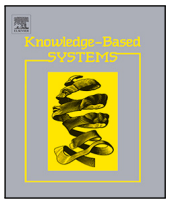
Knowledge-Based Systems 309 (2025) 112758

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Soft prompt tuning for augmenting dense retrieval with large language models

Zhiyuan Peng [a,1], Xuyang Wu [a,1], Qifan Wang [b], Yi Fang [a,*]

[a] Santa Clara University, Santa Clara, 95053, CA, USA
[b] Meta AI, USA

ARTICLE INFO

Keywords:
Large language models
Dense retrieval
Soft prompt tuning
Data augmentation

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 major 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. Recently, 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 are suboptimal and the generated weak queries are often sensitive to the prompts. To tackle this, we propose soft prompt tuning for augmenting DR (SPTAR): 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 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. Moreover, unlike much of the existing work, ours is based on popular open-source LLMs to ensure reproducible and deterministic results. Our experimental results demonstrate that SPTAR outperforms both unsupervised baselines and the recently proposed LLMs-based augmentation method for DR.

1. Introduction

Information retrieval (IR) plays a pivotal role in a wide array of applications, ranging from prominent web search engines such as Google and Bing to personalized recommendation systems like Walmart's product recommendations and Apple Music's song suggestions. Traditional IR methods, like TF-IDF and BM25 [1], are built on token-level similarity matching, which can sometimes fall short due to a lexical gap [2]. This gap occurs when semantically similar terms, such as synonyms, are overlooked because of their lexical differences. This oversight can potentially impact the quality of search results and the user experience.

Given these constraints, researchers have turned to advancements in deep learning to tackle the lexical gap in conventional IR. One notable approach is Dense Retrieval (DR), which aims to capture the overarching semantic essence of content rather than fixating on individual tokens. DR models like dense passage retrieval (DPR) [3] and ColBERT [4,5] encode each query or document into dense vectors, with the dimensionality determined by the neural networks. In practice, dense retrievers pre-compute document embeddings and construct an approximate nearest neighbor (ANN) index for rapid search. When a new query is introduced, only its embedding is computed and subsequently processed by the ANN search system. Unlike TF-IDF and BM25, DR places greater emphasis on assessing the similarity of the overall semantic context.

While DR methods have made strides in bridging the lexical gap, they are still constrained by the limited availability of domain-specific training data, hindering their performance in specialized domains. Although some researchers have proposed to leverage transfer learning to mitigate this challenge, studies [6,7] indicate that not all DR models and domains can benefit from transfer learning equally. Recently, LLMs like GPT-3 [8], LLaMA [9], and Vicuna [10] have demonstrated potent zero-shot and few-shot learning. Rather than fine-tuning the LLMs on task-specific data, prompting integrates task instructions (e.g., TL;DR translate to English) and a few relevant examples as input and extracts the answers from the output of large language model (LLM). The terms "hard prompt" and "soft prompt" refer to different approaches to guiding the LLM's behavior during text generation or other tasks. A hard

* Corresponding author.
E-mail addresses: zpeng@scu.edu (Z. Peng), xwu5@scu.edu (X. Wu), wqfcr@meta.com (Q. Wang), yfang@scu.edu (Y. Fang).
[1] Both authors contributed equally to this research.

https://doi.org/10.1016/j.knosys.2024.112758
Received 23 November 2023; Received in revised form 8 September 2024; Accepted 12 November 2024
Available online 22 November 2024

prompt [11] involves using explicitly defined and unchangeable text inputs to instruct the model. The prompt does not involve additional training or fine-tuning of the model. On the other hand, a soft prompt involves using trainable vectors or learnable embeddings to guide the model's behavior. Unlike hard prompts, soft prompts are not explicit text instructions but rather embeddings that influence the model's output. These embeddings are typically learned through a process known as prompt tuning [12–15]. The existing work [16,17] suggested that prompts provide a method for injecting task-specific guidance, which is beneficial in low-data regimes. Recent research [18] further quantified this benefit through comprehensive testing of prompts. The results showed that well-crafted prompts can significantly reduce the dependency on large volumes of training data across downstream tasks. Both InPars [19] and PROMPTAGATOR [7] employ hard prompts to guide LLMs in tagging unlabeled documents with weak queries, subsequently training task-specific retrievers. Nonetheless, hard prompts come with limitations: (a) Crafting effective hard prompts is challenging and often requires iterative human effort, intuition, and sometimes a bit of luck; (b) Even with hand-crafted prompts, the downstream tasks still underperform tuned models. For instance, compared with the performance of fine-tuned T5-XXL [20] on SuperGLUE [21], GPT-3 175B few-shot gets a 17.5 points smaller score despite using 16 times more parameters [12]. These limitations of hard prompts underscore their effectiveness and addressing these challenges draws academic interest as well as generating industrial value.

Given the limitations of hard prompts, we investigate an alternative. Rather than utilizing humanly-readable words as hard prompts [22], the soft prompt [12–15] comprises a set of embeddings which are unrecognizable to humans and are prepended at the beginning of the neural network input. During the soft prompt tuning, the parameters of the LLM are frozen, and only the parameters associated with the soft prompt are updated. While both [12,13] demonstrate that soft prompts surpass the hard prompts, 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 the 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 sometimes sensitive to the example document–query pairs in the prompt. Thus, we designed a filter to select high-quality example document–query pairs in the prompt to further improve the quality of weakly tagged queries as well as the DR tasks. In addition, 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. Instead, our work is based on widely used open source LLMs [23]. Our main contributions can be summarized as follows:

- To the best of our knowledge, our work stands as one of the early attempts of LLMs in combination with soft prompt tuning for enhancing DR tasks.
- We introduce a soft prompt filter designed to curate document–query pairs within the prompt, thus enhancing the overall quality of the generated weak data. Additionally, we design a BM25 filter to reduce noise in the generated data, further improving performance.
- We conduct a comprehensive set of experiments involving four datasets and seven retrievers and re-rankers, demonstrating the generality and superior performance of our approach over several state-of-the-art baselines.
- Experiments are based on the recent open-source LLMs to ensure reproducible and deterministic experimental results. All code and data are publicly available.[2]

[2] https://github.com/zhiyuanpeng/SPTAR.git.

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 [3] employs 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 [4,5] share the same BERT but utilize a different special token following the "[CLS]" to distinguish query and document. Unlike DPR directly measures the similarity between query embedding and document embeddings, ColBERT 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 [24] 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. TAS-B [25] groups queries by their embedding similarities and employs a training data sampling technique coupled with dual-teacher supervision distillation. Contriever [26] trains a bi-encoder model through contrastive learning. Instead of training the model on the labeled dataset, Contriever generates positive query–document pairs from unlabeled corpus by "independent cropping". ReContriever [27] adopts the same method as Contriever to generate the weak query–document pairs, but ReContriever scores the weak query–document pairs by itself during the training and the loss is weighted by the weights. BM25CE [28] 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. Data augmentation for dense retrieval

For DR datasets, usually, only a fraction of documents are labeled with queries, for instance, MS MARCO [29], a widely used dataset in DR, has a corpus of 8 841 823 documents but only has 532 761 training document–query pairs. Given DR demands substantial training data to achieve quality dense embeddings, some researchers have turned to data augmentation to generate more document–query pairs to train better dense embeddings. InPars [19] feeds a task-specific human-written prompt and 3 example document–query pairs to a 6B GPT-3 [8] 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 [19] employs the same dense retrieval model proposed in [30], 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. Improved variations of InPars [19], such as InPars-v2 [31] and InPars-Light [32], have been introduced to enhance the original methodology. Like InPars [19], PROMPTAGATOR [7] 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, PROMPTAGATOR 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 [7] utilizes a much bigger LLM: a 175B model Flan [33] which cannot be accessed by most researchers. DAR [34] argues that the method that generates queries from unlabeled documents is costly as well as does not add variations to the documents. To do data augmentation efficiently, DAR [34] not only interpolates two different document representations associated with the labeled query but also stochastically perturbs the representations of labeled documents in embedding space. RocketQA [35] applies a pre-trained cross-encoder retriever to retrieve positive and negative documents for a new collection of queries with high confidence scores. RocketQAv2 [36] augments the DR by jointly optimizing the bi-encoder structure DR and cross-encoder structure reranking model to have similar output distributions. DRAGON [37] fuses multiple teacher models by progressively training the base DR model.

2.3. LLMs in dense retrieval

Most of the current literature in this domain explores the potential of LLMs to improve DR tasks through various data generation techniques, including query generation [7,19,31,32,38,39], relevance generation [40], and permutation generation [41–43]. PROMPTAGATOR [7] and InPars [19] with its variations InPars-v2 [31] and InPars-Light [32] are illustrated in Section 2.2. UPR [38] 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 log-likelihood of the question tokens conditioned on the input document as the relevance score. Due to the intensive computational resources required to train LLMs, all these works utilize LLMs as query generators instead of fine-tuning them. HyDE [39] leverages LLMs to augment queries by generating hypothetical documents, effectively capturing relevance patterns for unsupervised retrieval. LRL [41] trains a listwise zero-shot reranker 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 [42], 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.

2.4. Prompt tuning

Prompt tuning offers a promising avenue for adapting pre-trained LLMs to specific tasks by focusing on tuning the prompt module instead of fine-tuning the entire model [44]. 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 preserves 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*" [45], compresses arbitrary prompts into a condensed set of virtual "*gist*" tokens using a meta-learning approach. Building upon T5 [20], Lester et al. [12] propose 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. [46] introduce 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. Prompt tuning can benefit from multi-task learning. For instance, ATTEMPT proposed by Wang et al. [47] 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. [48] explore 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. [49] employs a dual-encoder, 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.

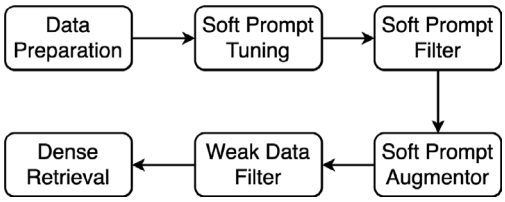


Fig. 1. The pipeline of the proposed Soft Prompt Tuning for Augmenting dense Retrieval (SPTAR).

In contrast, unlike the current DR data augmentation works that only prompt LLMs to generate weak queries for unlabeled documents with a few labeled document–query pairs as examples, we propose to learn task-specific soft prompts on a small proportion of the labeled data and 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. Soft prompt tuning for augmenting dense retrieval

As shown in Fig. 1, SPTAR comprises six modules: (a) data preparation; (b) soft prompt tuning; (c) soft prompt filter; (d) soft 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 introduce the soft prompt filter (Section 3.3) which identifies optimal example document–query pairs to optimize the task-specific prompt. We then prompt LLMs to generate weak queries for unlabeled documents (Section 3.4), yielding enough training data to train DR. Finally, we train the DR (Section 3.6) models on filtered weak data (Section 3.5). The notations used in this paper are provided in Table 1.

3.1. Data preparation

We study the augmentation of DR using limited data. The initial step involves sampling 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 has more unlabeled documents denoted as $C_{unlabeled}$. After training, DR encodes all the documents in C into

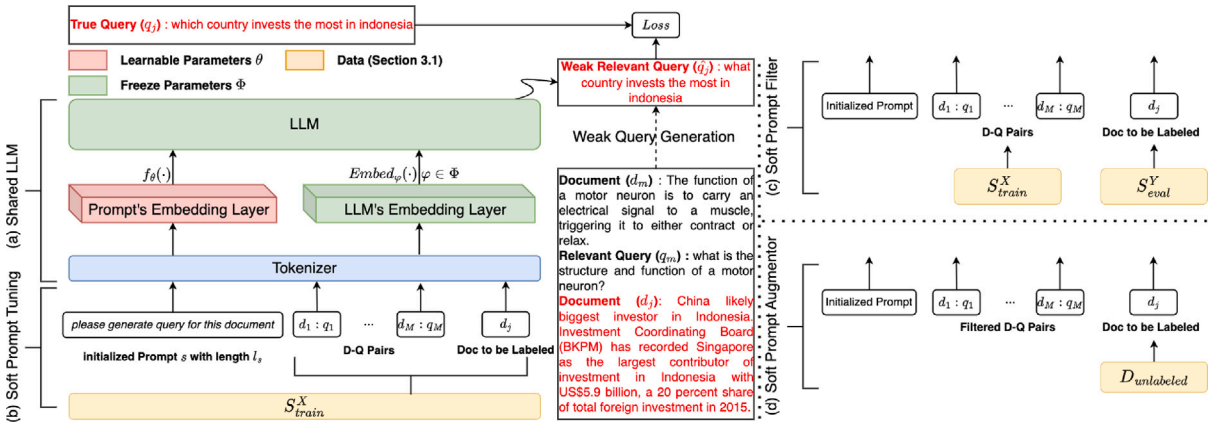


Fig. 2. The main architecture of the proposed SPTAR: (a) The same LLM is shared by soft prompt tuning module, soft prompt filter module and soft prompt augmentor module; (b) soft prompt tuning module fixes the LLM's original parameters Φ and only fine-tune the parameters of soft prompt's embedding layer 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–query pairs with the smallest loss will be utilized in the soft prompt augmentor module; (d) with the learned parameters θ^* and a group of filtered example document–query pairs, the soft prompt augmentor module iterates over the unlabeled document dataset $D_{unlabeled}$ to generate weak queries; (e) Training document–query pair d_j, q_j and generated weak query $\hat{q}_j$ are colored in red.

Table 1
Summary of notation.

Notation	Definition
q, d	Query, document
D_{train}	Collection of query–document pairs for training
D_{test}	Collection of query–document pairs for testing
D_{eval}	Collection of query–document pairs for evaluation
C	Collection of all the documents
$C_{unlabeled}$	Unlabeled documents in C
Φ	Freezed original parameters of large language model
S^X_{train}	Collection of X sampled queries associated with corresponding documents from D_{train}
$f_\theta(\cdot)$	Prompt's embedding layer parametrized by θ
$(d_m, q_m)_{m=1}^M$	Collection of M sampled document–query pairs from S^X_{train}, as examples in prompt
$f_\theta(s)$	Soft prompt initialized based on a hard prompt s with a length of l_s
$(d_j, q_j)_{j=1}^{NumPair(X)-M}$	For each epoch, loss is computed on $NumPair(X) - M$ document–query pairs from S^X_{train}
$NumPair(X)$	The number of document–query pairs in dataset with X different queries
c_j	Concatenation of $(d_j, q_j) \in (d_j, q_j)_{j=1}^{NumPair(X)-M}$ and all the example pairs $(d_m, q_m)_{m=1}^M$
t_j	Concatenation of s and c_j
$p_{\theta,\Phi}(q_j \mid t_j)$	The probability of q_j conditioned on t_j given parameters θ and Φ
$h_{j,i}$	Output hidden vector of ith time step for jth instance
$z_{j,i}$	ID of ith token of jth instance
L	Loss function
S^Y_{eval}	Collection of Y queries associated with corresponding documents from D_{eval}
W_{large}	Collection of 100K sampled documents from $C_{unlabeled}$ and each document has a generated weak query
W_{small}	Collection of 5000 sampled documents W_{large}
$F_k(\cdot)$	Top k weak data filter function
$F_k(W_{large})$	W_{large} filtered by weak data filter F_k

vectors. When a new query comes in, DR encodes the query into a vector and searches the top-k similar documents in vector space.

We randomly sample document–query pairs from the original training dataset D_{train} to construct the training and evaluation datasets for the soft prompt module, namely S^X_{train} and S^Y_{eval} where indices X and Y signify the number of distinct queries within the training and evaluation datasets respectively. The function $NumPair(x)$ designates the quantity of document–query pairs given x distinct queries in the dataset. Since each query may have more than one positive document, $NumPair(x)$ may be bigger than $|x|$. Hence, S^X_{train} contains $NumPair(X)$ document–query pairs, similarly, S^Y_{eval} comprises $NumPair(Y)$ document–query pairs. For illustration, in our experiment, we draw 50 unique queries and their corresponding documents from the training dataset S_{train} to form $S^{50}_{train}(X = 50)$. From the remaining data in S_{train}, we randomly select 100 unique queries and their associated documents to compose $S^{100}_{eval}(Y = 100)$. S^{50}_{train} serves for optimizing the soft prompt, while S^{100}_{eval} is employed to assess the model's convergence, enabling us to terminate the training process in advance (Section 4.3) 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

Soft prompts [50,51] introduce a novel technique to steer a model's behavior without the need for extensive fine-tuning. Unlike hard prompts, which are human-readable instructions, soft prompts comprise trained embeddings optimized for specific tasks. The soft prompt tuning module learns a task-specific soft prompt on a small proportion of labeled data. Fig. 2 (b) illustrates the structure of 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 during the training. s represents the initialized hard prompt with size l_s, like repeating "please generate query for document" until the length of s equals l_s. Let $f_\theta(\cdot)$ denote the prompt's embedding layer implemented by an embedding matrix initialized as the embeddings of s encoded by LLM's original embedding layer. $f_\theta(s)$ represents the soft prompt.

For each training epoch, we first randomly sample M document–query pairs from training dataset S^X_{train} as example document–query pairs $(d_m, q_m)_{m=1}^M$, then iterate over the left document–query pairs $(d_j, q_j)_{j=1}^{NumPair(X)-M}$ to compute loss. Example pairs $(d_m, q_m)_{m=1}^M$ are concatenated with each pair (d_j, q_j) by keywords like "*document*" and "*query*" as c_j. Finally, we concatenate s with c_j as one training instance $t_j = [s; c_j]$ and there are $NumPair(X) - M$ instances in each epoch. When t_j is inputted into the 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. $f_\theta(\cdot)$ takes the IDs of s as inputs and outputs its embeddings while the embeddings of c_j are generated by LLM's original embedding layer $Embed_\varphi(\cdot)$ where $\varphi \in \Phi$. For simplicity, we postulate that 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 $h_{j,i} = \left[h^{(1)}_{j,i}; \cdots; h^{(k)}_{j,i}\right]$ and k is the number of layers in LLM. The objective

function for training is given by:

$$\max_{\theta} \log p_{\theta,\phi}(q_j \mid t_j) = \max_{\theta} \sum_{i \in idx_{q_j}} \log p_{\theta,\phi}\left(z_{j,i} \mid h_{j,<i}\right) \tag{1}$$

where idx_{q_j} denotes the indexes corresponding to the IDs of d_j. Additionally, $p_{\theta,\phi}\left(z_{j,i} \mid h_{j,<i}\right)$ signifies the probability of the subsequent token with ID $z_{j,i}$. For loss function L, we employ the negative log-likelihood defined as:

$$L = -\log p_{\theta,\phi}(q_j \mid t_j) \tag{2}$$

We implemented our soft prompt tuning module based on a public prompt tuning package PEFT [52].

3.3. Soft prompt filter

During the development of the soft prompt module, we observe that the choice of example document–query pairs $(d_m, q_m)_{m=1}^{M}$ profoundly affects 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 pairs from S^X_{train} as example document–query pairs in soft prompt augmentor. For $M = 2$, there are 1225 ($C^2_{50} = 50 * 49/2 = 1225$) groups of example pairs, which makes it impractical to evaluate all. To reduce the computation complexity, we randomly sample X groups of example pairs from S^X_{train} to evaluate them on the evaluation dataset S^Y_{eval} and the group of example pairs with the best evaluation metric will be chosen as the example pairs in soft prompt augmentor. For instance, we sample 50 groups of document–query pairs from $S^{X=50}_{train}$ where each group contains two ($M = 2$) document–query pairs. As shown in Fig. 2(c), soft prompt filter adopts a similar input format to soft prompt tuning, with the main difference being the dataset from which d_j is derived. 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^Y_{eval} has $NumPair(Y)$ document–query pairs and example pairs $(d_m, q_m)_{m=1}^{M}$ are concatenated with each pair (d_j, q_j) by keywords like "*document*" and "*query*" 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 (Eq. (2)). We study the effectiveness of soft prompt filter in Section 5.2.3 and the filtered example document–query pairs are documented in Appendix.

3.4. Soft prompt augmentor

Generating high-quality queries for unlabeled documents remains a formidable challenge. Our soft prompt augmentor module harnesses both the potency of the learned soft prompts and the context offered by the best example document–query pairs, providing a synergistic effect that ensures not just relevance, but also the superior quality of the generated queries. As shown in Fig. 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 this paper, for each dataset, we first created 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}. 5000 document–query pairs are sampled from W_{large}. Then, we filtered W_{large} and W_{small} by weak data filter, described in Section 3.5, to get $F_k(W_{large})$ and $F_k(W_{small})$. During the weak query generation process, LLM not only utilizes the soft prompt embeddings to capture domain-specific information but also benefits from the supplementary context provided by the best example document–query pairs.

3.5. Weak data filter

Both InPars [19] and PROPAGATE [7] emphasize the importance of filtering weak document–query pairs as the generated weak queries are not guaranteed to be always relevant to the input documents. Employing the methodology from InPars [19], we clean the weak data. Upon acquiring these generated weak pairs (Section 3.4), we apply a BM25-based filtering: For each weak query, BM25 retrieves the top k documents from the corpus C. If the document linked to a weak query isn't among the top k results, the pair gets discarded. This filtering approach is denoted as $F_k(\cdot)$. For datasets MS MARCO and FiQA-2018, we experimented with different top k values from the set $\{10, 30, 50, 70\}$ and reported the best results. The effectiveness of this weak data filter module is discussed in Section 5.2.4.

3.6. Dense retrieval

DR serves as the concluding step in our methodology, where we harness the capabilities of neural networks to retrieve relevant documents. We conducted the experiments on five popular dense retrievers: DPR, ColBERT, TAS-B, Contriever, and ReContriever. The descriptions of the models can be found in Section 2.1. For TAS-B, we used pre-trained bi-encoder model[3] for clustering queries and cross-encoder model[4] and ColBERTv2[5] for teacher models. For Contriever, we refer to the officially released checkpoint[6] as $Contriever_{base}$ and used it for initial evaluations. Additionally, we fine-tuned $Contriever_{base}$ as a bi-encoder dense retrieval model, denoted $Contriever_{be}$. Similarly, for ReContriever, we have $ReContriever_{base}$[7] and $ReContriever_{be}$. Further, we incorporated the cross-encoder model BM25CE, as established in previous literature [28], which re-ranks the top 1000 items retrieved by BM25. Like BM25CE, we employed the DPR model as a bi-encoder to re-rank the top 1000 items retrieved by BM25, referring to this method as BM25BE.

4. Experimental setup

4.1. Datasets

Experiments were performed on four datasets: MS MARCO [29], FiQA-2018 [53] sourced from BEIR [6], DL2019 [54] and DL2020 [55]. DL2019 and DL2020 are two human-labeled datasets associated with MS MARCO, sharing the same training and validation datasets as MS MARCO. As shown in Table 2, DL2019 and DL2020 have significantly more annotated documents per query than MS MARCO, making them more suitable for evaluating retrieval performance. The two datasets were used to evaluate popular retrieval models like DPR and ColBERT. The description of the four datasets can be found in Table 2. We follow BEIR [6] 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, $Contriever_{base}$ and $ReContriever_{base}$ are evaluated on the original testing split D_{test}; (b) InPars [19] models are trained on S^{50}_{train} and S^{100}_{eval} plus $F_k(W_{large})$ (W_{large} filtered by F_k, Section 3.5) generated by human-written prompts. (c) SPTAR-Tuning (SPTAR's soft prompt tuning module as shown in Fig. 1) is trained on S^{50}_{train} and evaluated on S^{100}_{eval}; SPTAR's DR models (SPTAR-DR) are trained on S^{50}_{train} and S^{100}_{eval} plus $F_k(W_{large})$ (W_{large} filtered by F_k, Section 3.5) generated by soft prompt augmentor (Section 3.4); (d) W/O

[3] https://huggingface.co/sentence-transformers/msmarco-bert-base-dot-v5.
[4] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2.
[5] https://downloads.cs.stanford.edu/nlp/data/colbert/colbertv2/colbertv2.0.tar.gz.
[6] https://huggingface.co/facebook/contriever.
[7] https://huggingface.co/Yibin-Lei/ReContriever.

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

Task	Domain	Dataset	Train	Eval	Test			Avg. Word lengths	
			#Pairs	#Query	#Query	#Corpus	Avg. D/Q	Query	Document
Passage retrieval	Misc.	MS MARCO [29]	532,761	N/A	6980	8,841,823	1.1	5.96	55.98
Passage retrieval	Misc.	DL2019 [54]	532,761	N/A	43	8,841,823	215.3	5.96	55.98
Passage retrieval	Misc.	DL2020 [55]	532,761	N/A	54	8,841,823	210.9	5.96	55.98
Question answering	Finance	FiQA-2018 [53]	14,166	500	648	57,638	2.6	10.77	132.32

Table 3
Dataset partition for different methods.

Model	Train	Eval	Test
BM25	N/A	N/A	D_{test}
$Contriever_{base}$	N/A	N/A	D_{test}
$ReContriever_{base}$	N/A	N/A	D_{test}
W/O Aug	$S^{50}_{train} + S^{100}_{eval}$	D_{eval}	D_{test}
InPars	$S^{50}_{train} + S^{100}_{eval} + F_k(W_{large})$	D_{eval}	D_{test}
SPTAR-Tuning	S^{50}_{train}	S^{100}_{eval}	N/A
SPTAR-DR	$S^{50}_{train} + S^{100}_{eval} + F_k(W_{large})$	D_{eval}	D_{test}

Aug, InPars [19] and SPTAR are all evaluated and tested on the same splits for a fair comparison; (e) For $F_k(\cdot)$, we tried $k \in (10, 30, 50, 70)$ and selected the k with the best NDCG@10 score on evaluation dataset.

4.2. Baseline methods

Our study incorporates five baseline methods: BM25, $Contriever_{base}$, $ReContriever_{base}$, Without Augmentation (W/O Aug), and InPars [19] (Section 2.3). The training, evaluation, and testing datasets are documented in Section 4.1. For BM25 [1], we use Anserini [56] with the default Lucene parameters ($k = 0.9$ and $b = 0.4$). TAS-B, *Contriever*, and *ReContriever* are described in Section 3.6. The differences between InPars [19] and SPTAR are twofold: (a) InPars [19] 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 [19], we choose the same example document–query pairs in the prompt of SPTAR for InPars [19] 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 [19], we choose Vicuna as the weak data generator. For SPTAR, LLaMA is better than Vicuna and we choose LLaMA for SPTAR. As shown in Table 4, we trained SPTAR-DPR on $S^{50}_{train}+S^{100}_{eval}+F_{30}(W_{large})$ and tested on D_{test}. For InPars, Vicuna beats LLaMA on 6 out of 8 metrics, and for SPTAR, LLaMA beats Vicuna on all the 8 metrics.

4.3. 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 5.

The training hyper-parameters of dense retrievers are in Table 6. We stop the training process if no better evaluation results are found in continuous "Early Stop" epochs. For ColBERT, there is no early stop in the official code, and we saved a checkpoint after each epoch. After training, we manually evaluated some checkpoints (3, 5, 10, 15, 18, 20) and reported the testing results of the checkpoint with the highest MRR@10 score.

4.4. 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 soft prompt filter modules.

In our evaluation of DR models, we adhere to the metrics established in previous studies [57,58]. For the MS MARCO and FiQA-2018 datasets, we utilize Mean Reciprocal Rank at 10 (MRR@10) and Recall@100. For the DL2019 and DL2020 datasets, we employ Mean Average Precision (MAP), Normalized Discounted Cumulative Gain at 10 (nDCG@10), and Recall@100. For the BM25CE and BM25BE models, which both re-rank the top 1000 items retrieved by BM25, Recall@100 is used to specifically evaluate their re-ranking effectiveness. These metrics together provide a comprehensive assessment of how augmented queries influence the performance of DR models.

4.5. Research questions

An extensive set of experiments was 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? (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)

RQ8: How does each module (Fig. 1) contribute to the retrieval performance? (Section 5.2.7)

5. Experimental results

5.1. SPTAR vs baseline models (RQ1)

We conducted evaluations of our SPTAR method on MS MARCO and its related datasets, including DL2019, DL2020, and FiQA 2018. Since we aim to demonstrate that the weak data generated by SPTAR enhances the performance of dense retrieval in scenarios where ground truth data is limited, we did not train the models on all the available ground truth data. Instead, we adopted the data partition shown in Table 3. As presented in Table 7, the SPTAR data augmentation method consistently outperforms established baselines, including W/O Aug and InPars, across a spectrum of widely used datasets and key retrieval metrics only except the R@100 of SPTAR's ColBERT which is slightly outdone by InPars. TAS-B method gets the most best results (4 out of 10, there are 10 metrics in each row of Table 7) as it is distilled from

Table 4
InPars-DPR and SPTAR-DPR with different LLMs. InPars-DPR and SPTAR-DPR are both trained on $S^{50}_{train} + S^{100}_{eval} + F_{30}(W_{large})$. The better scores are underlined.

Retriever		MS MARCO		DL2019			DL2020		
		MRR@10	R@100	MAP	nDCG@10	R@100	MAP	nDCG@10	R@100
InPars	Vicuna	0.1519	0.6092	0.2474	0.4460	0.3919	0.2407	0.3913	0.4646
	LLaMA	0.1465	0.5968	0.2357	0.4434	0.4031	0.2310	0.3841	0.4824
SPTAR	Vicuna	0.1919	0.6922	0.2815	0.4871	0.4727	0.2955	0.4751	0.5458
	LLaMA	0.2114	0.7118	0.3091	0.5253	0.4928	0.3042	0.5219	0.5585

Table 5
Hyperparameters of soft prompt tuning.

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)

two teacher dense retrieval models (cross-encoder and ColBERT) that are both trained on the full MS MARCO dataset. For all 7 retrievers, 55 out of 70 improvements are statistically significant, with p-values < 0.05. ColBERT takes 8 out of 15 un-significant improvements.

The token-level matching complexity of ColBERT between queries and documents could explain why SPTAR's enhancements on ColBERT did not significantly surpass the InPars baseline on MS MARCO related datasets. DPR's simple architecture makes it have better generalization ability than ColBERT, which is evidenced by the fact that on MS MARCO related datasets, DPR' W/O Aug is better than ColBERT's W/O Aug. Also, there exists noise in the generated queries, especially when the soft prompt is learned from a small dataset (same as the dataset of W/O Aug), and ColBERT's token-level interaction mechanism is more sensitive than DPR.

Regarding the re-ranking models, BM25CE and BM25BE, both re-rank the top 1000 items initially retrieved by BM25. BM25CE employs a cross-encoder for re-ranking, whereas BM25BE integrates a DPR model. Without data augmentation, BM25BE beats BM25CE on 7 out of 10 metrics when trained on a small labeled dataset (W/O Aug). The cross-encoder model of BM25CE, requiring more extensive data to effectively learn complex interactions and mitigate overfitting risks, ultimately demonstrated superior performance under SPTAR over BM25BE on 6 out of 10 metrics.

The officially released checkpoint of $Contriever_{base}$ is loaded to be evaluated, and we further improved its performance by fine-tuning it as a bi-encoder model denoted as $Contriever_{be}$ in Table 7. We found performance improvement even fine-tuning $Contriever_{base}$ on a small dataset as Contriever's W/O Aug is better than $Contriever_{base}$ on all four datasets. Our method, $Contriever_{be}$'s SPTAR, can further improve $Contriever_{base}$ over the second-best value significantly. $Contriever_{be}$'s InPars only beats W/O Aug on 4 out of 10 metrics. Similarly, we evaluated the Re$Contriever_{base}$ and $ReContriever_{be}$ and the same patterns as Contriever are observed as well.

By harnessing the benefits of soft prompt tuning and LLMs, our model generates high-quality weak queries that greatly enhance DR tasks. Moreover, the consistent improvements observed across DPR, TAS-B, BM25BE, BM25CE, Contriever, and ReContriever substantiate the general applicability of our approach, extending beyond specific dense retrievers. It is worth noting that in the absence of augmentation data, all dense retrievers except for the Contriever and ReContriever 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, our primary objective is to evaluate the distinct contributions of each module to the overall efficacy of the SPTAR framework. Our experiments focus on evaluating the perplexity and NDCG@10 metrics. The perplexity metric, derived from the S^{100}_{eval} 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 $S^{50}_{trail} + S^{100}_{eva} + W_{small}$, D_{eval} and D_{test} respectively. We didn't filter W_{small} so that the NDCG@10 score can genuinely represent the quality of the weak data.

5.2.1. The impact of soft prompt tuning module (RQ2)

To gain deeper insights into the optimized parameters θ^*, we employed the t-SNE algorithm [59] to visualize the virtual token vectors of the learned soft prompt $f_{\theta^*}(s)$ when θ^* are converged with different datasets and LLMs.

Fig. 3(a) illustrates 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 Fig. 3(b), various colors distinguish distinct LLMs: GPT-2, LLaMA-7B, and Vicuna-7B. We kept all the hyperparameters the same except for the language model to evaluate 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. Fig. 3(b) 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 absorb 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 using the LLaMA-7B and Vicuna-7B models, we discovered that these models not only retain the initial prompt tokens but also acquire additional symbols and representations that align closely with the task of query generation. Specifically, terms such as "*query*", "*rewrite*", "*argument*", "*enhance*" and "*adding*" emerged. Although these may appear to be general terms, their consistent presence indicates that the fine-tuned parameters have learned

Table 6
Hyperparameters of DR Models.

Hyperparameters	DPR	ColBERT	$Contriever_{be}$	$ReContriever_{be}$	BM25CE	TAS-B
Batch size	32	32	32	32	96	40
Max length	350	350	350	350	512	300
Learning rate	$2e{-}5$	$2e{-}5$	$2e{-}5$	$2e{-}5$	$2e{-}5$	$2e{-}5$
DDP	No	Yes	No	No	No	No
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Early stop	10	None	10	10	10	10
Max epochs	20	20	20	20	20	20
GPU	4 A100s (40G)	4 A100s (40G)	4 A100s (40G)	4 A100s (40G)	4 A100s (40G)	4 A100s (40G)

Table 7
SPTAR vs baseline models: (a) $Contriever_{base}$ is the original Contriever model pre-trained on CC-net and English Wikipedia; (b) $ReContriever_{base}$ is the original ReContriever model pre-trained on the same datasets as Contriever; (c) W/O Aug doesn't use any augmented data and only applies the training data specified in Table 3; (d) InPars [19] utilizes human-written prompts and it has no soft prompt filter mechanism; (e) For SPTAR, $l_s = 50$, $M = 2$; (f) For each retriever, $top - 1000$ documents are retrieved; (g) Within each method, the best results are underscored and the symbols † denote statistically significant enhancements over the second best result, with p-values < 0.05, as determined by a t-test; (h) The best results cross different methods are denoted with symbol ⋆; (i) Table 3 documents the data splits for each method, and the filtered example document–query pairs of SPTAR are documented in Appendix.

Retriever		MS MARCO		FiQA-2018		DL2019			DL2020		
		MRR@10	Recall@100	MRR@10	Recall@100	MAP	nDCG@10	Recall@100	MAP	nDCG@10	Recall@100
BM25		0.1840	0.6578	0.2956	0.5395	0.3013	0.5058	0.4910	0.2856	0.4796	0.5599
$Contriever_{base}$		0.1501	0.6412	0.1726	0.3634	0.2187	0.4243	0.4330	0.2324	0.4028	0.4957
$ReContriever_{base}$		0.1606	0.6729	0.2199	0.4871	0.2489	0.4559	0.4806	0.2449	0.4159	0.5177
DPR	W/O Aug	0.1303	0.5141	0.1433	0.3738	0.1796	0.3484	0.3289	0.2196	0.3806	0.4218
	InPars	0.1519	0.6092	0.2720	0.5289	0.2474	0.4460	0.3919	0.2407	0.3913	0.4646
	SPTAR	†0.2114	†0.7118	0.2885	†0.5747	†0.3091	†0.5253	†0.4928	†0.3042	†0.5219	†0.5585
ColBERT	W/O Aug	0.0665	0.3157	0.1498	0.3811	0.0679	0.2257	0.1998	0.0654	0.1612	0.2408
	InPars	0.1965	0.5232	0.3031	0.4715	0.1908	0.4618	0.3402	0.2278	0.4645	0.4209
	SPTAR	0.2000	0.5312	†0.3472	†0.5107	0.1923	0.4703	0.3332	0.2338	0.4752	0.4259
BM25BE	W/O Aug	0.1456	0.6225	0.1659	0.4779	0.2330	0.4211	0.4155	0.2623	0.4236	0.5239
	InPars	0.1660	0.6799	0.2891	0.5700	0.2939	0.4905	0.4720	0.2765	0.4286	0.5552
	SPTAR	†0.2159	†0.7222	0.3000	†0.6046	†0.3379	†0.5596	†0.4999	†0.3194	†0.5395	†0.5823
BM25CE	W/O Aug	0.0876	0.5978	0.2319	0.6013	0.2137	0.3108	0.4337	0.1510	0.2263	0.5055
	InPars	0.1889	0.6230	0.3646	0.6155	0.1549	0.2188	0.3732	0.1425	0.1700	0.4780
	SPTAR	†0.2009	†0.7484	0.3705	0.6193	†0.3440⋆	†0.5488	†0.5459⋆	†0.2921	†0.4405	†0.6395⋆
$Contriever_{be}$	W/O Aug	0.1872	0.7228	0.2963	0.5920	0.2876	0.4735	0.4959	0.2944	0.4618	0.5851
	InPars	0.1752	0.7253	0.3541	0.6196	0.2704	0.4782	0.4910	0.2748	0.4443	0.5736
	SPTAR	†0.2148	†0.7717⋆	†0.3836⋆	†0.6567	†0.3284	†0.5272	†0.5402	†0.3325	†0.5241	†0.6267
$ReContriever_{be}$	W/O Aug	0.1938	0.7393	0.3402	0.6212	0.2979	0.4801	0.4913	0.2965	0.4661	0.5902
	InPars	0.1743	0.7367	0.3535	0.6500	0.2914	0.4554	0.5323	0.2849	0.4270	0.5927
	SPTAR	†0.2121	†0.7676	†0.3757	†0.6715⋆	†0.3343	†0.5207	0.5451	†0.3258	†0.5206	†0.6231
$TAS-B$	W/O Aug	0.0297	0.2046	0.1625	0.3903	0.0525	0.1306	0.1312	0.0388	0.0902	0.1115
	InPars	0.2089	0.6911	0.2582	0.5221	0.2828	0.5257	0.4470	0.2982	0.4812	0.5065
	SPTAR	†0.2541⋆	†0.7353	†0.2943	†0.5625	0.3049	†0.5972⋆	0.4737	†0.3488⋆	†0.5612⋆	†0.5690

to prioritize concepts critical to our task, such as query formulation and document enhancement. This suggests that the fine-tuning process successfully adapted the model's parameters to capture task-specific knowledge.

In Fig. 3(c), we aim to understand the effects of different soft prompt lengths on the tuning module by examining the virtual token vector distribution of the learned soft prompt. This experiment was conducted on LLaMA-7B and dataset MS MARCO and all the hyperparameters are the same except for the soft prompt length. The three lengths 40, 50, and 80 are represented by the colors red, blue, and green, respectively. Fig. 3(c) demonstrates the impact of varying soft prompt lengths on the embedding space distribution. Shorter prompts (40 and 50 tokens) produce more compact clusters, while longer prompts (80 tokens) result in a more dispersed distribution. The overlap between clusters suggests that while prompt length affects the range of the embedding space explored, the core task-relevant features remain consistent. This behavior reflects the adaptability of the soft prompt tuning process, where longer prompts enable the model to capture additional nuances while retaining stability in the core representation.

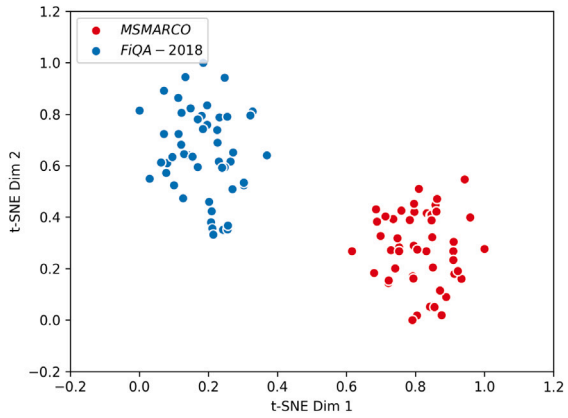
For RQ2, we have conclusions: (a) datasets can be distinguished from the learned soft prompts, demonstrating that soft prompt tuning does learn task-specific soft prompts; (b) both the LLMs and the length of soft prompts influence the learned soft prompts.

Table 8
Efficiency evaluation of SPTAR's soft prompt tuning module on MS MARCO S^{50}_{train} and S^{100}_{eval} (Section 3.1).

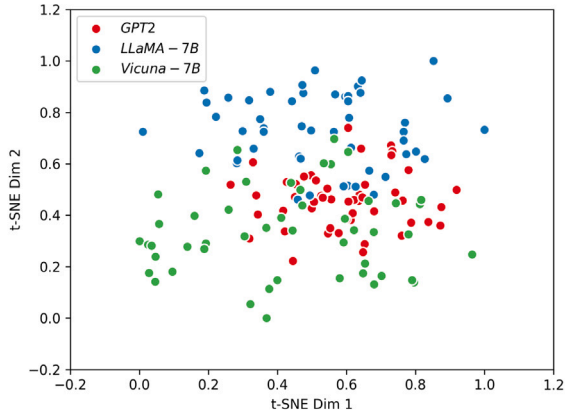
LLM	$count(\theta)/count(\Phi)$	Best epoch #
GPT-2	0.0308%	17
LLaMA-7B	0.0030%	5
Vicuna-7B	0.0030%	4

5.2.2. The efficiency of soft-prompt tuning (RQ3)

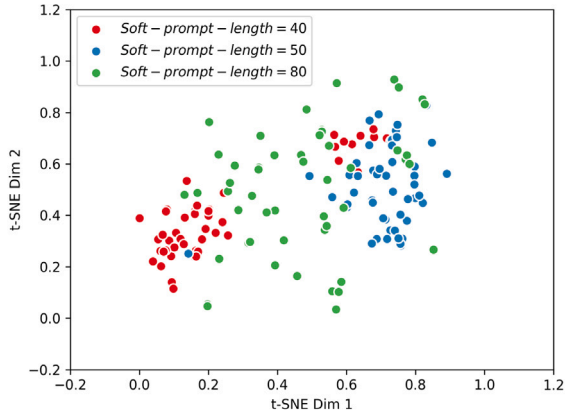
Table 8 presents the number of learnable parameters and convergence efficiency of soft prompt tuning for different LLMs on the MS MARCO dataset. For the soft prompt tuning module in our proposed SPTAR, despite the vast number of LLM's original parameters Φ, Φ remains frozen and does not require fine-tuning. The trainable parameters θ associated with the fine-tuning of the soft prompt are substantially fewer. The percentages in the second column highlight that the soft prompt module's fine-tuning involves an exceptionally small set of parameters θ, roughly equating to **0.003%** of the size of Φ. Notably, the size of θ stays constant, irrespective of the growth 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.



(a) Different datasets.



(b) Different LLMs.



(c) Different lengths.

Fig. 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.

Furthermore, for a new task or dataset, SPTAR can efficiently complete the fine-tuning process of the soft prompt tuning module within a few epochs. As highlighted in the third column of the table, we examined the convergence speed of the soft prompt tuning model on the evaluation dataset S^{100}_{eval} (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.

Table 9
Evaluation of SPTAR-DPR with the best and worst example document–query pairs in soft prompt augmentor module. SPTAR-DPR is trained on $S^{X}_{train} + S^{100}_{eval} + W_{small}$ and tested on D_{test}. Results are obtained on LLaMA-7B. For MS MARCO and FiQA-2018, $M = 2$ and $M = 1$ respectively. Perplexity score is computed on S^{100}_{eval} as described in Section 3.3.

Dataset	Filter	Perplexity (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%)

5.2.3. The impact of soft prompt filter module (RQ4)

With the learned parameters θ^* in SPTAR's soft prompt tuning module, we observe the example document–query pairs in SPTAR's soft prompt augmentor module do influence the quality of the generated weak data, so it is necessary to select certain M document–query pairs from S^{X}_{train}. In this section, we study the impact of SPTAR's soft prompt filter module. In Table 9, 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 $S^{50}_{train} + S^{100}_{eval} + W_{small}$ and tested on D_{test}. The best and worst M example pairs in Table 9 are filtered by the method proposed in Section 3.3.

As shown in Table 9, the results demonstrate that perplexity can be used as a metric to select example document–query pairs. The group of document–query pairs with a lower perplexity score on S^{100}_{eval} shows better NDCG@10 on D_{test}. Additionally, this soft prompt filter module enhances retrieval performance. 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.

5.2.4. The impact of weak data filter module (RQ5)

To assess the enhancements achieved by filtering weak data, we applied various top-k values to filter the generated weak data W_{large}, yielding $F_k(W_{large})$. We then assessed the performance of the SPTAR-DPR model, trained on $S^{50}_{train} + S^{100}_{eval} + F_k(W_{large})$, on D_{test}. This comparison served to quantify the gains over the approach devoid of a weak data filter. Optimal parameters, namely LLM and M, were identified and held constant in this section to exclusively assess the influence of top-k.

As shown in Fig. 4, on MS MARCO, the SPTAR-DPR model without the data filter gets an NDCG@10 score of **0.2319** while the score rises to **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 a data filter. These consistent gains across different datasets underscore the effectiveness of the weak data filter module (Section 3.5). We did not discern any correlation between top-k and the NDCG@10 metric; thus, in real-world scenarios, top-k acts as a hyperparameter requiring tuning per dataset.

Table 10
SPTAR-DPR with different modules. The best scores are underlined.

Retriever	MS MARCO		DL2019			DL2020		
	MRR@10	R@100	MAP	nDCG@10	R@100	MAP	nDCG@10	R@100
W/O Aug	0.1303	0.5141	0.1796	0.3484	0.3289	0.2196	0.3806	0.4218
+ Tuning	0.1507	0.6138	0.2178	0.3799	0.4015	0.2169	0.4017	0.4369
+ Prompt filter	0.1790	0.6566	0.2521	0.4564	0.4444	0.2543	0.4348	0.5199
+ Data filter	0.2114	0.7118	0.3091	0.5253	0.4928	0.3042	0.5219	0.5585

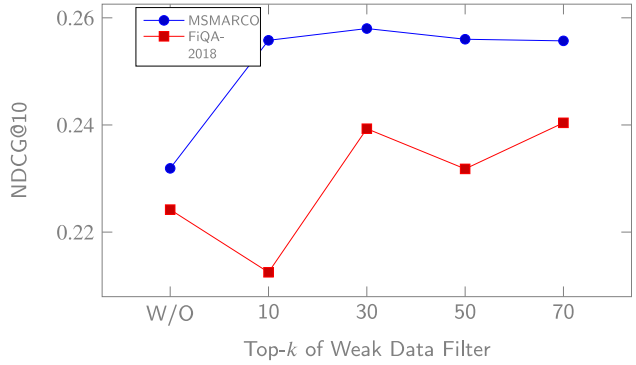


Fig. 4. SPTAR-DPR NDCG@10 scores with different top-k of weak data filter. SPTAR-DPR is trained on $S^{50}_{train} + S^{100}_{eval} + F_k(W_{large})$ (Section 4.1). Results are obtained on LLaMA-7B. For MS MARCO and FiQA-2018, $M = 2$ and $M = 1$ respectively.

5.2.5. The impact of training size X (RQ6)

In this section, we analyze 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^{X}_{train} and evaluated the perplexity on S^{100}_{eval}. Notably, perplexity serves as an intrinsic metric to measure the impact of X on the quality of generated weak queries. We then generated W_{small} and tested the SPTAR-DPR model, trained on W_{small}, on D_{test}. To assess the impact of X on downstream dense retrieval models like DPR, we used NDCG@10, MRR@10, and Recall@100 as evaluation metrics. Initially, the perplexity of LLaMA-7B without soft prompt tuning on S^{100}_{eval} was 1694.2576, but it dramatically decreased to 6.8764 after soft prompt tuning on S^{10}_{train}. As shown in Fig. 5, the perplexity decreases as X increases, with a sharp drop from $X = 10$ to $X = 50$, followed by a slower decline from $X = 50$ to $X = 1000$. The retrieval metrics follow a similar pattern: rapid improvement from $X = 10$ to $X = 50$, stabilizing with some variance from $X = 50$ to $X = 500$. However, all retrieval metrics deteriorate at $X = 1000$, indicating that LLaMA-7B overfitted on S^{100}_{eval} after soft prompt tuning at this scale. Interestingly, it's apparent that enhancing perplexity is more straightforward than improving retrieval metrics, suggesting a disparity between these metrics.

Different from InPars [19] and Promptagator [7], which only utilizes several example document–query pairs in human-written prompts, our findings underscore the benefits of a marginally larger training size X in soft prompt tuning, leading to better performance. This superiority is manifest in the reduced perplexity and the enhanced NDCG@10 scores in downstream tasks with the increment of training size X.

5.2.6. The impact of number of example pairs M (RQ7)

In SPTAR's soft prompt augmentor 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. Initially, we selected LLaMA-7B as the LLM and did soft prompt tuning on S^{50}_{train}, computing perplexity on S^{100}_{eval}. Subsequently, with the filtered M example document–query pairs from SPTAR's soft prompt filter module (Section 3.3), we generated W_{small}. Ultimately, SPTAR-DPR trained on $S^{50}_{train} + S^{100}_{eval} + W_{small}$ 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 the Vicuna-7B model, no matter whether $M = 1$ or $M = 2$. Thus, we only report the results on LLaMA-7B in Fig. 6.

As depicted in Fig. 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. These results contradict our initial assumption that the bigger M is the better the perplexity and NDCG@10 are. We attribute this inconsistency to varying dataset distributions. Considering that many QA datasets contain documents with multiple relevant queries, where each query is constructed from a subset of the document, it implies a heightened level of uncertainty and complexity for the learning model. These intricacies, in turn, produce varying performances across different datasets. Therefore, we acknowledge the importance of delving deeper into this topic in subsequent research.

5.2.7. The impact of each module on retrieval performance (RQ8)

To study how each module contributes to the retrieval performance, we evaluated SPTAR-DPR on MS MARCO, DL 2019, and DL 2020, gradually adding each module. As shown in Table 10, "W/O Aug" represents the performance of SPTAR-DPR trained on $S^{50}_{train} + S^{100}_{eval}$. We then fine-tuned LLaMA using the soft-prompt tuning (Section 3.2) and prompted LLaMA to generate queries for unlabeled documents to get weak dataset $W^{M=0}_{large}$ where $M = 0$ represents there is no example in the prompt. "+Tuning" represents the performance of SPTAR-DPR trained on $S^{50}_{train} + S^{100}_{eval} + W^{M=0}_{large}$. We further adopted the soft prompt filter (Section 3.3) to select two example document–query pairs and applied them in the prompt to regenerate weak dataset $W^{M=2}_{large}$ where $M = 2$ represents there are two examples in the prompt. "+ Prompt Filter" shows the performance of SPTAR-DPR trained on $S^{50}_{train} + S^{100}_{eval} + W^{M=2}_{large}$. Last, we filtered (Section 3.5) $W^{M=2}_{large}$ to get $F_{30}(W^{M=2}_{large})$ and "+Data Filter" denotes the performance of SPTAR-DPR trained on $S^{50}_{train} + S^{100}_{eval} + F_{30}(W^{M=2}_{large})$. We achieved consistent improvements across the three datasets and eight metrics except for DL 2019, where "+ Tuning" is slightly worse than "W/O Aug". These consistent improvements demonstrate the effectiveness of each module in our proposed framework.

6. Conclusion and future work

In this paper, we introduce the soft prompt tuning for augmenting DR (SPTAR) framework to tackle the challenge of limited domain-specific training data in DR tasks. Our approach harnesses soft prompt tuning to optimize soft prompts on limited ground truth data. By prompting LLMs with these optimized soft prompts as well as example document–query pairs, 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 as well as a weak data filter module to clean the generated weak data. The effectiveness of our proposed approach is validated through comprehensive experiments. This work represents an

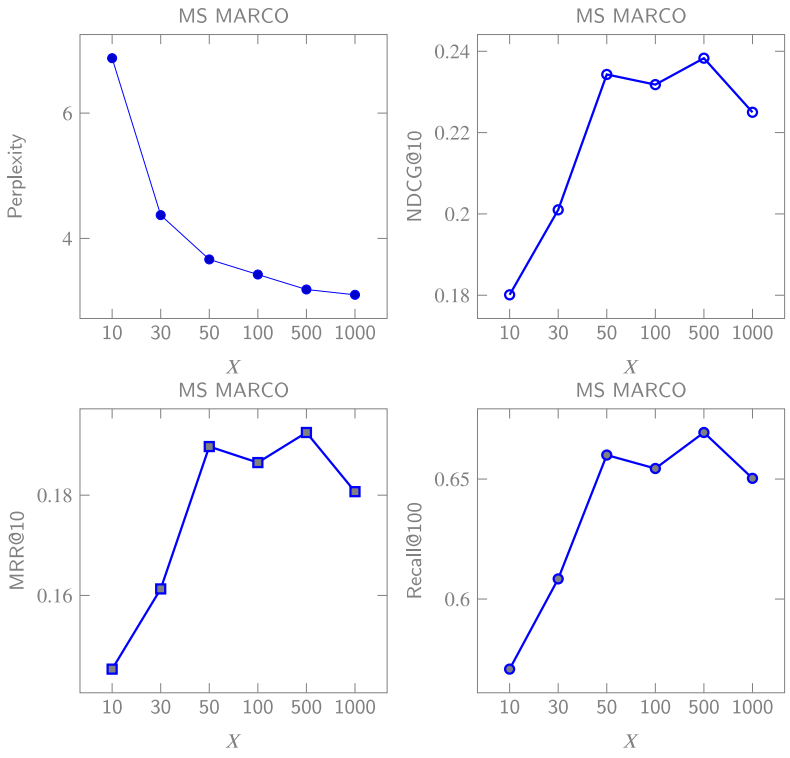


Fig. 5. Evaluation of SPTAR-DPR with different X. SPTAR-DPR is trained on W_{small}. Results are obtained on LLaMA-7B and MS MARCO.

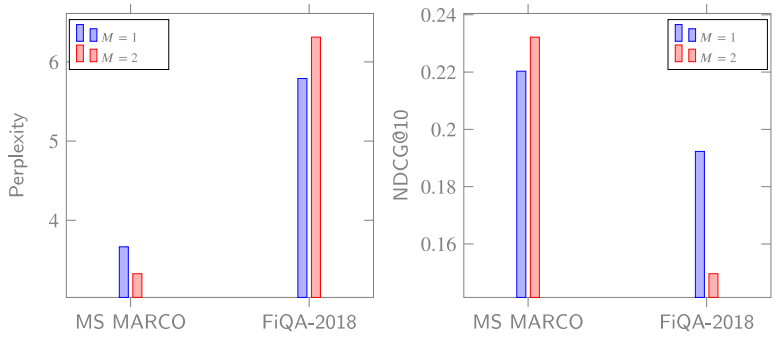


Fig. 6. Evaluation of SPTAR-DPR with different numbers of example pairs M. SPTAR-DPR is trained on $S^{X}_{train} + S^{100}_{eval} + W_{small}$ and tested on D_{test}. Results are obtained on LLaMA-7B and MS MARCO.

initial step toward a promising research direction. In future work,we aim to scrutinize SPTAR's broad applicability by testing it across diverse datasets. It's noteworthy that the loss function employed herein is a pointwise loss, implying a suboptimal utilization of negative instances. Future studies might benefit from delving into pairwise and listwise losses. Moreover, there lies potential in probing multi-task soft prompt tuning methods to bolster both efficiency and outcome.

CRediT authorship contribution statement

Zhiyuan Peng: Writing – original draft, Software, Methodology, Conceptualization. **Xuyang Wu:** Writing – original draft, Software, Methodology, Conceptualization. **Qifan Wang:** Writing – review & editing. **Yi Fang:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Filtered example document–query pairs

See Tables 11 and 12.

Data availability

Data will be made available on request.

Table 11
Filtered example document–query pairs for MS MARCO and LLaMA. $M = 2$ is better.

M	Example
1	**Document**: According to price comparison website Gocompare.com, the cost of becoming a new driver has soared by almost a fifth in the past five years. A survey of 2000 parents found the average cost of a young driver's first car is Â£3825, with insurance priced at Â£2232. Scroll down for video. Expensive: A survey of 2000 parents found the average cost of a young driver's first car is Â£3825. The typical learner also needs Â£480 of driving lessons. Survey of 2000 parents found the average cost of a young driver's first car is Â£3825, with insurance priced at Â£2232. Scroll down for video. Expensive: A survey of 2000 parents found the average cost of a young driver's first car is Â£3825. **Query**: average insurance cost for new drivers
2	**Document**: Oakland weather forecast from AccuWeather.com. Extended forecast in Oakland, MD 21550 for up to 25 days includes high temperature, RealFeel and chance of precipitation Oakland weather forecast from AccuWeather.com. Extended forecast in Oakland, MD 21550 for up to 25 days includes high temperature, RealFeel and chance of precipitation my recent locations Â°f Oakland, MD 41Â° **Query**: weather in Oakland md **Document**: As their name suggests, the triglycerides are composed of one molecule of glycerol and joined via ester bonds with three molecules of fatty acids. As is shown in Figure 12, fatty acids are long chains of carbon and hydrogen usually between 14–24 carbons long (and they always have an even number of carbons). Phospholipids: This class of lipids are really derivatives of triglycerides. The are composed of a glycerol molecule with two fatty acids (a diglyceride). The third carbon contains a phosphate group and usually some added polar molecule (such as ethanolamine, serine or choline). **Query**: what are triglycerides composed of

Table 12
Filtered example document–query pairs for FiQA-2018 and LLaMA. $M = 1$ is better.

M	Example
1	**Document**: As your is a very specific case, please get an advice of CA. It should not cost you much and make it easier. The sale of agriculture land is taxable in certain conditions and exempt from tax in other cases. Sale of agricultural land is subject to capital gains tax. But there are certain exemptions under Section 54B, subject to conditions, which are as follows: If deemed taxable, you can avail indexation, ie the price at which you grandfather got [the date when he inherited it as per indexation] and pay 10% on the difference. If the price is not known, you can take the govt prescribed rate. As there is a large deposit in your fathers account, there can be tax queries and need to be answered. Technically there is no tax liable even if your grandfather gifts the money to your father. More details at http://www.telegraphindia.com/1130401/jsp/business/story_16733007.jsp and http://www.incometaxindia.gov.in/publications/4_compute_your_capital_gains/chapter2.asp **Query**: Is the amount taxable if my grandfather sells agricultural land
2	**Document**: Others have already commented on the impact of anything which dissuades merchants from raising possible breaches, so I wonń dwell on that. Maybe we need stronger legislation, maybe we donń, but it doesnń change todayś answer. Often it works the other way around to what you might expect - rather than the merchant noticing and notifying Visa/MC/others, Visa/MC/others spot patterns of suspicious activity (example 1). I donń have any data on the relative numbers of who is being notified/notifying between merchants and payment processors, but at the point when your card is identified as compromised thereś no reason to suppose that an individual merchant in the traditional sense has been compromised, let alone identified. In fact because thereś a fast moving investigation it could even be a false alarm that led to your card getting cancelled. Conversely it could be a hugely complex multinational investigation which would be jeopardized. Itś simply not safe to assume that simply ""brand X"" has been compromised, therefore everything ""brand X"" knows about you is also compromised: Furthermore thereś no reason to assume the merchant has even admitted to, or discovered the root cause. MC/Visa/Banks, at the point at which theyŕe cancelling cards simply canń say (at least not in a way that might expensively backfire involving lots of lawyers) because the standard of proof needed to go on record blaming someone is simply not yet met. So: yes itś common that you arenń told anything for all of the above reasons. And of course if you really want to find out more you may have some success with your local data protection legislation and formally make a subject access request (or local equivalent) to see what that brings back. Be sure to do it in writing, to the official address of both mastercard and your bank. **Query**: MasterCard won't disclose who leaked my credit card details **Document**: Surprised nobody has mentioned Freshbooks yet. It's lightweight, easy to use, and free for low-end use (scaling up price-wise as you scale up). **Query**: What's the best application, software or tool that can be used to track time?

References

[1] S. Robertson, H. Zaragoza, et al., The probabilistic relevance framework: BM25 and beyond, Found. Trends® Inf. Retr. 3 (4) (2009) 333–389.

[2] A.L. Berger, R. Caruana, D. Cohn, D. Freitag, V.O. Mittal, Bridging the lexical chasm: statistical approaches to answer-finding, in: SIGIR, ACM, 2000, pp. 192–199.

[3] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, 2020, arXiv preprint arXiv:2004.04906.

[4] O. Khattab, M. Zaharia, Colbert: Efficient and effective passage search via contextualized late interaction over bert, in: SIGIR, ACM, 2020, pp. 39–48.

[5] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, M. Zaharia, Colbertv2: Effective and efficient retrieval via lightweight late interaction, 2021, arXiv preprint arXiv:2112.01488.

[6] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. Gurevych, BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021, arXiv preprint arXiv:2104.08663.

[7] Z. Dai, V.Y. Zhao, J. Ma, Y. Luan, J. Ni, J. Lu, A. Bakalov, K. Guu, K.B. Hall, M.-W. Chang, Promptagator: Few-shot dense retrieval from 8 examples, 2022, arXiv preprint arXiv:2209.11755.

[8] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, 2020.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, 2023, arXiv preprint arXiv:2302.13971.

[10] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J.E. Gonzalez, I. Stoica, E.P. Xing, Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality, 2023, URL https://lmsys.org/blog/2023-03-30-vicuna/.

[11] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, T. Goldstein, Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, la, USA, December 10 - 16, 2023, 2023, URL http://papers.nips.cc/paper_files/paper/2023/hash/a00548031e4647b13042c97c922fadf1-Abstract-Conference.html.

[12] B. Lester, R. Al-Rfou, N. Constant, The power of scale for parameter-efficient prompt tuning, in: EMNLP, Association for Computational Linguistics, 2021, pp. 3045–3059.
[13] X.L. Li, P. Liang, Prefix-tuning: Optimizing continuous prompts for generation, in: ACL/IJCNLP, Association for Computational Linguistics, 2021, pp. 4582–4597.
[14] F. Ma, C. Zhang, L. Ren, J. Wang, Q. Wang, W. Wu, X. Quan, D. Song, Xprompt: Exploring the extreme of prompt tuning, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, Association for Computational Linguistics, 2022, pp. 11033–11047, URL https://preview.aclanthology.org/emnlp-22-ingestion/2022.emnlp-main.758.pdf.
[15] L. Yan, C. Han, Z. Xu, D. Liu, Q. Wang, Prompt learns prompt: Exploring knowledge-aware generative prompt collaboration for video captioning, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, ijcai.org, 2023, pp. 1622–1630, http://dx.doi.org/10.24963/ijcai.2023/180, URL https://doi.org/10.24963/ijcai.2023/180.
[16] T. Schick, H. Schütze, Exploiting cloze-questions for few-shot text classification and natural language inference, in: P. Merlo, J. Tiedemann, R. Tsarfaty (Eds.), Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021, Association for Computational Linguistics, 2021, pp. 255–269, http://dx.doi.org/10.18653/V1/2021.EACL-MAIN.20.
[17] T. Schick, H. Schütze, It's not just size that matters: Small language models are also few-shot learners, in: K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou (Eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, Association for Computational Linguistics, 2021, pp. 2339–2352, http://dx.doi.org/10.18653/V1/2021.NAACL-MAIN.185.
[18] T.L. Scao, A.M. Rush, How many data points is a prompt worth? 2021, arXiv preprint arXiv:2103.08493.
[19] L.H. Bonifacio, H. Abonizio, M. Fadaee, R.F. Nogueira, InPars: Unsupervised dataset generation for information retrieval, in: SIGIR, ACM, 2022, pp. 2387–2392.
[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (1) (2020) 5485–5551.
[21] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, S.R. Bowman, SuperGLUE: A stickier benchmark for general-purpose language understanding systems, in: NeurIPS, 2019, pp. 3261–3275.
[22] E. Perez, D. Kiela, K. Cho, True few-shot learning with language models, in: NeurIPS, 2021, pp. 11054–11070.
[23] R. Pradeep, S. Sharifymoghaddam, J. Lin, RankVicuna: Zero-shot listwise document reranking with open-source large language models, 2023, http://dx.doi.org/10.48550/arXiv.2309.15088, CoRR abs/2309.15088 arXiv:2309.15088 URL https://doi.org/10.48550/arXiv.2309.15088.
[24] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, A. Overwijk, Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020, arXiv preprint arXiv:2007.00808.
[25] S. Hofstätter, S. Lin, J. Yang, J. Lin, A. Hanbury, Efficiently teaching an effective dense retriever with balanced topic aware sampling, in: F. Diaz, C. Shah, T. Suel, P. Castells, R. Jones, T. Sakai (Eds.), SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, ACM, 2021, pp. 113–122, http://dx.doi.org/10.1145/3404835.3462891.
[26] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, E. Grave, Unsupervised dense information retrieval with contrastive learning, Trans. Mach. Learn. Res. 2022 (2022) URL https://openreview.net/forum?id=jKN1pXi7b0.
[27] Y. Lei, L. Ding, Y. Cao, C. Zan, A. Yates, D. Tao, Unsupervised dense retrieval with relevance-aware contrastive pre-training, in: A. Rogers, J.L. Boyd-Graber, N. Okazaki (Eds.), Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023, Association for Computational Linguistics, 2023, pp. 10932–10940, http://dx.doi.org/10.18653/V1/2023.FINDINGS-ACL.695.
[28] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, in: NeurIPS, 2020.
[29] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, MS MARCO: A human generated machine reading comprehension dataset, choice 2640 (2016) 660.
[30] R. Nogueira, Z. Jiang, J. Lin, Document ranking with a pretrained sequence-to-sequence model, 2020, arXiv preprint arXiv:2003.06713.
[31] V. Jeronymo, L.H. Bonifacio, H. Abonizio, M. Fadaee, R.d. Lotufo, J. Zavrel, R.F. Nogueira, InPars-v2: Large language models as efficient dataset generators for information retrieval, 2023, CoRR abs/2301.01820.
[32] L. Boytsov, P. Patel, V. Sourabh, R. Nisar, S. Kundu, R. Ramanathan, E. Nyberg, InPars-Light: Cost-effective unsupervised training of efficient rankers, 2023, CoRR abs/2301.02998.
[33] J. Wei, M. Bosma, V.Y. Zhao, K. Guu, A.W. Yu, B. Lester, N. Du, A.M. Dai, Q.V. Le, Finetuned language models are zero-shot learners, 2021, arXiv preprint arXiv:2109.01652.
[34] S. Jeong, J. Baek, S. Cho, S.J. Hwang, J.C. Park, Augmenting document representations for dense retrieval with interpolation and perturbation, in: S. Muresan, P. Nakov, A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022, Association for Computational Linguistics, 2022, pp. 442–452, http://dx.doi.org/10.18653/v1/2022.acl-short.48.
[35] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, W.X. Zhao, D. Dong, H. Wu, H. Wang, RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering, in: K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou (Eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, Association for Computational Linguistics, 2021, pp. 5835–5847, http://dx.doi.org/10.18653/v1/2021.naacl-main.466.
[36] R. Ren, Y. Qu, J. Liu, W.X. Zhao, Q. She, H. Wu, H. Wang, J. Wen, RocketQAv2: A joint training method for dense passage retrieval and passage Re-ranking, in: M. Moens, X. Huang, L. Specia, S.W. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, Association for Computational Linguistics, 2021, pp. 2825–2835, http://dx.doi.org/10.18653/v1/2021.emnlp-main.224.
[37] S. Lin, A. Asai, M. Li, B. Oguz, J. Lin, Y. Mehdad, W. Yih, X. Chen, How to train your DRAGON: Diverse augmentation towards generalizable dense retrieval, 2023, http://dx.doi.org/10.48550/arXiv.2302.07452, CoRR abs/2302.07452 arXiv:2302.07452 URL https://doi.org/10.48550/arXiv.2302.07452.
[38] D.S. Sachan, M. Lewis, M. Joshi, A. Aghajanyan, W.-t. Yih, J. Pineau, L. Zettlemoyer, Improving passage retrieval with zero-shot question generation, 2022, arXiv preprint arXiv:2204.07496.
[39] L. Gao, X. Ma, J. Lin, J. Callan, Precise zero-shot dense retrieval without relevance labels, 2022, CoRR abs/2212.10496.
[40] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C.D. Manning, C. Ré, D. Acosta-Navas, D.A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L.J. Orr, L. Zheng, M. Yüksekgönül, M. Suzgun, N. Kim, N. Guha, N.S. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S.M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, Y. Koreeda, Holistic evaluation of language models, 2022, CoRR abs/2211.09110.
[41] X. Ma, X. Zhang, R. Pradeep, J. Lin, Zero-shot listwise document reranking with a large language model, 2023, arXiv preprint arXiv:2305.02156.
[42] W. Sun, L. Yan, X. Ma, P. Ren, D. Yin, Z. Ren, Is ChatGPT good at search? Investigating large language models as re-ranking agent, 2023, arXiv preprint arXiv:2304.09542.
[43] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, M. Bendersky, Large language models are effective text rankers with pairwise ranking prompting, 2023, CoRR arXiv:2306.17563.
[44] W.L. Tam, X. Liu, K. Ji, L. Xue, X. Zhang, Y. Dong, J. Liu, M. Hu, J. Tang, Parameter-efficient prompt tuning makes generalized and calibrated neural text retrievers, 2022, arXiv preprint arXiv:2207.07087.
[45] J. Mu, X.L. Li, N.D. Goodman, Learning to compress prompts with gist tokens, 2023, CoRR abs/2304.08467.
[46] J. Zhou, L. Tian, H. Yu, Z. Xiao, H. Su, J. Zhou, Dual context-guided continuous prompt tuning for few-shot learning, in: ACL, Association for Computational Linguistics, 2022, pp. 79–84.
[47] Z. Wang, R. Panda, L. Karlinsky, R. Feris, H. Sun, Y. Kim, Multitask prompt tuning enables parameter-efficient transfer learning, in: ICLR, OpenReview.net, 2023.
[48] Z. Huang, H. Zeng, H. Zamani, J. Allan, Soft prompt decoding for multilingual dense retrieval, 2023, CoRR abs/2305.09025.
[49] Z. Tang, B. Wang, T. Yao, DPTDR: Deep prompt tuning for dense passage retrieval, in: COLING, International Committee on Computational Linguistics, 2022, pp. 1193–1202.
[50] L. Yang, Q. Wang, J. Wang, X. Quan, F. Feng, Y. Chen, M. Khabsa, S. Wang, Z. Xu, D. Liu, MixPAVE: Mix-prompt tuning for few-shot product attribute value extraction, in: Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics, ACL 2023, Association for Computational Linguistics, 2023.
[51] C. Han, Q. Wang, Y. Cui, Z. Cao, W. Wang, S. Qi, D. Liu, E^2vpt: An effective and efficient approach for visual prompt tuning, 2023, http://dx.doi.org/10.48550/arXiv.2307.13770, CoRR arXiv:2307.13770 arXiv:2307.13770 URL https://doi.org/10.48550/arXiv.2307.13770.
[52] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, PEFT: State-of-the-art parameter-efficient fine-tuning methods, 2022, https://github.com/huggingface/peft.
[53] M. Maia, S. Handschuh, A. Freitas, B. Davis, R. McDermott, M. Zarrouk, A. Balahur, Www'18 open challenge: Financial opinion mining and question answering, in: WWW, ACM, 2018, pp. 1941–1942.

[54] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E.M. Voorhees, Overview of the TREC 2019 deep learning track, 2020, CoRR abs/2003.07820 arXiv:2003.07820 URL https://arxiv.org/abs/2003.07820.

[55] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, Overview of the TREC 2020 deep learning track, in: E.M. Voorhees, A. Ellis (Eds.), Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020, in: NIST Special Publication, Vol. 1266, National Institute of Standards and Technology (NIST), 2020, URL https://trec.nist.gov/pubs/trec29/papers/OVERVIEW.DL.pdf.

[56] J. Lin, M. Crane, A. Trotman, J. Callan, I. Chattopadhyaya, J. Foley, G. Ingersoll, C. MacDonald, S. Vigna, Toward reproducible baselines: The open-source IR reproducibility challenge, in: ECIR, in: Lecture Notes in Computer Science, Vol. 9626, Springer, 2016, pp. 408–420.

[57] J. Lin, X. Ma, S. Lin, J. Yang, R. Pradeep, R.F. Nogueira, Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations, in: F. Diaz, C. Shah, T. Suel, P. Castells, R. Jones, T. Sakai (Eds.), SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, ACM, 2021, pp. 2356–2362, http://dx.doi.org/10.1145/3404835.3463238.

[58] E. Kamalloo, N. Thakur, C. Lassance, X. Ma, J.-H. Yang, J. Lin, Resources for brewing BEIR: Reproducible reference models and statistical analyses, 2024.

[59] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).