Soft Prompt Tuning for Augmenting Dense Retrieval with Large Language Models

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

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

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

ABSTRACT

Dense retrieval (DR) converts queries and documents into dense embeddings and measures the similarity between queries and documents in vector space. One of the challenges in DR is the lack of domain-specific training data. While DR models can learn from large-scale public datasets like MS MARCO through transfer learning, evidence shows that not all DR models and domains can benefit from transfer learning equally. Recently, some researchers have resorted to large language models (LLMs) to improve the zero-shot and few-shot DR models. However, the hard prompts or human-written prompts utilized in these works cannot guarantee the good quality of generated weak queries. To tackle this, we propose soft prompt tuning for augmenting DR (SPTAR[1]): For each task, we leverage soft prompt-tuning to optimize a task-specific soft prompt on limited ground truth data and then prompt the LLMs to tag unlabeled documents with weak queries, yielding enough weak document-query pairs to train task-specific dense retrievers. We design a filter to select high-quality example document-query pairs in the prompt to further improve the quality of weak tagged queries. To the best of our knowledge, there is no prior work utilizing soft prompt tuning to augment DR models. The experiments demonstrate that SPTAR outperforms the unsupervised baselines BM25 and the recently proposed LLMs-based augmentation method for DR.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; • **Computing methodologies** → **Natural language generation**.

KEYWORDS

Large Language Models, Dense Retrieval, Prompt Tuning

1 INTRODUCTION

Traditional informational retrieval (IR) methods like TF-IDF and BM25 [25] are based on token-level similarity matching and then suffer from lexical gap [1]. Inspired by the progress in deep learning, researchers have proposed to utilize neural networks to overcome the lexical gap. DR is such a kind of method based on neural networks. DR models like DPR [10] and Col-BERT [11] [27] encode each query or document to a dense vector, the dimensionality of which is determined by the neural networks. In practice, dense retrievers pre-compute the embeddings of documents and then on which build an approximate nearest neighbor (ANN) index for fast search. When a new query comes in, only its embedding is

computed and fed into the following ANN search system. Unlike TF-IDF and BM25, DR cares more about the similarity of the overall semantic meaning.

Even though neural retrieval mitigates the lexical gap, it still suffers from the challenge of lacking domain-specific training data. Some researchers have proposed to leverage transfer learning to tackle this challenge. To tackle this problem. Evidence [32] [6] shows that not all DR models and domains can benefit from transfer learning equally. Recently, LLMs like CPT-3 [4], LLaMA [33], and Vicuna [5] show the strong ability of zero-shot and few-shot learning. Instead of fine-tuning the LLMs on task-specific data, prompting concatenates the instructions for certain tasks (e.g., TL;DR translate to English) and a few corresponding examples as input and obtains the answers from the output of large language model (LLM). These kinds of human-written prompts are also called hard prompts. Researchers [28] recently have estimated that a good language classifier prompt is worth hundreds to thousands of extra data points. InPars [2] and PROMPTAGATOR [6] both utilize hard prompts to prompt the LLMs to tag the unlabeled documents with weak queries and then train task-specific retrievers. However, hard prompts have some drawbacks: a) It is not easy to find good hard prompts. Hard prompts must be hand-crafted by humans through trial and error, and sometimes intuition and luck are needed; b) Even with hand-crafted prompts, the downstream tasks still underperform tuned models. For instance, compared with the performance of fine-tuned T5-XXL [24] on SuperGLUE [35], GPT-3 175B few-shot gets a 17.5 points smaller score despite using 16 times more parameters [12].

Instead of utilizing humanly readable words as a hard prompt [22], soft prompt [12] [13] is a list of embeddings, unrecognizable to the human eye, appended to the input of the neural network. During the soft prompt tuning, the parameters of the LLM are frozen, and only the parameters associated with soft prompt are updated. Even though [12] and [13] both demonstrate that soft prompt outperforms the hard prompt, there is no work utilizing soft prompt tuning to augment DR. In this paper, we propose soft prompt tuning for augmenting DR (SPTAR). Specifically, for each task, we leverage soft prompt tuning to optimize the parameters associated with soft prompt on limited ground truth data and then prompt the LLMs to tag unlabeled documents with weak queries, yielding enough weak document-query pairs to train task-specific retrievers. Moreover, we find that even with the optimized soft prompt, the quality of generated weak queries is sensitive to the example document-query pairs in the prompt. So, we design a filter to select high-quality example document-query pairs in the prompt to further improve the quality of weak tagged queries as well as the DR tasks. Our main contributions can be summarized as follows:

- To the best of our knowledge, this is the first work that utilizes LLMs with soft prompt tuning for augmenting DR tasks.
- We introduce a novel soft prompt filter to select high-quality document-query pairs in the prompt to further improve the quality of generated weak data.
- Comprehensive experiments are conducted to demonstrate our approach outperforming BM25 and InPars [2].
- Experiments are based on the recent open-source LLMs, and we will make the code publicly available upon paper acceptance.

2 RELATED WORK

2.1 Dense Retrieval

DR converts the queries and documents into dense vectors on which the ANN index can be built for fast search. DPR [10] is a two-tower structure: one BERT model for queries and another for documents. For each query with one positive document and several negative documents, DPR measures the similarity between query embedding and document embeddings and then maximizes the log-likelihood of the positive passage. A variant of DPR is to utilize one BERT by concatenating query and document as input and extracting the query embedding and document embedding after the encoding. The query encoder and document encoder of ColBERT [11] [27] share the same BERT but utilize a different special token after "[CLS]" to distinguish query and document. Unlike DPR directly measures the similarity between query embedding and document embeddings, Col-BERT introduces a late interaction mechanism. Specifically, for each token in the query, ColBERT computes its similarity with all the tokens in the document and applies a maximum pooling on these similarity scores. The similarity score of a pair of query and document is the summarization of all the scores after the maximum pooling. Given a query with one positive document and one negative document, ColBERT is optimized by the pairwise softmax cross-entropy loss over the computed scores of the positive and negative documents. ANCE [39] is a bi-encoder trained on (query, positive document, negative document) tuples where the negative document is retrieved from an ANN built on the checkpoint of the last step. BM25CE [36] is a re-ranking-based DR. BM25CE first applies BM25 to retrieve documents and then employs the trained crossed-encoder to re-rank the retrieved documents. Our contribution is not to propose new dense retrievers but to propose a novel method to augment the existing dense retrievers.

2.2 LLMs in Dense Retrieval

Most of the existing researches in this field primarily concentrate on leveraging LLMs to improve DR tasks through various data generation techniques, including query generation [2, 3, 6, 7, 9, 26], relevance generation [14], and permutation generation [16, 23, 29]. InPars [2] feeds a task-specific human-written prompt and 3 example document-query pairs to a 6B GPT-3 [4] model Curie to generate 100K weak document-query pairs and selects the top 10K queries with respect to the probability of query q to augment the training data. InPars [2] employs the same dense retrieval model proposed in [21], which treats the retrieval as a sequence-to-sequence task by concatenating a query and a document as input to T5 mode and

outputs the relevance score. Like InPars [2], PROMPTAGATOR [6] also feeds a task-specific human-written prompt and at most 8 example document-query pairs to LLM to generate weak data. Instead of selecting the top weak queries by their probabilities, PROMPTA-GATOR first trains a filter on uncleaned document-query pairs to filter the weak queries by dropping the weak queries that cannot retrieve their paired documents in the Top-k retrieved documents. By repeating this process multiple times, the filter significantly improves the performance of a dual-encoder DPR retriever. Besides, PROMPTAGATOR [6] utilizes a much bigger LLM: a 175B model Flan [38] which cannot be accessed by most researchers. UPR [26] utilizes LLM as a zero-shot reranker to re-rank the passages retrieved by retrievers like BM25 and DPR. Given a query, for each retrieved passage, UPR utilizes a prompt "Please write a question based on this passage" to prompt a LLM and computes the average loglikelihood of the question tokens conditioned on the input document as the relevance score. Because of the expensive training of LLMs, all these works utilize LLMs as query generators instead of fine-tuning them. Improved variations of InPars [2], such as InPars-v2 [9] and InPars-Light [3], have been introduced to enhance the original methodology. HyDE [7] leverages LLMs to augment queries by generating hypothetical documents, effectively capturing relevance patterns for unsupervised retrieval. LRL [16] trains a listwise zero-shot re-ranker that leverages LLMs without task-specific supervised training. Unlike pointwise re-rankers, LRL considers all candidate documents to determine their relative ranking positions. Another approach involves instructional permutation generation [29], where the focus is on instructing LLMs to directly output permutations of passages. Permutation distillation techniques are employed to transfer the passage ranking capabilities of ChatGPT into a smaller, specialized ranking model. While these works utilize LLMs as query generators without fine-tuning, our SPTAR approach takes a different approach. We first perform soft prompt tuning to optimize task-specific soft prompts and then employ data filtering to enhance the quality of the generated weak data.

2.3 Prompt Tuning

Prompt tuning has demonstrated significant potential in adapting pre-trained LLMs to specific tasks by focusing on tuning the prompt module instead of fine-tuning the entire model [30]. Prefix-Tuning [13] introduces a prompt module with learnable parameters θ outputting embeddings which are prepended to the embeddings of other inputted tokens. This approach keeps the original training objective intact, while updating only the prefix parameters θ through gradient descent for each task. Another similar technique, referred to as "gisting" [19], compresses arbitrary prompts into a condensed set of virtual "gist" tokens using a meta-learning approach. Building upon T5 [24], Lester et al. [12] proposed a method where the learnable embeddings of a task-specific prompt are prepended to the encoder's output. The concatenated embeddings are then passed through the decoder to compute the training objective. This approach enables the model to incorporate task-specific information into the decoding process. Zhou et al. [40] introduced Dual Context-guided Continuous Prompt (DCCP), which employs soft prompt tuning using dual inputs: context-aware prompt and label-aware

context representations. This approach leverages both prompt information and contextual understanding to enhance the model's performance. The use of multi-task learning techniques can benefit prompt learning in various tasks. For instance, ATTEMPT proposed by Wang et al. [37] introduces a multi-task tuning method that transfers knowledge across different tasks through a mixture of soft prompts. In the context of Multilingual Information Retrieval, Huang et al. [8] explores a soft prompt decoding approach that treats retrieval in each language as a separate task while jointly modeling them to capture shared underlying structures. They use decomposable prompts in KD-SPD to model languages, highlighting that languages share common features and concepts despite their unique properties. Regarding IR tasks, DPTDR by Tang et al. [31] employs a dual-encoder, utilizing two RoBERTa models for retrieval. It initializes the dual-encoder through contrastive learning and appends learnable soft prompts for query and document. Both the dual-encoder and the learnable prompts are updated during the training process.

In contrast, even though we apply the same Prefix-Tuning [13] method to learn a task-specific soft prompt, we focus on improving the data augmentation for DR tasks, and we propose a novel soft prompt filter method to select high-quality example document-query pairs in the prompt to improve the DR tasks further. The whole augmentation pipeline makes our approach different from the current works.
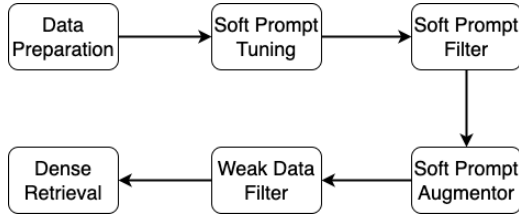
3 SPTAR

Figure 1: The pipeline of SPTAR.

As shown in Figure 1, SPTAR mainly consists of six modules: a) data preparation; b) soft prompt tuning; c) soft prompt filter; d) prompt augmentor; e) weak data filter; f) DR. In Section 3.1, we elaborate on how to generate the training and evaluation datasets of soft prompt tuning. With the training and evaluation datasets, we conduct soft prompt tuning (Section 3.2) to learn a task-specific soft prompt. To further improve the quality of the weak generated queries, we explore the soft prompt filter in Section 3.3 to find high-quality example document-query pairs for the learned task-specific soft prompt. We then prompt LLMs to generate weak queries for unlabeled documents, yielding enough training data to train DR (Section 3.4). Finally, we train the DR (Section 2.1) models on filtered weak data.

3.1 Data Preparation

We study how to augment DR with limited data, so the first step is to sample a small dataset on which we fine-tune a task-specific soft prompt. We define dataset D as $D = \{(q_n, d_n)\}_{n=1}^N$ where for each query q_n, there is a relevant document d_n. There may exist duplicated queries as one query may have multiple relevant documents. This domain-specific dataset D is categorized into train, test, and evaluation subsets, denoted as D_{train}, D_{test}, and D_{eval}, respectively. Apart from dataset D, there is a much bigger document collection C which contains all the documents in D but have more unlabeled documents denoted as $C_{unlabeled}$. After training, DR encodes all the documents in C into vectors. When new query comes in, DR encodes the query into an vector and search the top-k similar documents in vector space.

We random rample document-query pairs from origianl training dataset D_{train} to construct the training and evaluation datasets for the soft prompt module, namely S_{train}^X and S_{eval}^Y where X and Y represent the number of different queiers in training and evaluation datasets respectively. We define $Num(x)$ represent the number of document-query pairs when there are x different queries in the dataset, so, the number of document-query pairs in S_{train}^X is $Num(X)$. Similarly, S_{eval}^Y has $Num(Y)$ document-query pairs. For instance, in our experiment, we randomly sample 50 unique queries and their corresponding documents from the training dataset S_{train} to form $S_{train}^{50}(X = 50)$. From the remaining data in S_{train}, we randomly select 100 unique queries and their associated documents to compose $S_{eval}^{100}(Y = 100)$. The data in S_{train}^{50} primarily serves for optimizing the soft prompt, while the data in S_{eval}^{100} is employed to assess the model's convergence, enabling us to terminate the training process in advance and mitigate overfitting risks. We also tried other values of X, and the influence of X is studied in Section 5.2.5.
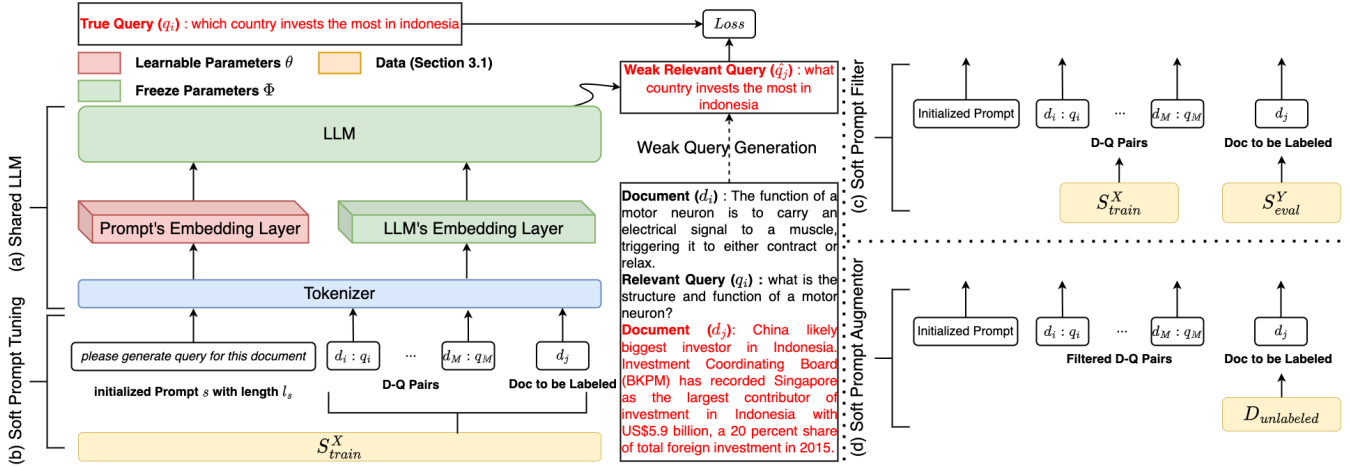
3.2 Soft Prompt Tuning

In our soft prompt tuning module, we apply the same prompt tuning method Prefix-Tuning [13] and we implemented our soft prompt tuning module based on a public prompt tuning package PEFT [18]. Figure 2 (b) illustrates the soft prompt tuning module, where the red boxes represent the parameters θ to be optimized during model training and the green boxes represent LLM's original parameters Φ that are retained. Let $f_{\theta_{emb}}(\cdot)$ denotes the prompt's embedding layer parameterd by θ_{emb}.

Document: d_1
Relevant Query: q_1
...
Document: d_M
Relevant Query: q_M
Document: d_j
Relevant Query:

Table 1: The format of c_j

For each training epoch, we first random sample M document-query pairs from training dataset S_{train}^X as example document-query pairs $(d_m, q_m)_{m=1}^M$, then iterate over the left document-query pairs $(d_j, q_j)_{j=1}^{Num(X)-M}$ to compute loss. As shown in Table 1, each pair (d_j, q_j) is concatenated with example pairs $(d_m, q_m)_{m=1}^M$ by keywords like "document" and "relevant query" as c_j. We initialize a

Figure 2: The main architecture of the proposed SPTAR: a) The same LLM is shared by soft prompt tuning module, soft prompt filter module and soft prompt augmentor module; b) soft prompt tuning module fixes the LLM's original parameters Φ and only fine-tune the parameters of soft prompt's embedding layer θ on the sampled small dataset (Section 3.1); c) soft prompt filter module fixes the learned parameters θ, and for each group of sampled example document-query pairs, computes the loss on evaluation dataset. The group of example document-qeury pairs with the smallest loss will be utilized in soft prompt augmentor module; d) with the learned parameters θ and a group of filtered example document-query pairs, soft prompt augmentor module iterates over the unlabeled document dataset $D_{unlabeled}$ to generate weak queries.

prompt s to be optimized with size l_s by repeating "please generate query for document" until the length of s equals l_s. Finally, we concatenate s with c_j as one training instance $t_j = [s; c_j]$ and there are $Num(X) - M$ instances in each epoch. Assume we have an autoregressive LLM denoted by $p_{\theta,\Phi}(q_j|t_j)$. When t_j is inputted into soft prompt tuning module, it is first tokenized into a list of ids z_j indexed by i then the embeddings of ids are extracted and fed into the following layers to compute the hidden vectors. For concise, we assume each token in t_j has one corresponding id in z_j. For training instance t_j, the hidden vector of ith time step is defined as $h_{j,i} \in \mathbb{R}^d$ where $h_{j,i} = \left[h_{j,i}^{(1)}; \cdots ; h_{j,i}^{(l)} \right]$ and l is the number of layers in LLM. An autoregressive LLM takes current $z_{j,i}$ and its left context to computes h_i as follows:

$$h_{j,i} = \begin{cases} \text{LLM}_{\theta,\phi}\left(z_{j,i}, h_{j,<i}\right), & \text{if } i \leq l_s \\ \text{LLM}_{\phi}\left(z_{j,i}, h_{j,<i}\right), & \text{if } i > l_s \end{cases} \quad (1)$$

The training objective function is:

$$\max_{\theta} \log p_{\theta,\phi}(q_j \mid t_j) = \max_{\theta} \sum_{i \in idx_{q_j}} \log p_{\theta,\phi}\left(z_{j,i} \mid h_{j,<i}\right) \quad (2)$$

where idx_{q_j} represent the indexes of d_j's ids and $p_{\theta,\phi}\left(z_{j,i} \mid h_{j,<i}\right)$ represent the possibility of next token the id of which is $z_{j,i}$. We utilize the negative log-likelihood and perplexity as loss L, which is defined as:

$$L = -\log p_{\theta,\phi}(q_j \mid t_j) + exp-\log p_{\theta,\phi}(q_j \mid t_j) \quad (3)$$

3.3 Soft Prompt Filter

During the training process of the soft prompt module, we observe that the choice of example document-query pairs $(d_m, q_m)_{m=1}^M$ significantly impacts the quality of text generation. Therefore, upon

completing the soft prompt training, with the learned parameters θ, we try to select the best group of document-query pair from S_{train}^X as example document-query pairs in soft prompt augmentor. For $M = 2$, there are 1475 (50 ∗ 59/2) groups of example pairs which are too many to evaluate them all efficiently. To reduce the computation complexity, we random sample X groups of example pairs from S_{train}^X to evaluate them on the evaluation dataset S_{eval}^Y and the group of example pairs with the best evaluation metric will be chosen as the example pairs in soft prompt augmentor. As shown in Figure 2 (c), the only difference between soft prompt tuning and soft prompt filter is the dataset where the d_j comes from. Suppose we sampled X groups of document-query pairs each of which has M document-query pairs $(d_m, q_m)_{m=1}^M$. Evaluation dataset S_{eval}^Y has $Num(Y)$ document-query pairs and for each pair (d_j, q_j), similar to Table 1, we concatenate d_j with the $(d_m, q_m)_{m=1}^M$ as c_j. Then, c_j is concatenated with the initialized prompt s as $t_j = [s, c_j]$. The evaluation metric is the same as the loss function L (Equation 3). We study the effectiveness of soft prompt filter in Section 5.2.3.

3.4 Soft Prompt Augmentor

As shown in Figure 2 (d), with the learned parameters θ and the filtered group of example document-query pairs, soft prompt augmentor generates a weak query for an unlabeled document d_j sampled from $D_{unlabeled}$. In practice, for each dataset, we create two weak datasets: a) W_{large}. 100K unlabled documents are sampled from $D_{unlabeled}$ to generate their weak queries. If the number of unlabeled documents in $D_{unlabeled}$ is smaller than 100K, all the unlabeled documents are utilized to generate weak queries; b) W_{small}. Sample 5000 document-query pairs from W_{large}. During the weak query generation process, LLM not only utilizes the soft prompt

Task	Domain	Dataset	Train	Eval	Test			Avg. Word Lengths	
			#Pairs	#Query	#Query	#Corpus	Avg. D/Q	Query	Document
Passage Retrieval	Misc.	MS MARCO [20]	532,761	–	6,980	8,841,823	1.1	5.96	55.98
Question Answering	Finance	FiQA-2018 [17]	14,166	500	648	57,638	2.6	10.77	132.32

Table 2: Statistics of datasets in BEIR benchmark. Avg. D/Q indicates the average number of relevant documents per query.

embeddings to capture domain-specific information but also bene-fits from the supplementary context provided by the best example document-query pairs.

3.5 Weak Data Filter

InPars [2] and PROPAGATE [6] demonstrated that it is necessary to clean the weak document-query pairs. We selected the same method as InPars [2] to clean the weak data. After we obtain generated weak document-query pairs (Section 3.4), we filter them by BM25: For each weak query, we run BM25 to retrieve the top k documents from the corpus C. We will drop the weak document-query pair if its paired document is not in the retrieved top k documents. For datasets MS MARCO and FiQA-2018, we tried top $k \in (10, 30, 50, 70)$. For BM25CE, we re-ranked top $k \in (5, 10, 30, 50, 70)$ documents retrieved by BM25. For all the models, we report the best NDCG@10 score along with other metrics in Table 6.

3.6 Dense Retrieval

We conducted the experiments on three popular dense retrievers: DPR (cross-encoder version), ColBERT, and BM25CE. The descrip-tions of the three models can be found in Section 2.1. We only trained DPR[2] and ColBERT[3]. For BM25CE, we directly loaded the BERT model of the best DPR as a re-ranker to re-rank the tok k documents retrieved by BM25.

4 EXPERIMENTAL SETUP

4.1 Datasets

Model	Train	Eval	Test
BM25	–	–	D_{test}
W/O Aug	$S_{train}^{50} + S_{eval}^{100}$	D_{eval}	D_{test}
InPars	$S_{train}^{50} + S_{eval}^{100} + F_k W_{large}$	D_{eval}	D_{test}
SPTAR-Tuning	S_{train}^{50}	S_{eval}^{100}	–
SPTAR-DR	$S_{train}^{50} + S_{eval}^{100} + F_k W_{large}$	D_{eval}	D_{test}

Table 3: Dataset partition for different methods.

We conducted experiments on two datasets MS MARCO [20] and FiQA-2018 [17] from BEIR [32]. The description of the two datasets is shown in Table 2. We follow BEIR [32] to report the metrics on the evaluation dataset instead of test data for MS MARCO, so, for MS MARCO, D_{test} is the same as D_{eval}. As shown in Table 3:

a) BM25 is evaluated on the original testing split D_{test}; b) W/O Aug models are trained on datasets S_{train}^{50} and S_{eval}^{100} utilized to fine-tune the soft prompt; c) InPars [2] models are trained on S_{train}^{50} and S_{eval}^{100} plus $F_k W_{large}$ (W_{large} filtered by k Section 3.4) generated by human-written prompts. d) SPTAR's soft prompt tuning module (SPTAR-Tuning) is trained on S_{train}^{50} and evaluated on S_{eval}^{100}; SP-TAR's DR models (SPTAR-DR) are trained on S_{train}^{50} and S_{eval}^{100} plus $F_k W_{large}$ (W_{large} filtered by k Section 3.4) generated by soft prompt augmentor (Section 3.4); e) W/O Aug, InPars [2] and SPTAR are all evaluated and tested on the same splits for a fair comparison.

4.2 Training Details

To train the soft prompt module, we performed fine-tuning using two open-source LLMs: LLaMA-7B and Vicuna-7B. The specific training hyper-parameters are documented in Table 4.

Hyperparameters	LLaMA-7B	Vicuna-7B
Batch Size	4	2
Max Length	1024	1024
Learning Rate	$3e-2$	$3e-2$
Optimizer	AdamW	AdamW
Early Stop	5	5
Max epochs	100	100
GPU	1 A100 (80G)	1 A100 (80G)

Table 4: Hyperparameters of soft prompt tuning

Hyperparameters	DPR	ColBERT
Batch Size	32	32
Max Length	350	350
Learning Rate	$2e-5$	$2e-5$
DDP	No	Yes
Optimizer	AdamW	AdamW
Early Stop	10	None
Max epochs	20	20
GPU	4 A100s (40G)	4 A100s (40G)

Table 5: Hyperparameters of DR Models

The training hyper-parameters of dense retriever DPR and Col-Bert are in Table 5. For ColBERT, there is no early stop and we saved a checkpoint after each epoch in the official code. After train-ing, we manually evaluated some checkpoints (3, 5, 10, 15, 18, 20) and reported the testing results of the checkpoint with the highest NDCG@10 score.

Metric	Dataset	BM25	W/O Aug			InPars			SPTAR		
			DPR	ColBERT	BM25CE	DPR	ColBERT	BM25CE	DPR	ColBERT	BM25CE
NDCG	MS MARCO	0.2284	0.1726	0.0819	0.2288	0.1901	0.2418	0.2311	0.2580	0.2445	0.2647
	FiQA-2018	0.2361	0.1332	0.1224	0.2290	0.2226	0.2370	0.2673	0.2404	0.2749	0.2785
	Avg Imp% over BM25		-34.17%	-56.02%	-1.44%	-11.15%	3.08%	7.30%	7.30%	11.82%	16.94%
MAP	MS MARCO	0.1803	0.1371	0.0643	0.1806	0.1486	0.1939	0.1790	0.2047	0.1958	0.2104
	FiQA-2018	0.1784	0.0934	0.0896	0.1684	0.1654	0.1854	0.2052	0.1804	0.2154	0.2130
	Avg Imp% over BM25		-35.74%	-57.1%	-2.7%	-12.46%	5.74%	7.11%	7.36%	14.64%	18.04%
Recall	MS MARCO	0.3787	0.2842	0.1373	0.3787	0.3206	0.3913	0.3953	0.4246	0.3965	0.4336
	FiQA-2018	0.2951	0.1810	0.1604	0.2951	0.2843	0.2814	0.3258	0.2933	0.3256	0.3363
	Avg Imp% over BM25		-30.96%	-55.82%	0.00%	-10.23%	-0.16%	7.02%	6.54%	7.17%	14.26%

Table 6: SPTAR vs baseline models: a) BM25 is an unsupervised method; b) W/O Aug is a supervised mehtod and the three DR models are trained on dataset without augmentation; c) InPars [2] utilizes human-written prompts and it has no soft prompt filter mechanism; d) Table 3 docuemts the data splits for each method.

4.3 Evaluation Metrics

In the context of text generation models, Perplexity is a commonly employed metric that quantifies the level of uncertainty exhibited by a language model when generating new tokens. This metric is defined as the exponentiated average negative log-likelihood of a sequence, and a lower perplexity value indicates a higher-quality language model. Perplexity is used to evaluate the soft prompt tuning and filter modules.

For DR models, we reporte the testing results of the checkpoint with the highest NDCG@10 score. Besides NDCG metric, we have also incorporated several other DR evaluation metrics to facilitate the assessment of the improvement achieved by synthesized augmented queries. These additional metrics include Mean Average Precision (MAP) [41] and Recall [41]. By utilizing these metrics, we can comprehensively evaluate the effectiveness of the synthesized augmented queries in enhancing the performance of DR models.

4.4 Baseline Methods

We choose three baseline methods: BM25, Without Augmentation (W/O Aug), and InPars [2] (Section 2.2). The training, evaluation, and testing datasets are documented in Section 4.1. For BM25 [25], we use Anserini [15] with the default Lucene parameters($k = 0.9$ and $b = 0.4$). The differences between InPars [2] and SPTAR are twofold: a) InPars [2] utilizes the human-written prompt while SPTAR utilizes an optimized soft prompt; b) SPTAR has a soft prompt filter module to select example document-query pairs. To make it a fair comparison with InPars [2], we choose the same example document-query pairs in the prompt of SPTAR for InPars [2] and utilize InPars' original human-written prompt to prompt the LLaMA and Vicuna to obtain weak document-query pairs. We find for InPars' human-written prompt, the quality of generated weak document-query pairs of Vicuna is much better than that of LLaMA, so, for InPars [2], we choose Vicuna as the weak data generator.

4.5 Research Questions

An extensive set of experiments were designed to address the following research questions:

RQ1: Can the proposed SPTAR framework achieve improved performance on DR tasks over the baseline models? (Section 5.1)

RQ2: During the soft prompt tuning process, does the soft prompt tuning module indeed distill the knowledge from the dataset to the learned soft prompt? What factors contribute to the learned soft prompts? (Section 5.2.1)

RQ3: What are the costs of the soft prompt tuning module? Does the soft prompt tuning module greatly increase the training time and computational resources? (Section 5.2.2)

RQ4: What specific role does the soft prompt filter play in SPTAR? Which metric can be utilized to filter the M example document-query pairs? (Section 5.2.3)

RQ5: Can the weak data filter further improve the performances of DR models? (Section 5.2.4)

RQ6: For SPTAR's soft prompt tuning module, what is the influence of the size of training data X? Is a larger X better than a smaller one? (Section 5.2.5)

RQ7: For SPTAR's soft prompt augmentor module, what is the influence of the number of example document-query pairs M? Is a larger M better than a smaller one? (Section 5.2.6)

5 EXPERIMENTAL RESULTS

5.1 SPTAR vs Baseline Models (RQ1)

As presented in Table 6, our SPTAR approach obtains the best results for all three retrievers, showcasing an average improvement of 7.3% to 18.04% in NDCG@10 compared with BM25. The next is InPars [2] which improves ColBERT and BM25CE in NDCG@10 by 3.08% and 7.3% on average, respectively compared with BM25 while for DPR, InPars [2] is 11.15% lower than BM25. W/O Aug exhibits the worst performance that cannot beat BM25. These outcomes serve as compelling evidence for the effectiveness of our proposed SPTAR model. By harnessing the benefits of soft prompt tuning and LLMs, our model generates high-quality weak queries that

(a) Different datasets. **(b) Different LLMs.** **(c) Different lengths.**
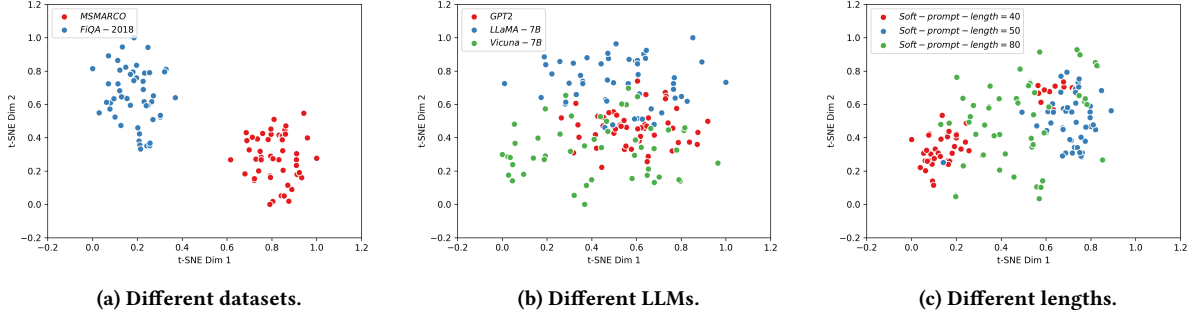
Figure 3: T-SNE embedding visualization of soft prompt's virtual tokens: a) soft prompt's virtual tokens with different datasets; b) soft prompt's virtual tokens with different LLMs; c) virtual tokens of soft prompt with different lengths.

greatly enhance DR tasks. Moreover, the consistent improvements observed across all three retrievers substantiate the general applicability of our approach, extending beyond specific dense retrievers. Notably, among the three retrievers, BM25CE consistently achieves the best overall results across all three methods (W/O, InPars [2], and SPTAR), aligning with the findings of other studies such as [32]. It is worth noting that in the absence of augmentation data, all three dense retrievers perform worse than the unsupervised model BM25. This underscores the significant reliance of DR on domain-specific labeled data and highlights the limitations of directly training dense retrievers in scenarios with limited ground-truth data, where the expected performance may not be attainable.

5.2 Ablation Study

In this section, we primarily aim to assess the individual contributions of different modules to the overall performance of our proposed SPTAR. We conducted experiments focusing on evaluating the perplexity and NDCG@10 metrics. The perplexity metric, derived from the S_{eval}^{100} dataset, provided insights into the model's text generation quality. The default NDCG@10 scores in this section are obtained by evaluating the SPTAR-DPR model trained, evaluated, and tested on $S_{trail}^{50} + S_{eva}^{100} + W_{small}$, D_{eval} and D_{test} respectively. We didn't filter W_{small} so that the NDCG@10 score can directly reflect the quality of the weak data.

5.2.1 The Impact of Soft Prompt Tuning Module (RQ2). To gain deeper insights into the learning process of parameters θ, we employed the t-SNE algorithm [34] to visualize the virtual token vectors of the prompt s when θ are converged with different datasets and LLMs.

Figure 3a presents the distribution of virtual token vectors in a two-dimensional space. We utilized the LLaMA-7B language model with a virtual token length $l_s = 50$ for this experiment. The red and blue points indicate the MS MARCO and FiQA datasets, respectively. The visual analysis clearly reveals that the virtual token vectors from the two datasets exhibit distinct distributions in the two-dimensional space, with minimal overlap. Notably, at the model initialization phase, both datasets share the same prompt s, making the observed changes in vector distribution after convergence particularly significant. These findings highlight the remarkable capability of prompt tuning to distill domain-specific knowledge from

datasets to the learned prompt token vectors. This accomplishment is particularly noteworthy in the scenario where ground-truth data are too limited that human-written prompts struggle to capture domain-specific information and incorporate it effectively into the prompt design.

In Figure 3b, the points of different colors represent distinct LLMs: GPT-2, LLaMA-7B, and Vicuna-7B. We kept all the hyperparameters same except for the language model to analyze the influence of different language models on the parameters θ. The dispersion of points with the same color indicates the extent of parameter updated during training. Figure 3b clearly illustrates that the red point cloud representing the GPT-2 model has less dispersion, with points tightly clustered together. In contrast, the blue point cloud representing LLaMA-7B and the green point cloud representing Vicuna-7B exhibit greater dispersion of virtual token vectors. This observation suggests that, when trained on the same dataset, the LLaMA-7B and Vicuna-7B models enable the soft prompt module to acquire more domain-specific knowledge, leading to an enhancement in the generation of synthesized queries. Moreover, similar findings were obtained when decoding the virtual tokens into corresponding words. For instance, after training the GPT-2 model, we observed that the resulting soft prompt merely replicates the prompt tokens used during initialization, essentially duplicating the manual prompt without additional learning. In contrast, when decoding the virtual token vectors into words utilizing the LLaMA-7B and Vicuna-7B, we discovered that these models not only retain the initial prompt tokens but also acquire additional symbols and representations associated with relevant text, such as "query," "rewrite", "argument", "enhance" and "adding", indicating parameters θ does learn task-specific knowledge.

In Figure 3c, we analyze the influence of different soft prompt lengths on the soft prompt tuning module by examining the distribution of virtual token vectors of the learned soft prompt in vector space. This experiment was conducted on LLaMA-7B and dataset MS MARCO and all the hyperparameters are same except for the soft prompt length. The three lengths 40, 50, and 80 are represented by the colors red, blue, and green, respectively. From the point distribution in Figure 3c, we observe partial overlap between the red and blue points, as well as some distinct points. As the virtual token length increases, the embedding distribution area of the longer soft

prompt encompasses the regions corresponding to the shorter ones: 40 and 50. This outcome is consistent with our expectations: with different lengths of soft prompts, the embedding distributions of soft prompts' virtual tokens are different. Nevertheless, the distributions of the three soft prompts with different lengths should demonstrate relative concentration with substantial overlapping regions.

For RQ2, we have conclusions: a) we can distinguish the datasets from the learned soft prompts, demonstrating that soft prompt tuning does learn task-specific soft prompts; b) both the LLMs and the length of soft prompts influence the learned soft prompts.

LLM	$count(\theta)/count(\Phi)$	Best Epoch #
GPT-2	0.0308%	17
LLaMA-7B	0.0030%	5
Vicuna-7B	0.0030%	4

Table 7: Efficiency evaluation of SPTAR's soft prompt tuning module on MS MARCO S_{train}^{50} and S_{eval}^{100} (Section 3.1).

5.2.2 The Efficiency of Soft-Prompt Tuning (RQ3). Table 7 presents a comparison of the training parameters and convergence efficiency achieved through soft prompt tuning, utilizing different language models during the training process on the MS MARCO dataset. For the soft prompt tuning module in our proposed SPTAR, although the count of LLM's original parameters Φ is extremely large, Φ remain frozen and do not require fine-tuning. The count of trainable parameters θ associated with the fine-tuning of the soft prompt is much smaller. The percentage values in the second column reveal that the fine-tuning of the soft prompt module necessitates a remarkably small number of parameters θ, roughly amounting to 0.003% of the count of Φ. Notably, the count of θ does not increase alongside the expansion of Φ. This characteristic significantly enhances the practicality and training efficiency of SPTAR, as we can fine-tune task-specific soft prompts with a minimal fraction of parameters for optimization.

Furthermore, for a new task or dataset, SPTAR can swiftly complete the fine-tuning process of the soft prompt tuning module within a few epochs. As exemplified in the third column of the table, we examined the convergence speed of the soft prompt tuning model on the evaluation dataset S_{eval}^{100} (Section 3.1) by the best epoch number and the lower this number is, the faster it converges. It becomes apparent that employing a more advanced language model expedites the convergence of the soft prompt tuning module, requiring a mere four or five epochs for convergence. Considering both the count of θ and the convergence speed, we can confidently conclude that the soft prompt tuning module leverages the advantages offered by LLMs while effectively mitigating the computational resource consumption associated with fine-tuning the whole LLMs.

In conclusion, the soft prompt tuning model only fine-tunes a small part of the parameters θ, and the training converges quickly on LLMs.

5.2.3 The Impact of Soft Prompt Filter Module (RQ4). With the learned parameters θ in SPTAR's soft prompt tuning module, we find the example document-query pairs in SPTAR's soft prompt

Dataset	Fliter	PPL (Dec%)	NDCG@10 (Imp%)
MS MARCO	Worst	4.1934	0.2132
	Best	3.6649 (+12.60%)	0.2376 (+11.44%)
FiQA-2018	Worst	410.9207	0.1855
	Best	5.7898 (+98.59%)	0.1923 (+3.67%)

Table 8: Evaluation of SPTAR-DPR with the best and worst example document-query pairs in soft prompt augmentor module. SPTAR-DPR is trained on $S_{train}^X + S_{eval}^{100} + W_{small}$ and tested on D_{test}. Results are obtained on LLaMA-7B. For MS MARCO and FiQA-2018, $M = 2$ and $M = 1$ respectively.

augmentor module does influence the quality of the generated weak data, so it is necessary to select certain M document-query pairs from S_{train}^X. In this section, we study the impact of SPTAR's soft prompt filter module. In Table 8, we report the best results of SPTAR-DPR (Section 5.2.6): a) for MS MARCO, we report the results of SPTAR-DPR with LLaMA-7B and $M = 2$; b) for FiQA-2018, we report the results of SPTAR-DPR with LLaMA-7B and $M = 1$. The SPTAR-DPR is trained on $S_{train}^{50} + S_{eval}^{100} + W_{small}$ and tested on D_{test}. The best and worst M example pairs in Table 8 are filtered by the method proposed in Section 3.3.

As shown in Table 8, the results unequivocally demonstrate that the soft prompt filter significantly enhances performance across all comparisons. Specifically, we observe a noteworthy 12.60% to 98.59% decrease in perplexity and a substantial 3.67% to 11.44% improvement on NDCG@10 in the downstream DPR model. Furthermore, our experimental findings indicate that while the utilization of in-context learning theory, complemented by limited examples, greatly enhances the quality of generated weak queries, the choice of example document-query pairs also exerts a considerable influence on text generation quality.
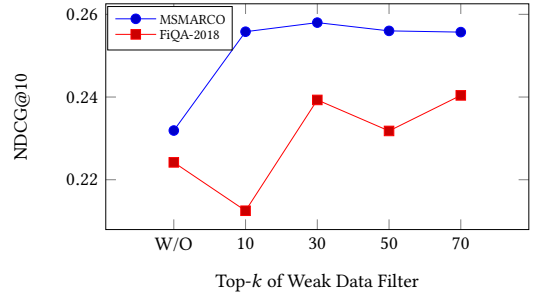

Figure 4: SPTAR-DPR NDCG@10 scores with different top-k of weak data filter. SPTAR-DPR is trained on $S_{train}^{50} + S_{eval}^{100} + F_k W_{large}$ (Section 4.1). Results are obtained on LLaMA-7B. For MS MARCO and FiQA-2018, $M = 2$ and $M = 1$ respectively.

5.2.4 The Impact of Weak Data Filter Module (RQ5). We utilized different top-k to filter the generated weak data W_{large} to get $F_k W_{large}$ and tested SPTAR-DPR model trained on $S_{train}^{50} + S_{eval}^{100} + F_k W_{large}$ on D_{test} to show how many improvements we can obtain compared with the method without weak data filter. We choose the best parameters LLM and M from Section 5.2.3 and Section 5.2.6,

then, we fix them in this section to study the impact of top-k only. As shown in Figure 4, on MS MARCO, SPTAR-DPR model without the data filter gets an NDCG@10 score of 0.2319 while it gets a NDCG@10 score of 0.2580 with data filter top-k=30. On FiQA-2018, SPTAR-DPR with filter top-k=70 gets the highest NDCG@10 score of 0.2404, while it gets an NDCG@10 score of 0.2242 without data filter. The consistent improvements across different datasets prove the effectiveness of the weak data filter module (Section 3.5). As we did not find any pattern between the top-k and the NDCG@10 metric, in practice, top-k is a hyperparameter and needs to be tuned for different datasets.
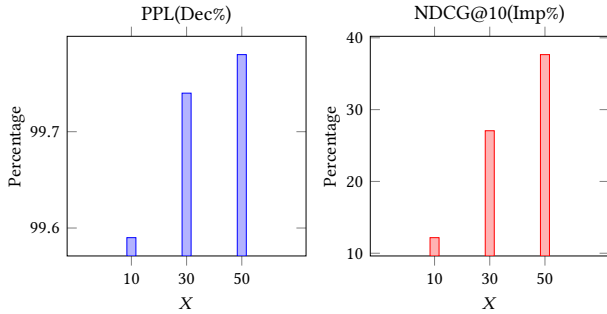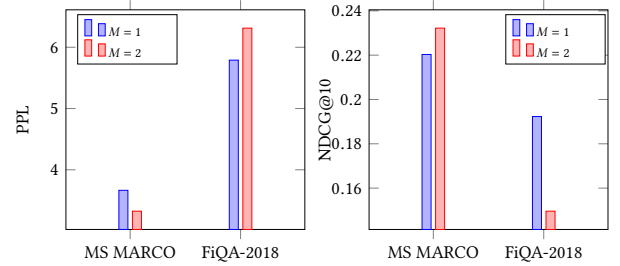

Figure 5: Evaluation of SPTAR-DPR with different X. SPTAR-DPR is trained on $S_{train}^{X} + S_{eval}^{100} + W_{small}$ and tested on D_{test}. Results are obtained on LLaMA-7B and MS MARCO.

5.2.5 The Impact of X (RQ6). In this section, we present an analysis of the impact of different training sizes X in SPTAR's soft prompt tuning module. To evaluate the impact of X, we first conducted soft prompt tuning on S_{train}^{X} and evaluated the PPL on S_{eval}^{100}. PPL is a direct metric to measure the impact of X on the quality of generated weak queries. Then, we generated W_{small} and tested the SPTAR-DPR model trained on $S_{train}^{X} + S_{eval}^{100} + W_{small}$ on D_{test}. NDCG@10 score is applied to measure the impact of X on downstream DR models, like DPR. As shown in Figure 5, the findings conclusively demonstrate substantial improvements when employing soft prompt tuning with varying training sizes X compared with the results obtained without soft prompt tuning. For instance, when $X = 50$, PPL is decreased by 99.78%, and an impressive 37.66% enhancement is observed. PPL is much easier than NDCG@10 to be improved, which means that there is a gap between the two metrics.

Different from InPars [2] and Promptagator [6], which only utilizes several example document-query pairs in human-written prompts, our experimental results highlight the advantages of a little larger training size X in soft prompt tuning, leading to better performance. This is evident in both the decreased PPL, as well as the improvement of NDCG@10 in downstream tasks as the training size X expands.

5.2.6 The Impact of M (RQ7). In SPTAR's soft prompt agumentor module, when tagging the unlabeled documents with weak queries, M filtered example document-query pairs are utilized to instruct the LLM. In this section, we explore the impact of different M. We first chose LLaMA-7B as the LLM and did soft prompt tuning

on S_{train}^{50}. PPL is computed on S_{eval}^{100}. Then, with the filtered M example document-query pairs from SPTAR's soft prompt filter module (Section 3.3), we generated W_{small}. Finally, SPTAR-DPR trained on $S_{train}^{50} + S_{eval}^{100} + W_{small}$ is tested on D_{test} to compute NDCG@10. We also did the same experiments on Vicuna, and we found LLaMA-7B model consistently delivers better results than Vicuna-7B model, no matter $M = 1$ or $M = 2$, so, we only report the results on LLaMA-7B in Figure 6.

Figure 6: Evaluation of SPTAR-DPR with different M. SPTAR-DPR is trained on $S_{train}^{X} + S_{eval}^{100} + W_{small}$ and tested on D_{test}. Results are obtained on LLaMA-7B.

As shown in Figure 6, for dataset MS MARCO, $M = 2$ achieves the best performance in terms of perplexity and NDCG@10. In contrast, for dataset FiQA-2008, $M = 1$ demonstrates superior performance. This is inconsistent with our expectation that the bigger M is the better the PPL and NDCG@10 are. We attribute this inconsistency to varying dataset distributions. Given that most QA datasets in which a document has multiple relevant queries and each query is only based on a subset of the document, leading to increased uncertainty and heightened learning complexity for the model. Consequently, these factors contribute to divergent outcomes for different datasets. Thus, we recognize the need for further investigation and exploration of this matter in future studies.

6 CONCLUSION AND FUTURE WORK

In this paper, we introduce the Soft Prompt Tuning for Augmenting DR (SPTAR) framework as a solution to address the challenge of limited domain-specific training data in DR tasks. Our approach leverages soft prompt tuning to optimize prompts using a limited ground truth dataset. By prompting the Language Models (LLMs) with these optimized prompts, we generate weak queries for unlabeled documents, resulting in an abundant collection of weak document-query pairs for training domain-specific dense retrievers. To further enhance the quality of the generated weak tagged queries, we incorporate a soft prompt filter that selects high-quality example document-query pairs in the prompt. The effectiveness of our proposed approach is validated through comprehensive experiments. This work represents an initial step toward a promising research direction. Moving forward, we plan to assess the generalizability of SPTAR by evaluating it on more datasets. Additionally, investigating the feasibility of learning a multi-task soft prompt tuning module to enhance efficiency is another direction to be explored.

REFERENCES

[1] Adam L. Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu O. Mittal. 2000. Bridging the lexical chasm: statistical approaches to answer-finding. In *SIGIR*. ACM, 192–199.

[2] Luiz Henrique Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Frassetto Nogueira. 2022. InPars: Unsupervised Dataset Generation for Information Retrieval. In *SIGIR*. ACM, 2387–2392.

[3] Leonid Boytsov, Preksha Patel, Vivek Sourabh, Riddhi Nisar, Sayani Kundu, Ramya Ramanathan, and Eric Nyberg. 2023. InPars-Light: Cost-Effective Unsupervised Training of Efficient Rankers. *CoRR* abs/2301.02998 (2023).

[4] 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. (2020).

[5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[6] Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755* (2022).

[7] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise Zero-Shot Dense Retrieval without Relevance Labels. *CoRR* abs/2212.10496 (2022).

[8] Zhiqi Huang, Hansi Zeng, Hamed Zamani, and James Allan. 2023. Soft Prompt Decoding for Multilingual Dense Retrieval. *CoRR* abs/2305.09025 (2023).

[9] Vitor Jeronymo, Luiz Henrique Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto de Alencar Lotufo, Jakub Zavrel, and Rodrigo Frassetto Nogueira. 2023. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. *CoRR* abs/2301.01820 (2023).

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

[11] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*. ACM, 39–48.

[12] 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.

[13] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL/IJCNLP*. Association for Computational Linguistics, 4582–4597.

[14] 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).

[15] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig MacDonald, and Sebastiano Vigna. 2016. Toward Reproducible Baselines: The Open-Source IR Reproducibility Challenge. In *ECIR (Lecture Notes in Computer Science, Vol. 9626)*. Springer, 408–420.

[16] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-Shot Listwise Document Reranking with a Large Language Model. *arXiv preprint arXiv:2305.02156* (2023).

[17] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering. In *WWW*. ACM, 1941–1942.

[18] 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.

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

[20] 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. *choice* 2640 (2016), 660.

[21] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713* (2020).

[22] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True Few-Shot Learning with Language Models. In *NeurIPS*. 11054–11070.

[23] 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).

[24] 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. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[25] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[26] 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. *arXiv preprint arXiv:2204.07496* (2022).

[27] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488* (2021).

[28] Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? *arXiv preprint arXiv:2103.08493* (2021).

[29] Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *arXiv preprint arXiv:2304.09542* (2023).

[30] 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. *arXiv preprint arXiv:2207.07087* (2022).

[31] 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.

[32] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663* (2021).

[33] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[34] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[35] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. (2019), 3261–3275.

[36] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. (2020).

[37] 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.

[38] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652* (2021).

[39] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[40] 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.

[41] Mu Zhu. 2004. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo* 2, 30 (2004), 6.