Besides these retrievers, recently, some new supervised methods are proposed like ColBERT [13], SPLADE [7] and SparseEmbed [14]. In this study we rerank the top-k passages retrieved by BM25, MSS, Contriever, DPR and MSS-DPR to show its effectiveness across different retrieval paradigms.

GTR converts a reranking task into a generation task to utilize LLMs. Such as, query generation [2, 6, 32, 49], relevance generation [18, 47, 48], and permutation generation [20, 26, 27, 35, 37, 42]. The query generation method computes the relevance score between a query and a document by the log-likelihood of LLMs to generate the query based on that document. UPR [32] calculates the query generation log-likelihood based on T0-3B [34] while dos Santos et al. [6] fine-tune GPT-2 [28] and BART [17] with unlikelihood loss and pairwise loss respectively and then computed the query generation log-likelihood. The relevance generation method [19, 35, 48] adopts the logit on certain word, like "yes", "no" or "\s", as the relevance score. The permutation generation methods prompts LLMs to directly output the ordered documents ranked by relevance. RankVicuna [26] to output the document order directly. Similarly, LRL [20] prompts GPT-3 [1] for ranking input documents. Our method is different from the existing GTR methods above in that we learn an efficient passage-specific prompt module on limited question-passage relevance pairs to enhance LLM's strong generation ability and guide the LLMs in the passage reranking of QA task.

2.2 Prompt Tuning

Prompt Tuning is a parameter-efficient technique that adapts pre-trained LLMs for specific tasks by adjusting the prompt module, rather than fine-tuning the entire model [16]. Some notable techniques include Prefix-Tuning [36], which prepends learnable embeddings to input tokens, "gisting" [22] which compresses prompts using a meta-learning approach, and task-specific prompt embeddings [16, 29] which incorporate task-specific information. SPTAR [25] optimizes a task-specific soft prompt to guide LLMs in tagging documents with weak queries, generating pairs that improve task-specific retriever performance. DCCP [46] leverages both prompt information and context to boost performance. ATTEMPT [43] transfers knowledge across tasks using a mixture of soft prompts. For Information Retrieval tasks, DPTDR [38] employs dual encoders and learnable soft prompts to enhance retrieval models. T5 [24], a text-to-text language model, models relevance in ranking tasks using document-query pairs and "true/false" label. Our method extends the soft prompt tuning approach from [16] by learning a passage-specific soft prompt, thereby enhancing the reranking capabilities of LLMs in QA tasks.

3 PASSAGE-SPECIFIC PROMPT TUNING

PSPT only fine-tunes a small number of parameters θ while keeping LLM's original parameters Φ fixed, learning a soft prompt and a set of embeddings in the training process. Subsequently, it reranks passages based on the log-likelihood of the question, conditioned on each retrieved passage along with the learned prompt. This method sets itself apart from the prompt tuning or soft prompt technique described in [36]. While the method in [36] employs a soft prompt that is consistent across all passages within the same dataset and varies only across different tasks or datasets, PSPT enriches this approach by incorporating passage-specific knowledge into the learned prompt, thereby boosting adaptability across a diverse range of passages. Consequently, the learnable prompt in PSPT dynamically adjusts not just to different tasks but also to individual passages.

Figure 1 illustrates the architecture of PSPT. The yellow blocks in Figure 1 are to learn a task-specific soft prompt e_1, and we follow [36] to initialize the soft prompt by extracting the LLM's original embeddings of prompt s "*please generate question for this passage*" which is repeated until the length of s equals pre-defined soft prompt length l_s. Suppose the dimensionality of the embeddings is dim; then the learned e_1 has shape $|l_s| \times dim$. Apart from the task-specific soft prompt, PSPT employs the red blocks in Figure 1 to learn a prompt e_2 with shape $|d_i| \times dim$ for a passage d_i. Instead of learning a new embedding layer with $V \times d$ weights for passages, inspired by LoRA [8], we decompose $V \times dim$ by the product of two low-rank matrices $|V| \times r$ and $r \times dim$ which are initialized by random Gaussian and zero respectively to reduce the

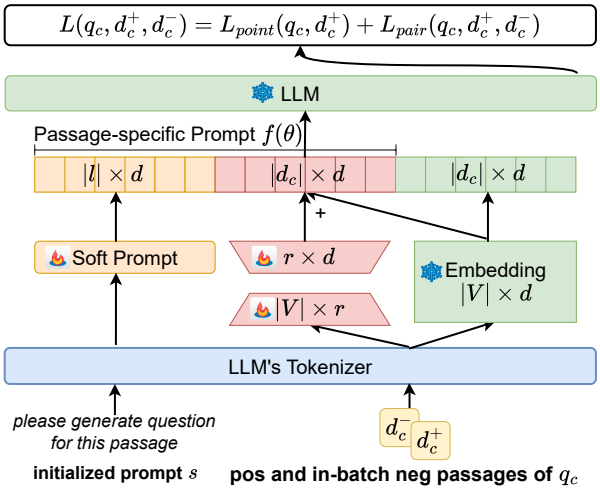

Figure 1: The architecture of the proposed PSPT. The original LLM parameters Φ (green blocks) are frozen during training, with only θ parameters (red and yellow blocks) updated.

number of learnable parameters. For a passage d_i, we first look up its embeddings $|d_i| \times r$ in learnable embedding layer $|V| \times r$ and then product it with $r \times dim$ to get e_3 with shape $d_i \times dim$. Finally, we obtain $e_2 = e_3 * (\alpha/r) + e_4$ where e_4 represents the embeddings of d_i encoded by LLM's original embedding layer and α helps to reduce the need to re-tune hyper-parameters when we vary r [45]. We concatenate e_1, e_2 and e_4 as the input of LLM to compute the log-likelihood of question q conditioned on passage d and passage-specific prompt $f_\theta(s, d)$ which is defined as:

$$I_{\theta,\Phi}(q|s, d) = \sum_{l=1}^{|q|} \log P_{\theta,\Phi}\left(q_l \mid q_{<l}, s, d\right) \tag{1}$$

where $P_{\theta,\Phi}\left(q_l \mid q_{<l}, s, d\right)$ represents the possibility of predicting the current token by looking at previous tokens. For a dataset of questions, each of which has some positive and negative passages, we follow [12] to sample one positive passage and one negative passage for each question and apply in-batch negative strategy to generate more negative passages. We can assume the training data is a collection of instances $\left\langle q_i, d_i^+, d_i^- \right\rangle_{i=1}^{i=N}$.

Pointwise loss is applied to constrain the model to output a ground-truth-like question based on the input positive or relevant passage, which is defined on one instance $\left\langle q_i, d_i^+, d_i^- \right\rangle$ as:

$$L_{point}(q_i, d_i^+) = -I_{\theta,\Phi}(q_i|s, d_i^+) \tag{2}$$

To improve the model's ranking ability, inspired by hinge loss, on one instance $\left\langle q_i, d_i^+, d_i^- \right\rangle$, we also apply a pairwise loss L_{pair} which is defined as:

$$L_{pair}(q_i, d_i^+, d_i^-) = \max\left\{0, I_{\theta,\Phi}(q_i|d_i^-, s) - I_{\theta,\Phi}(q_i|d_i^+, s)\right\} \tag{3}$$

Our final loss L directly combines the pointwise and pairwise losses:

$$L(q_i, d_i^+, d_i^-) = L_{point}(q_i, d_i^+) + L_{pair}(q_i, d_i^+, d_i^-) \tag{4}$$

During the inferencing process, the PSPT model reranks the top-k passages, denoted as $z_1, z_2, ...z_k$, which are retrieved by the retriever R. The relevance score for this reranking is based on the log-likelihood of the generated question q conditioned on each passage z_j along with the leaned passage-specific prompt. This is expressed as $I_{\theta^*,\Phi}(q|s, z_j)$, where s represents the initialized prompt and θ^* denotes the learned optimal parameters after training phase.

Dataset	Train	Ret.Train	Eval	Test
Natural Questions	79,168	58,880	8,757	3,610
TriviaQA	78,785	60,413	8,837	11,313
SQuAD	78,713	70,096	8,886	10,570

Table 1: The statistics of the datasets. The column Ret.Train refer to the actual questions used for training supervised retrievers after filtering in the dataset.

4 EXPERIMENTS

4.1 Datasets, Baselines and Evaluation Metrics

Following the existing work of DPR [12] and aiming for fair comparisons in QA tasks, our study utilized several widely used Open-domain question answering datasets including Natural Questions (NQ) [15], TriviaQA [10], and SQuAD [30]. The statistics of our datasets are presented in Table 1.

We selected the Unsupervised Passage Retrieval (UPR) approach as a competitive baseline model. UPR utilizes a pre-trained language model to estimate the probability of an input question conditioned on a retrieved passage. Specialized, we replace the pre-trained language model in UPR with Llama-2-chat-7B to examine its capabilities and performance, ensuring a fair comparison. Furthermore, by leveraging the instruction tuning strategy, we use a high-quality question-passage pair to guide the generation process in UPR, aiming to enhance its performance. We name this baseline UPR-Inst. In the Appendix A.3, we provide detailed descriptions of the instruct prompt formats used by the baseline models.

In our work, we employed Llama-2 [40]. Specifically, we utilized the fine-tuned Llama-2-Chat model with 7 billion parameters[2], which was fine-tuned using publicly available instruction datasets. We employed the top-k Recall (R@k) and Hit Rate (H@k) to evaluate the reranking performance.

4.2 Implementation Details

For the baseline models, we used the base configuration as specified in each respective paper. We implemented PSPT based on the publicly available prompt tuning package PEFT [21]. We choose hard prompt initialization s as "please generate question for this passage" with pre-defined soft prompt length $l_s = 50$. For hyper-parameters, $r = 1$ and $\alpha = 16$ are selected for learning passage-specific embedding e_2 in Figure 1. We fine-tuned the PSPT model on Nvidia A100 GPUs, using the bfloat16 [11] data type, across different training sample sizes ranging from 320 to 1280 for each dataset. The training involved a batch size of 4 and an in-batch hard negative sample size of 4. The learning rates for yellow blocks and red blocks in Figure 1 are set at 3e-2 and 3e-5, respectively, with each undergoing linear decay. A maximum of 20 fine-tuning epochs were executed,

[2]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

Retriever	NQ		SQuAD		TriviaQA	
	R@10	H@10	R@10	H@10	R@10	H@10
Unsupervised Retrievers						
BM25	22.01	49.94	26.33	49.67	22.85	62.77
+UPR	32.31	59.45	42.33	64.78	36.17	71.80
+UPR-Inst	31.75	58.86	42.04	64.65	36.37	71.59
+PSPT	†‡**36.89**	†‡**62.24**	†‡**46.04**	†‡**66.76**	†‡**42.63**	†‡**73.71**
MSS	19.19	51.27	20.28	42.51	19.97	60.52
+UPR	33.71	63.91	38.22	60.14	37.44	72.29
+UPR-Inst	33.35	63.19	37.77	59.76	38.01	72.70
+PSPT	†‡**37.95**	†‡**66.45**	†‡**41.80**	†‡**62.20**	†‡**44.30**	†‡**74.68**
Contriever	22.31	58.73	26.06	54.65	20.27	68.00
+UPR	32.38	67.12	41.25	69.06	32.58	75.86
+UPR-Inst	31.27	66.07	40.75	68.74	32.90	75.74
+PSPT	†‡**37.45**	†‡**70.42**	†‡**46.09**	†‡**72.16**	†‡**39.46**	†‡**78.03**
Supervised Retrievers						
DPR	38.74	74.54	25.68	51.42	27.93	76.50
+UPR	41.73	75.60	41.57	66.01	36.83	80.28
+UPR-Inst	40.28	74.46	41.51	65.94	36.88	80.19
+PSPT	†‡**45.73**	†‡**77.84**	†‡**45.27**	†‡**68.45**	†‡**42.53**	†‡**81.67**
MSS-DPR	37.47	77.48	33.25	65.85	25.54	79.15
+UPR	38.79	76.81	46.96	77.10	31.33	81.56
+UPR-Inst	37.16	75.35	46.88	76.93	31.44	81.68
+PSPT	†‡**43.02**	†‡**79.09**	†‡**51.23**	†‡**79.36**	†‡**36.24**	†‡**82.83**

Table 2: Comparative analysis of Top-10 Recall and Hit Rate (R@10 and H@10) across test datasets, highlighting the best results for each retriever in bold. The symbols † and ‡ indicate statistically significant improvements over basic retrievers and the UPR approach, respectively, determined by two-tailed t-tests with p-values < 0.01.

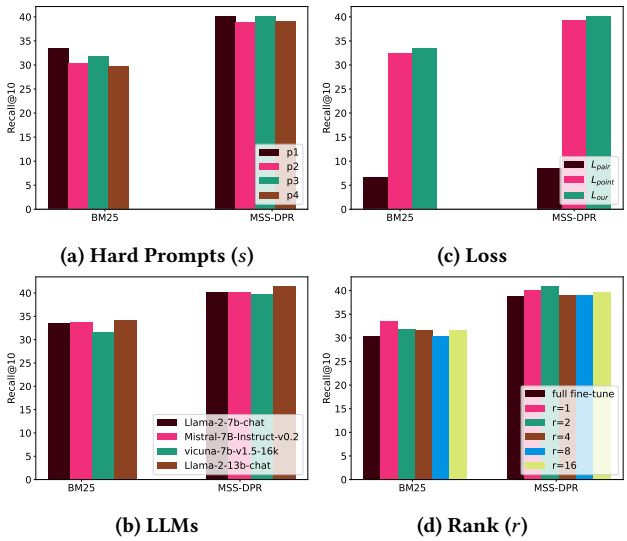

Figure 2: Performance analysis of key components in PSPT, evaluated using BM25 and MSS-DPR on a sampled NQ Dataset with Recall (R@10).

incorporating early stopping based on validation set performance. We have displayed additional results of hyper-parameter tuning in Appendix A.

4.3 Experimental Results

Table 2 presents a detailed evaluation of our proposed PSPT model's performance, showing that PSPT consistently surpasses both basic retrievers and baseline models. Essentially, our PSPT model achieves notable improvements across both unsupervised and supervised retrievers from three distinct datasets. This illustrates the powerful adaptability of our proposed model, capable of accommodating various potential datasets and retrieval environments. On the other hand, comparing the experimental results of unsupervised and supervised retrievers, firstly, our model can significantly enhance the reranking performance on the basis of the results from unsupervised retrievers. When the results from supervised retrievers are already good, the UPR-based model does not achieve consistent improvements in reranking performance, like on dataset NQ, UPR's H@10 is lower than that of MSS-DPR. In contrast, our model shows stable performance improvements across all datasets.

Additional findings are presented in Figure 2: (1) The PSPT module can continue to improve along with the enhancement of LLMs, such as the Llama-13B model with more parameters or the more powerful Mistral-7B model (in Figure 2b); (2) The soft prompt module demonstrates sensitivity to the initialization of hard prompts (in Appendix A.2), we selected only the best-performing initialization in our experiments (in Figure 2a). Furthermore, increasing the virtual prompt token length appropriately can provide additional space for the soft prompt module to adapt to new tasks during training (in Appendix A.1); (3) The passage-specific module effectiveness is sensitive to the extent of parameter changes. The LoRA-based technique offer a better alternative to full fine-tuning, as increase the rank r leads to slight worse performance. In addition, experiments using the LoRA-based approach consistently outperform the fully fine-tuning of the passage-specific module (in Figure 2d); (4) Using solely L_{point} or L_{pair} does not yield optimal results, as L_{pair} does not aim to optimize effective generation capabilities, and L_{point} is primarily optimized based only on positive sample data. Combining both losses enables the model to understand ranking orders and boosts generation effectiveness, which further improves model performance (in Figure 2c).

4.4 Ablation Study

We performed a detailed comparative analysis of various modules within PSPT to evaluate their efficiency and effectiveness. As shown in Table 3, for each experiment, we given the proportion of trainable parameters relative to Llama-2-chat-7B's total parameters to illustrate the fine-tuning process's efficiency. Our findings can be summary as: (1) All methods are capable of enhancing ranking performance of MSS-DPR and BM25 retrieval methods; (2) Converting hard prompts into trainable soft prompts enhances the performance; (3) Updates the embedding layer parameters of passage-specific module using the LoRA-based technique is more effective than fully fine-tuning of all parameters, regardless of the type of prompts used. However, this approach was slightly less effective than experiments using only soft prompts as more trainable parameters are needed.

Method	Trainable	BM25		MSS-DPR	
		R@10	H@10	R@10	H@10
Retriever Only	-	16.58	43.67	35.91	74.67
Hard Prompt Only (HP)	0.0000%	28.87	55.31	36.41	75.00
Soft Prompt Only (SP)	0.0007%	29.95	56.33	38.40	76.00
HP + passage-specific (FT)	1.9080%	29.34	55.67	36.81	73.67
HP + passage-specific (LoRA)	0.0005%	29.52	56.00	37.92	75.00
SP + passage-specific (FT)	1.9110%	30.35	55.33	38.86	75.67
SP + passage-specific (LoRA)	0.0012%	**33.44**	**59.33**	**40.03**	**76.33**

Table 3: Performance comparison of PSPT modules on BM25, MSS-DPR, evaluated on the sampled NQ Dataset using Recall (R@10) and Hit Rate (H@10), highlighting the best results in bold. Each experiment shows the percentage of trainable parameters relative to Llama-2's total parameters. FT and LoRA denote fully fine-tuning and LoRA-based tuning of the passage-specific module, respectively.

(4) Integrating the soft prompt module with the LoRA-based embedding layer configuration obtain the best performance.

5 CONCLUSION AND FUTURE WORK

In this paper, we present a parameter-efficient passage-specific prompt tuning approach to enhance LLMs for passage reranking in QA. Through extensive experimentation across three datasets, we demonstrate the effectiveness of our proposed PSPT. For future research, we will conduct experiments on more and larger datasets like MS MARCO[23], TREC2019[4], TREC2020[3] and BEIR[39] to thoroughly assess PSPT's generalizability in domains other than QA, and compare with more relevant state-of-the-art baseline models. Furthermore, since PSPT operates through a plug-in mechanism, adopting a learnable prompt module to enhance the probability of a frozen LLM generating true queries, we can combine our learnable prompt module with other methods, like LoRA [8] or more advanced ranking LLMs, like RankLLaMA [19], to further enhance ranking capabilities by fine-tuning the entire LLM.

REFERENCES

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

[2] Sukmin Cho, Soyeong Jeong, Jeongyeon Seo, and Jong C. Park. 2023. Discrete Prompt Optimization via Constrained Generation for Zero-shot Re-ranker. In *ACL*. Association for Computational Linguistics, 960–971.

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *CoRR* abs/2102.07662 (2021). arXiv:2102.07662 https://arxiv.org/abs/2102.07662

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *CoRR* abs/2003.07820 (2020). arXiv:2003.07820 https://arxiv.org/abs/2003.07820

[5] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[6] Cícero Nogueira dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through Ranking by Generation. In *EMNLP*. Association for Computational Linguistics, 1722–1727.

[7] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR*. ACM, 2288–2292.

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. https://openreview.net/forum?id=nZeVKeeFYf9

[9] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *Trans. Mach. Learn. Res.* 2022 (2022).

[10] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 1601–1611.

[11] Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. 2019. A Study of BFLOAT16 for Deep Learning Training. *CoRR* abs/1905.12322 (2019).

[12] 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*. Association for Computational Linguistics, 6769–6781.

[13] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*. ACM, 39–48.

[14] Weize Kong, Jeffrey M. Dudek, Cheng Li, Mingyang Zhang, and Michael Bendersky. 2023. SparseEmbed: Learning Sparse Lexical Representations with Contextual Embeddings for Retrieval. In *SIGIR*. ACM, 2399–2403.

[15] 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 (2019), 452–466.

[16] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *EMNLP*. Association for Computational Linguistics, 3045–3059.

[17] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*. ACL, 7871–7880.

[18] 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 (2022).

[19] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. *CoRR* abs/2310.08319 (2023).

[20] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-Shot List-wise Document Reranking with a Large Language Model. *CoRR* abs/2305.02156 (2023).

[21] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

[22] Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. 2023. Learning to Compress Prompts with Gist Tokens. *CoRR* abs/2304.08467 (2023).

[23] 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 (CEUR Workshop Proceedings, Vol. 1773)*, Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne (Eds.). CEUR-WS.org. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[24] Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *EMNLP (Findings of ACL, Vol. EMNLP 2020)*. Association for Computational Linguistics, 708–718.

[25] Zhiyuan Peng, Xuyang Wu, and Yi Fang. 2023. Soft Prompt Tuning for Augmenting Dense Retrieval with Large Language Models. *CoRR* abs/2307.08303 (2023).

[26] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. *CoRR* abs/2309.15088 (2023).

[27] 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 (2023).

[28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[30] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*. The Association for Computational Linguistics, 2383–2392.

[31] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (2009), 333–389.

[32] 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 *EMNLP*. Association for Computational Linguistics, 3781–3797.

[33] Devendra Singh Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. 2021. End-to-End Training of Neural Retrievers for Open-Domain Question Answering. In *ACL/IJCNLP*. Association for Computational Linguistics, 6648–6662.

[34] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[35] 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 *EMNLP*. Association for Computational Linguistics, 14918–14937.

[36] Weng Lam Tam, Xiao Liu, Kaixuan Ji, Lilong Xue, Xingjian Zhang, Yuxiao Dong, Jiahua Liu, Maodi Hu, and Jie Tang. 2022. Parameter-Efficient Prompt Tuning Makes Generalized and Calibrated Neural Text Retrievers. *CoRR* abs/2207.07087 (2022).

[37] Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2023. Found in the Middle: Permutation Self-Consistency Improves Listwise Ranking in Large Language Models. *CoRR* abs/2310.07712 (2023).

[38] Zhengyang Tang, Benyou Wang, and Ting Yao. 2022. DPTDR: Deep Prompt Tuning for Dense Passage Retrieval. In *COLING*. International Committee on Computational Linguistics, 1193–1202.

[39] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

[40] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023).

[41] Ellen M. Voorhees. 1999. The TREC-8 Question Answering Track Report. In *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999 (NIST Special Publication, Vol. 500-246)*. National Institute of Standards and Technology (NIST).

[42] Yuan Wang, Xuyang Wu, Hsin-Tai Wu, Zhiqiang Tao, and Yi Fang. 2024. Do Large Language Models Rank Fairly? An Empirical Study on the Fairness of LLMs as Rankers. *CoRR* abs/2404.03192 (2024).

[43] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogério Feris, Huan Sun, and Yoon Kim. 2023. Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning. In *ICLR*. OpenReview.net.

[44] Tomer Wolfson, Mor Geva, Ankit Gupta, Yoav Goldberg, Matt Gardner, Daniel Deutch, and Jonathan Berant. 2020. Break It Down: A Question Understanding Benchmark. *Trans. Assoc. Comput. Linguistics* 8 (2020), 183–198.

[45] Greg Yang and Edward J. Hu. 2021. Tensor Programs IV: Feature Learning in Infinite-Width Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 11727–11737. http://proceedings.mlr.press/v139/yang21c.html

[46] Jie Zhou, Le Tian, Houjin Yu, Zhou Xiao, Hui Su, and Jie Zhou. 2022. Dual Context-Guided Continuous Prompt Tuning for Few-Shot Learning. In *ACL*. Association for Computational Linguistics, 79–84.

[47] 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. *CoRR* abs/2310.14122 (2023).

[48] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. In *SIGIR*. ACM, 2308–2313.

[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 *EMNLP*. Association for Computational Linguistics, 8807–8817.

A APPENDIX

A.1 Pre-defined Soft Prompt Length

Soft Prompt Length (l_s)	BM25		MSS-DPR	
	R@10	H@10	R@10	H@10
Retriever Only	16.58	43.67	35.91	74.67
$l_s = 20$	30.10	57.33	36.94	72.00
$l_s = 30$	31.06	57.33	39.32	74.67
$l_s = 40$	30.34	55.00	39.08	75.00
$l_s = 50$	**33.44**	**59.33**	**40.03**	**76.33**
$l_s = 60$	32.27	57.33	39.20	75.33
$l_s = 80$	31.26	57.00	39.38	75.33
$l_s = 100$	29.75	56.00	39.95	76.00

Table 4: Performance comparison of PSPT modules on BM25, MSS-DPR, evaluated on the sampled NQ Dataset using Recall (R@10) and Hit Rate (H@10), highlighting the best results in bold. Each experiment demonstrates the impact of different pre-defined soft prompt virtual lengths on the experimental results, and we selected the best performance with $l_s = 50$.

A.2 Initialization of Hard Prompts

Name	Content
p1	Please generate question for this passage.
p2	Generate a question based on the content of this passage.
p3	Kindly craft a question based on the content provided in this passage.
p4	Craft questions based on the provided passage.

Table 5: Different initialization of hard prompts utilized in PSPT.

A.3 Instruction Prompt of Baselines

For the UPR model, we used a fixed hard prompt as the instruction prompt. For the UPR-inst model, based on the hard prompt of the UPR model, we have added high-quality question-passage pairs to guide the generation process of the UPR model. We select these high-quality question-passage pairs for each dataset based on the top 3 BM25 results, and these instructive question-passage pairs will not appear in the training and testing data. The data formatted as follows:

Baselines	Prompt Format
UPR	*Please generate question for this passage:* *Passage: [Passage]* *Question:*
UPR-inst	*Please generate question for this passage based on the example:* *Example:* *Passage: [Passage]* *Question: [Question]* *Passage: [document]* *Question:*

Table 6: Instruction Prompt of Baselines.

A.4 In-batch Hard Negative Sample Size

In-batch Negative Sampling	BM25	
	R@10	H@10
Retriever Only	16.58	43.67
2	31.37	58.33
4	**33.44**	**59.33**
8	28.54	53.33
16	26.4	54.33
32	20.9	47.67

Table 7: Performance comparison of PSPT modules on BM25, evaluated on the sampled NQ Dataset using Recall (R@10) and Hit Rate (H@10), highlighting the best results in bold. Each experiment demonstrates the impact of different in-batch hard negative sample sizes on the experimental results. We chose the best performance with an in-batch hard negative sample size of 4.

A.5 Different Training Samples

Training Sample Size	BM25		MSS-DPR	
	R@10	H@10	R@10	H@10
Retriever Only	16.58	43.67	35.91	74.67
80	29.11	57.00	36.15	72.67
160	29.70	55.33	37.61	72.00
320	**33.44**	**59.33**	40.03	76.33
640	32.30	58.00	39.43	75.00
1280	30.92	56.33	**40.10**	**76.67**

Table 8: Performance comparison of PSPT modules on BM25, MSS-DPR, evaluated on the sampled NQ Dataset using Recall (R@10) and Hit Rate (H@10), highlighting the best results in bold. Each experiment demonstrates the impact of different training sample sizes on the experimental results. Based on the model's training efficiency and effectiveness, we selected a training sample size of 320.