Pseudo-Relevance for Enhancing Document Representation

Jihyuk Kim
Yonsei University
jihyukkim@yonsei.ac.kr

Seung-won Hwang*
Seoul National University
seungwonh@snu.ac.kr

Seoho Song and **Hyeseon Ko,** and **Young-In Song**
NAVER Corp
{song.seoho, hyeseon.ko, song.youngin}@navercorp.com

Abstract

This paper studies how to enhance the document representation for the bi-encoder approach in dense document retrieval. The bi-encoder, separately encoding a query and a document as a single vector, is favored for high efficiency in large-scale information retrieval, compared to more effective but complex architectures. To combine the strength of the two, the multi-vector representation of documents for bi-encoder, such as ColBERT preserving all token embeddings, has been widely adopted. Our contribution is to reduce the size of the multi-vector representation, without compromising the effectiveness, supervised by query logs. Our proposed solution decreases the latency and the memory footprint, up to 8- and 3-fold, validated on MSMARCO and real-world search query logs[1].

1 Introduction

Information Retrieval (IR) aims to rank documents by their relevance scores to the given query, where neural IR models have achieved state-of-the-art performances by fine-tuning large language models. These models, encoding document d to query q into vectors for soft matching, can be categorized into **cross-encoder** jointly encoding q and d, and **bi-encoder** separately encoding the two into single vectors.

Our target scenario is commercial-scale IR, requiring online query latency to a web-scale corpus, where bi-encoders are generally preferred for lower latency. Meanwhile, as a hybrid of the two, a single vector representation can be enriched by multi-vector representation, such as ColBERT (Khattab and Zaharia, 2020) preserving all token embeddings, for higher effectiveness. With multi-vector bi-encoder, once query and document vectors are indexed on memory, relevant documents can be efficiently identified, by a nearest neighbor (NN) search on the index (Johnson et al., 2019). ColBERT, by fully preserving all tokens in document representation, sacrifices **on-memory efficiency**, for enriching document representation.

	query-aware (cost: H/L)	query-agnostic
document	**Ours** (L) IDCM (H)	D-Cluster ME-BERT
query	ColBERT-PRF (H)	

Table 1: Taxonomy of query and document representation approaches.

Our research question is thus: Can we improve memory efficiency, without compromising the effectiveness of ColBERT? To answer this question, our hypothesis is that: Given the differential contribution of terms in a document regarding relevant queries, a few query-relevant terms may be enough to match the queries, and non-relevant terms can be pruned out, to decrease index overhead. The closest existing work is **D-Cluster** (Tang et al., 2021) [2], where k centroids of document tokens are considered relevant, and their token embeddings are preserved, as highly relevant terms. This work is categorized as "query-agnostic" in Table 1, as actual queries formulated for this (or, similar) document in the training dataset, are not used.

Our distinction is selecting k terms with query-awareness. Existing query-aware term selection, such as **IDCM** (Hofstätter et al., 2021), waits for users to formulate query, then select query-relevant terms, though incurring high query latency. Another example is pseudo-relevance feedback (PRF): For the related goal of selecting terms to expand queries, **ColBERT-PRF** (Wang et al., 2021b) also waits for the query. Both leverage relevance feedbacks from user query or PRF, respectively, but

*Corresponding author.
[1] https://github.com/jihyukkim-nlp/PQA-ColBERT

[2] Since the authors did not provide the official name for the proposed method, we denote as "D-Cluster" for Document-based Clustering

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11639 - 11652
December 7-11, 2022

incur prohibitive online overhead for our target scenario of commercial-scale IR.

Our distinction is to capture query-aware relevance at indexing time, not to incur evaluation-time overhead. A key distinction is using a pseudo-query, by supervising a pseudo-query extractor from query logs, in contrast to query-agnostic unsupervised approaches, *e.g.*, D-Cluster. This can be viewed as finding a low-cost solution in the query-aware space– marked as **Ours** (L) in Table 1.

Our proposed pseudo-query extractor, following the intuition of D-Cluster and PRF, treats a subset of document as candidates of a pseudo-query. Our distinction is, based on q-d pairs seen from query log, to supervise such extraction, during indexing time, for which we address three research questions as follows:

RQ 1. How can we obtain supervision for training a pseudo-query extractor? To answer RQ 1, we should address the limitation of most benchmark datasets (e.g., MSMARCO), annotating relevance coarsely, at document-level. Though some dataset annotates term-level relevance (Hofstätter et al., 2020), its scale is often limited due to annotation overhead, and often not sufficient for training and used only for evaluation. Our contribution is obtaining term-level supervision from the coarse labels, by utilizing the late interaction layer of ColBERT which measures the contribution of each term to document-level relevance.

RQ 2. Would pseudo-query extractor preserve relevance? Our hypothesis, stated as RQ 2, is that only a few terms in each document contribute to query-document relevance, which we extract as "pseudo-queries". Our contribution is guiding document encoder before indexing, to keep only the terms relevant to pseudo-queries (as done in cross-encoder with real queries). Our pseudo-query extractor is thus trained to closely predict the relevance with the real query. We validated our hypothesis in both public benchmark and real-life search engine.

RQ 3. Finally, would pseudo-query extractor generalize? Though increasing efficiency, pseudo-query terms extracted *at indexing phase* without query, may overfit to seen query-document pairs in the training dataset. We validate ours generalizes well for unseen queries and documents, by comparing term-level relevance score predicted with manual annotation, and also automatically comparing scores with those from oracle and generalizable heuristics.

We show ours contributes to perform comparably to ColBERT-PRF, with 8-fold and 3-fold decrease in latency and memory footprint, using both public benchmark MSMARCO passage ranking tasks and real-world search engine queries.

2 Preliminaries and Related Work

This section motivates the importance of query-awareness for document understanding, then compares and contrasts with existing approaches.

2.1 Motivation

q	How long is the **flight** from **Chicago** to **Cairo**?
d	... total **flight duration** from **Chicago, IL** to **Cairo**, **Egypt** is 12 hours, 47 minutes. This assumes an average flight speed for a commercial airliner of ...

Table 2: A running example. **Bold-faced** terms denote relevant terms between q-d.

In Table 2, we present, as a running example, a relevant q-d pair from MSMARCO dataset. When not knowing a relevant query (e.g., during indexing), the entire document terms are equally important in representation. However, knowing the query, we can identify highly relevant terms in the document, e.g., **bold-faced** terms in Table 2. Our distinction is (a) identifying highly relevant terms before query is known, for (b) enriching document encoding to match query better.

2.2 Related Work

With the goal of capturing relevant terms between q-d, our work is built upon relevance model (Lavrenko and Croft, 2017), where the underlying relevance model R generates terms in both q and d by modeling the probability on each term w, i.e., $p(w|R)$. Since the true relevance model R is unknown, R is approximated as $\tilde{R}$. Depending on target scenarios, w can be either added to the given query (§2.2.1) or extracted from each document in a corpus (§2.2.2), and correspondingly the approximation $\tilde{R}$ also differs.

2.2.1 Adding w to q

As q is often short, relevant terms may be absent from q. Query expansion aims to expand q, where missing relevant terms can be sampled from $p(w|R)$.

Given q, to approximate $p(w|R)$, pseudo-relevance feedback documents can be leveraged as $\tilde{R}$:

$$p(w|R) \approx \sum_{d' \in \tilde{R}} p(w, d'|q). \quad (1)$$

For example, ColBERT-PRF first retrieves top-ranked documents by ColBERT (Khattab and Zaharia, 2020) retriever as the pseudo-relevance feedback, then finds the expansion terms as centroid terms in the documents, by applying k-means clustering on all token embeddings. Once augmenting q with the expansion terms, ColBERT-PRF repeats the same retrieval, but now using a better query.

While improving performance, this sacrifices efficiency, as it requires two rounds of retrieval using the given q and the expanded q, respectively. Instead, with the goal of devising an efficient retriever, we adopt the relevance model for *term extraction from* d. Different from queries, long documents often produce high recall but low precision on relevant terms. Therefore, in this work, we aim to increase precision by pruning non-relevant terms in d, so as to improve the efficiency with little sacrifice on the effectiveness.

2.2.2 Extracting w from d

For extracting w from d, we categorize two scenarios depending on whether q is given or not.

Query-agnostic selection When targeting extraction at indexing time, q is not given, such that R can be approximated by $\tilde{R} = \{d\}$, i.e., the singleton of d, and terms that summarize $\tilde{R}$ can can get high probability. For example, ME-BERT (Luan et al., 2021) selects first-k terms, by leveraging positional bias (Hofstätter et al., 2020) which has been effective for extractive summarization tasks (See et al., 2017). As an another example, D-Cluster (Tang et al., 2021) selects centroid terms, by applying k-means clustering on document term vectors.

Alternatively, R can be approximated by the entire collection, so that terms that discriminate relevant documents well can get high probability. For this purpose, inverse document frequency (IDF) (Lassance et al., 2022) can be used, by leveraging the fact that rare query terms can discriminate the given document from most of the other documents.

Query-aware selection Query-agnostic approach would select terms that are uniformly relevant to the entire document terms. For example, among bold terms in Table 2, query-agnostic approach can select *flight duration* which is related to many terms in d, e.g., *flight*, *hours*, *minutes*, *speed*, while cannot select terms like *Cairo, Egypt*.

This motivates query-aware selection, delaying term selection until evaluation time, when q is given. Targeting second-stage re-ranking for document ranking tasks, IDCM (Hofstätter et al., 2021) first selects a few relevant passages in a document to the given query using a bi-encoder, and then, computes the relevance of the document using a cross-encoder based on only the selected passages. While having an advantage of query-awareness, this sacrifices evaluation-time efficiency, violating our goal. Our distinction is introducing query-awareness to the indexing-time, so as to achieve efficiency. We argue that indexing-time selection suffices when properly trained via our proposed supervision, and empirically show that our proposed retriever achieves comparable performance to IDCM with significantly lower latency.

2.2.3 Application of identifying w from d

Though our primary goal is to improve the ranking effectiveness, identifying query terms in d can also help document owners with better organizing d by envisioning relevant queries in d. For example, Pickens et al. (2010) analyze the retrievability of documents (Azzopardi and Vinay, 2008) via reverted indexing which is a query-centric view of inverted indexing. Aiming to increase the transparency of search engines, Li et al. (2022) study identifying exposing queries for documents by which the document creators better understand search queries that surface their documents. As in the prior work, our pseudo-query extractor can improve the transparency of our search engine by providing pseudo-query terms (which are readable by humans in contrast to dense vectors in D-Cluster), and potentially improve the retrievability by enabling the document owners to modify contents to expose query terms of their interest in their documents. Different from the previous studies targeting mainly document owners, our distinction is increasing the ranking performance of search engines, to also satisfy users of search engines.

3 Proposed: Query-Aware Selection at Indexing Time

To enable query-aware selection at indexing time, we leverage existing relevant query logs for each d. However, in practice, most of the documents

in a corpus do not have query logs. Therefore, we train a pseudo-query extractor to predict relevant query terms from d, by designing supervision for the training based on the relevant q-d pairs from the query logs. At indexing time, the pseudo-query extractor selects relevant terms from both seen and unseen documents, by envisioning real, relevant queries.

Our extractor and retriever are built upon ColBERT (§3.1) which models term-level relevance between q-d. By leveraging soft matches between terms in q/d, we obtain supervisions on $p(w|R)$ for training pseudo-query extractor (§3.2).

3.1 ColBERT

ColBERT preserves all token embeddings to measure the relevance score between q-d, denoted by $\text{rel}(q, d)$, as sum of maximum similarity of each query token q_j over document token d_i:

$$\mathbf{h}_i^d = \mathbf{W}^\top \text{BERT}(\{\mathbf{x}_i^d + \mathbf{p}_i + \mathbf{e}_0\}_{i=1}^{|d|}) \quad (2)$$

$$\mathbf{h}_j^q = \mathbf{W}^\top \text{BERT}(\{\mathbf{x}_j^q + \mathbf{p}_j + \mathbf{e}_0\}_{j=1}^{|q|}) \quad (3)$$

$$\text{rel}(q, d) = \sum_{j \in [1,|q|]} \max_{i \in [1,|d|]} \mathbf{h}_i^{d\top} \mathbf{h}_j^q, \quad (4)$$

where $\mathbf{x}_{[*]}^{[q/d]} \in \mathbb{R}^{768}$, $\mathbf{p}_{[*]} \in \mathbb{R}^{768}$, and $\mathbf{e}_0 \in \mathbb{R}^{768}$ denote token, position, and segment embedding in BERT, respectively, $\mathbf{W} \in \mathbb{R}^{768\times128}$ transforms 768-dimension BERT features into 128-dimension features for efficiency, and $|q|$, $|d|$ denote the number of tokens in q and d respectively. Note that, the max-pooling in Eq 4 plays an important role in increasing ranking performance. For example, replacing max-pooling by mean-pooling significantly decreases the ranking performance (Khattab and Zaharia, 2020), which motivates our approach of keeping only a few highly relevant terms.

During training, $\mathbf{W}$ and parameters for BERT are optimized to maximize $\text{rel}(q, d^+)$ of an annotated relevant document d^+ and minimize the scores of a set of non-relevant documents $\{d^-\}$, by using noise-contrastive learning objective $\mathcal{L}$ (Mnih and Kavukcuoglu, 2013):

$$\mathcal{L} = -\log \frac{e^{\text{rel}(q,d^+)}}{e^{\text{rel}(q,d^+)} + \sum_{d^-} e^{\text{rel}(q,d^-)}}. \quad (5)$$

Though most of the documents in a corpus are non-relevant to q, to avoid trivial negatives, we use top-1000 ranked documents by BM25 as hard negatives, and for sample-efficient training, we also use in-batch negatives (Karpukhin et al., 2020).

3.2 Ours: Pseudo-Query-Aware ColBERT

Our distinction is to preserve only highly relevant terms that much contribute to $\text{rel}(q, d)$ in Eq 4, so as to reduce memory overhead of index and online evaluation latency, without losing effectiveness. To capture the relevant terms, we introduce a pseudo-query extractor and how we supervise the extractor to predict term score b_i for each document term d_i, by which the pseudo-query terms can be extracted as top-k terms according to b_i. We leverage the extracted pseudo-query terms for both **p**seudo-**q**uery-**a**ware relevance modeling, or **PQA-Relevance** and encoding d, or **PQA-Encoding**. As we integrate our pseudo-query extractor with ColBERT, we henceforth denote our retriever by **P**seudo-**Q**uery-**A**ware **ColBERT**, or PQA-ColBERT.

Pseudo-query extractor Given only the document-level annotations on relevance, it is non-trivial to obtain supervision for training the pseudo-query extractor. To obtain term-level supervision, we propose to leverage term-level soft matches of ColBERT. Specifically, based on the observation that only a few terms in d^+ contribute to $\text{rel}(q, d^+)$ after max-pooling in Eq 4, we treat those terms, denoted by $\bar{q}^*$, as pseudo-query terms of d^+. During training, our pseudo-query extractor is supervised to assign $p(w|R) = 1$ for any term w in $\bar{q}^*$ and $p(w|R) = 0$ for the others. Note that, $\bar{q}^*$ fully preserves relevance of d^+ to q, as $\text{rel}(q, d^+)$ is unchanged once we retain $\bar{q}^*$ from d^+. When multiple q are annotated to d, we take the union of the $\bar{q}^*$ for each q, to obtain the supervision on $p(w|R)$.

Given this supervision, we train a prediction layer f that predicts the probability b_i for each d_i^+ to be $\bar{q}^*$: $b_i = \sigma(f(\mathbf{h}_i^{d^+}))$, where σ denotes the sigmoid function and $f(\cdot)$ consists of two fully-connected layers with ReLU (Hahnloser et al., 2000) activation function. As the objective function, we employ binary cross entropy on b_i.

During inference, for each d in the corpus, we retain top-k terms regarding b, *i.e.*, pseudo-query terms $\bar{q}^d = \{\bar{q}_l^d\}_{l=1}^k$. In the following paragraphs, we propose to leverage $\bar{q}^d$ for both efficient relevance modeling and effective encoding.

PQA-Relevance While building upon ColBERT, our distinction is enabling efficient indexing and relevance modeling, where only $\{\bar{q}_l^d\}_{l=1}^k$ ($k \ll |d|$) are involved in $\text{rel}(q, \cdot)$, and accordingly in index-

ing, that is, $\text{rel}(q, d)$ is approximated by $\text{rel}(q, \overline{q}^d)$:

$$\text{rel}(q, d) \approx \text{rel}(q, \overline{q}^d) \quad (6)$$

$$= \sum_{j \in [1, |q|]} \max_{d_i \in \{\overline{q}_l^d\}_{l=1}^k} \mathbf{h}_i^{d\top} \mathbf{h}_j^q, \quad (7)$$

Our results in §4 show that ours incurs 1/3 of index memory overhead and evaluation latency of ColBERT-PRF, but perform comparably to ColBERT-PRF.

PQA-Encoding While adopting the bi-encoder design, our distinction is reflecting differential contribution of each d_i during encoding d, to give more emphasis on query-relevant terms, as in cross-encoder approaches.

Analogous to cross-encoder approaches, we provide our encoder with query context, as if q is given via $\overline{q}^d$. Specifically, we add a binary indicator $s_i = \mathbb{1}(d_i \in \overline{q}^d)$, where $s \in \{0, 1\}$, to segment embeddings in BERT by replacing $\mathbf{e}_0$ in Eq 2 with $\mathbf{e}_{s_i}$. By differentiating $\overline{q}^d$ and the other tokens in d through $\mathbf{e}_{s_i}$, the document encoder can give more emphasis on $\overline{q}^d$ which has full responsibility for relevance prediction (Eq 7).

For training PQA-Encoder, our goal is to fully preserve q-d relevance using only $\overline{q}^d$ in d. We thus design the training objective as the degree of preserving $\text{rel}(q, d)$ (Eq 4) with $\text{rel}(q, \overline{q}^d)$ (Eq 7), which can be implemented by Kullback–Leibler divergence with normalized relevance scores:

$$\mathcal{L} = D_{\text{KL}}(p_{\text{rel}(q,d)} || p_{\text{rel}(q,\overline{q}^d)}), \quad (8)$$

where $p_{\text{rel}(q,\cdot)} = \text{softmax}(\text{rel}(q, \cdot))$.

4 Experiments

In this section, we empirically validate the effectiveness of our PQA-ColBERT, with respect to ranking performance on passage ranking task. We report results for a single run.

4.1 Dataset

We train and evaluate our PQA-ColBERT using benchmark datasets MSMARCO (Nguyen et al., 2016) and TREC-DL (Craswell et al., 2020, 2021)[3], on both passage ranking and document ranking tasks[4]. For training, we use MSMARCO datasets, and, for evaluation, both MSMARCO and TREC-DL datasets.

In addition, we construct **RealQ**, where relevant q-d pairs are extracted from query clicks from real users. By not requiring human annotation, this dataset increases test queries by 100-fold from MSMARCO. Also, it captures realistic queries that are much shorter than queries from MSMARCO, where the average length is 3.15 and 6.37 for RealQ and MSMARCO, respectively. We leave details of the dataset construction process in A.1.

Finally, to further analyze the generalization ability to unseen q/d, we also evaluate the zero-shot ranking performance: Retrievers are trained on the MSMARCO Passage Ranking dataset and evaluated on TREC-COVID (Voorhees et al., 2021) dataset which contains medical or pathological q/d terms and thus is used as an out-of-domain evaluation dataset.

As evaluation metrics, for MSMARCO and RealQ that provide the binary relevance, we report MRR@10 on top-10 ranked documents. For TREC-DL datasets and TREC-COVID dataset that provide graded relevance ranging from 1 to 4, we report NDCG@10, following the convention of original paper for graded relevance. For other datasets with binary annotation, we follow their convention to report Recall@1000 (denoted by R@1k) on top-1000 ranked documents. To compare efficiency, we also report index size and average latency per query.

4.2 Baselines

To validate the effectiveness of our pseudo-query-aware indexing, we compare our method to existing low-cost retrievers that use either single vector or multiple vectors for each document, along with high-cost retrievers such as ColBERT (Khattab and Zaharia, 2020) and ColBERT-PRF (Wang et al., 2021b) that use all document terms for indexing.

As a baseline that uses single-embedding, we adopt ANCE (Xiong et al., 2021) that uses `[CLS]` token embedding from BERT encoder. For baselines that use multi-embeddings, we adopt ME-BERT (Luan et al., 2021) that uses first k tokens embeddings, and D-Cluster (Tang et al., 2021) that uses centroid embeddings from k-means clustering on token embeddings in a given passage. For a fair comparison to D-Cluster, we set $k = 24$ for the passage ranking task and $k = 48$ for the document ranking task, to have the same index size.

[3] MSMARCO and TREC-DL are intended for non-commercial research purposes. Please refer to https://microsoft.github.io/msmarco/ for further details.

[4] In the following sections, for simplicity, we use the term "document" to denote either "passage" or "document", if not explicitly mentioned.

	Retriever	MSMARCO Dev		TREC-DL 2019		TREC-DL 2020		Index size	Latency
		MRR@10	R@1k	NDCG@10	R@1k	NDCG@10	R@1k	(GB)	(ms)
H	ColBERT	36.0	96.8	69.3	78.9	68.7	82.5	143	338
H	ColBERT-PRF	35.5	**97.3**	**74.1**	79.8	**71.6**	84.4	143	4132
H	PQA-ColBERT-H (ours)	**37.0**	97.2	69.6	**84.9**	68.1	**85.7**	186	380
L	ANCE	33.0	95.9	64.8	75.5	64.6	77.6	13	< 50
L	ME-BERT	33.4	85.5	68.7	-	-	-	32	< 50
L	D-Cluster	34.5	96.4	-	-	-	-	43	< 50
L	ColBERT-First	35.7	**96.7**	-	-	-	-	103	73
L	ColBERT-IDF	35.5	**96.7**	-	-	-	-	102	73
L	PQA-ColBERT (ours)	**36.7**	96.5	**69.6**	**81.8**	**68.3**	**82.6**	43	40

Table 3: End-to-end retrieval performance on passage ranking datasets. **Bold-face** indicates the best performance for high- (H) and low-cost (L) approaches, respectively. We avoid reporting exact latency, denoted as "< 50 (ms)", of those having different inference setup from ColBERT, but it was comparable to ours, and significantly lower than ColBERT.

For example, for the passage ranking task, since D-Cluster uses four centroid vectors for each passage having 768 dimension per centroid, $k = 24$ matches the size of entire document embeddings (*i.e.*, $4 \times 768 = 24 \times 128$). Similarly, since D-Cluster uses eight centroid vectors for document ranking task, $k = 48$ matches the size. In addition, we also compare two query-agnostic term selection methods applied on ColBERT proposed by Lassance et al. (2022), that use ColBERT's document token embeddings of k tokens selected based on leading positions or largest IDF, denoted by ColBERT-First and ColBERT-IDF, respectively. Lassance et al. (2022) used 50 tokens, *i.e.*, $k = 50$, for both the passage and the document ranking task, paying more costs than the other low-cost baselines and ours.

4.3 Training details

In this section, we explain training details for training PQA-Encoder. We adopt the same architecture and model configuration to those of the original implementation of ColBERT. For query/document length, we set 32 and 180, respectively, as in the original implementation. We first train ColBERT using the official train triples for MSMARCO for 300k steps, and then fine-tune PQA-ColBERT for 500k steps[5]. For constructing hard negative passages d^- used in fine-tuning, we adopt the same strategy used in D-Cluster. In D-Cluster, however, the hard negative passages are periodically updated for every T steps using the retriever being trained, consuming huge training time costs (e.g., 10 hours on a single GPU for each update). Instead, we update the hard negatives only once, after 100k steps. For inference, we follow the same indexing-retrieval pipeline to that of ColBERT, consisting of FAISS indexing (Johnson et al., 2019) and NN search. We explain more details in A.2.

4.4 Evaluation result

Passage Ranking Task In Table 3, we present the end-to-end retrieval performance of our PQA-ColBERT along with that of baselines, on passage ranking tasks.

Among ColBERT and ColBERT-PRF, which require the full-term indexing, ColBERT-PRF shows better performance overall. This indicates the benefits of adding relevant terms to the given query. However, as enriching the query representation at evaluation time, ColBERT-PRF significantly sacrifices the efficiency on the latency. We show pseudo-query terms from PQA-ColBERT, can also further enhance such high-cost retriever, when we can afford increases in index size and query latency, which we denote as PQA-ColBERT-H (ours) for high-latency scenario. More precisely, PQA-ColBERT-H (ours) uses top-k terms selected by our extractor in addition to the all document terms, to emphasize the contribution of the pseudo-query terms to the term matching, achieving the best MRR@10 on the MSMARCO Dev set and the best Recall@1k on TREC datasets.

Among low-cost approaches [6], ours is the only approach performing comparably to ColBERT, decreasing query latency and index size by a factor of 8 and 3, respectively.

[5]This step is omitted for RealQ scenario for scalability.

[6]The latency reported as "< 50 (ms)" indicates that latency cannot be directly compared due to different inference setup, but comparable to ours, and significantly lower than ColBERT.

	Retriever	MSMARCO Dev		TREC-DL 2019	Index (GB)	Latency (ms)
		MRR@10	@100	NDCG@10		
H	ColBERT	**40.1**	**41.0**	64.0	330	476
	BM25 → IDCM (3)	37.5	-	67.1	-	125
	BM25 → IDCM (4)	38.0	-	**68.8**	-	149
	PQA-ColBERT-H (ours)	**40.1**	**41.0**	65.8	369	550
L	ANCE (First-P)	-	37.2	61.5	5	< 50
	ME-BERT	-	33.2	-	19	< 50
	D-Cluster	-	39.2	62.8	39	< 50
	ColBERT-First	-	34.7	-	38	74
	ColBERT-IDF	-	22.5	-	38	74
	PQA-ColBERT (ours)	**39.4**	**40.3**	**64.1**	39	74

Table 4: End-to-end retrieval performance on document ranking datasets: MRR@10/100 for MSMARCO Dev Document Ranking dataset and NDCG@10 for TREC-DL 2019 Document Ranking dataset. **Bold-face** indicates the best performance for high- (H) and low-cost (L) approaches, respectively.

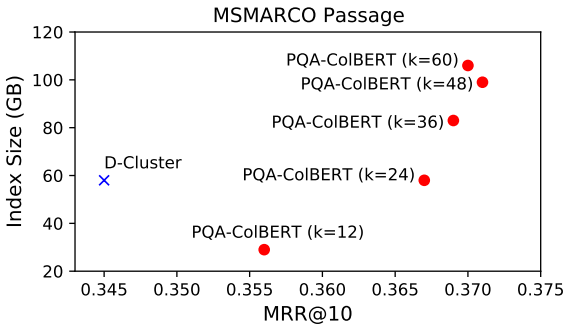

Figure 1: Ranking performance (MRR@10; x-axis) versus memory footprints (Index Size; y-axis) for D-Cluster and our PQA-ColBERT retrievers with different k values, on MSMARCO Passage Ranking dataset.

In Figure 1, we further compare results of our PQA-ColBERT with different k values, along with D-Cluster, while exploring the optimal value for k. Specifically, we compare memory footprints for indexing passages (y-axis) and ranking performance (x-axis) on MSMARCO Passage Ranking dataset. Compared to D-Cluster, our PQA-ColBERT with $k = 12$ already shows better MRR@10 with much smaller index size, showing the effectiveness of our pseudo-query extractor and pseudo-query-aware document representations. For PQA-ColBERT with different k values, while increasing k leads to better ranking performance, the gain decreased at higher k, and $k = 48$ shows the best performance. This indicates that only a few terms in a passage contribute to top-ranking results and our pseudo-query extractor accurately extracts those.

Document Ranking Task In Table 4, we compare retrievers on document ranking task, using MSMARCO Dev and TREC-DL 2019 datasets.

Retriever	MRR@10	Index size (GB)
ColBERT	69.6	722 (100.0%)
PQA-ColBERT (ours)	68.3	258 (-64.3%)

Table 5: End-to-end retrieval results on RealQ.

In addition to retriever baselines, we also report the ranking performance of the cross-encoder reranker IDCM, to compare query-aware document term selection. Note that, IDCM has an advantage of leveraging the given query for both term selection and encoding, while sacrificing efficiency. In contrast, we select pseudo-query terms before the real query is given, and separately encode documents by adopting bi-encoder approach, to devise an efficient, scalable retriever.

Among low-cost retrievers (presented in the "L" section at Table 4), our PQA-ColBERT outperforms all baselines, on both the two datasets. Compared to ColBERT (a high-cost retriever presented in the "H" section), while achieving comparable performance, ours decreases the index size and the query latency by a factor of 8.5 and 6.4, respectively. By paying additional cost to further increase the ranking effectiveness, our PQA-ColBERT-H outperforms ColBERT, as in the passage ranking tasks. Meanwhile, though IDCM reranker shows strong performance on TREC dataset, as re-encoding documents for each given query, IDCM still shows higher latency than ours. Furthermore, ours even outperforms IDCM on MSMARCO datasets.

In Table 5, we compare ColBERT with PQA-ColBERT on RealQ using more realistic queries. Since real queries in RealQ are much shorter than

Selection strategy	avg # of positive judgements
First-k	0.481
Top-k IDF	0.621
$\overline{q}^d$ (our prediction)	0.637
$\overline{q}^*$ (our supervision)	**0.701**

Table 6: Evaluation results of different query term selection strategies on FiRA. We present the average number of positive relevance judgments on each selected document term.

Retriever	NDCG@10	Index size (GB)
BM25	65.6	-
DPR	33.2	0.3
ANCE	65.4	0.3
ColBERT	67.7	5.8
PQA-ColBERT (ours)	64.3	1.5
+ prioritizing title	**68.2**	1.5
Abl. ColBERT-First	63.7	1.5

Table 7: End-to-end retrieval performance on the TREC-COVID dataset. NDCG@10 is used as the evaluation metric. **Bold-face** indicates the best performance, and red-color indicates the worse performance than that of BM25.

those in benchmark datasets, missing any of the query terms from document representation leads to significant loss of relevance score. We thus preserve more terms for indexing than the benchmark datasets, by setting $k = 65$. Ours captures query-relevant terms from d, showing comparable performance to that of ColBERT using all tokens.

4.5 Analysis on Pseudo-Query Extractor

In this section, we validate the effectiveness of our term-level supervision in diverse settings, using human and automatic evaluation. For baselines, we adopt two query-agnostic extractors, which select the first-k terms and top-k IDF terms. We present selected pseudo-query terms for an example document in Appendix A.4.

Human Evaluation For manual annotations, we employ FiRA dataset (Hofstätter et al., 2020), where multiple human annotators produce term-level relevance labels on relevant query-document pairs from TREC-DL 2019 Document Ranking dataset. To evaluate our supervision ($\overline{q}^*$ in §3.2), for each document term, we measure the average number of positive judgements for term relevance. For fair comparisons, we set different k for our predictions ($\overline{q}^d$ in §3.2) and baselines, by the number of gold query-relevant terms in $\overline{q}^*$ for each document. Our proposed supervision $\overline{q}^*$ outperforms the others, indicating the effectiveness of the supervision. The prediction from our pseudo-query extractor $\overline{q}^d$ is the second best, indicating our extractor indeed learns to extract relevant query terms from documents.

In the following paragraph, for more comprehensive analysis, instead of the limited amounts human annotated labels, we use $\overline{q}^*$ as gold query-relevant term, which can be obtained automatically, to scale up evaluation.

Automatic Evaluation To increase the reliability of the automatic evaluation, more precisely the reliability of the oracle $\overline{q}^*$, instead of FiRA, we use MSMARCO Dev Passage ranking dataset that provides much larger amounts of annotations for training ColBERT by which the oracle $\overline{q}^*$ is obtained. For 7433 labeled documents in MSMARCO Dev Passage ranking dataset with diverse characteristics, we compare distributions, on the position and IDF scores of query-relevant terms, from our selection ($\overline{q}^d$) with that of our oracle selection ($\overline{q}^*$). We set selection size k identical to that of oracle annotation for each document.

In Figure 2(a), we can observe the distribution of position from our extracted term closely resembles that of oracle distribution. The distribution is known to left-skewed, known as lead bias, which explains why First-k heuristic is often effective and generalizable (Hofstätter et al., 2020). However, this heuristic was less effective compared to our supervised selection, and we observe similar trends for IDF distribution (Figure 2(b)). Those observations explain the better ranking performance of our PQA-ColBERT compared to ColBERT-First and ColBERT-IDF (Table 3 and Table 4). For further qualitative and quantitative comparisons to the two selection methods, see Appendix A.5 and A.4, respectively.

4.6 Generalization to unseen q and d

In addition to the in-domain ranking performance, to compare the generalization ability of retrievers, we also evaluate the out-of-domain ranking performance using TREC-COVID dataset where standard dense retrievers such as DPR (Karpukhin et al., 2020) are known to not generalize well, underperforming BM25 (Thakur et al., 2021). Results are reported in Table 7. For PQA-ColBERT, we

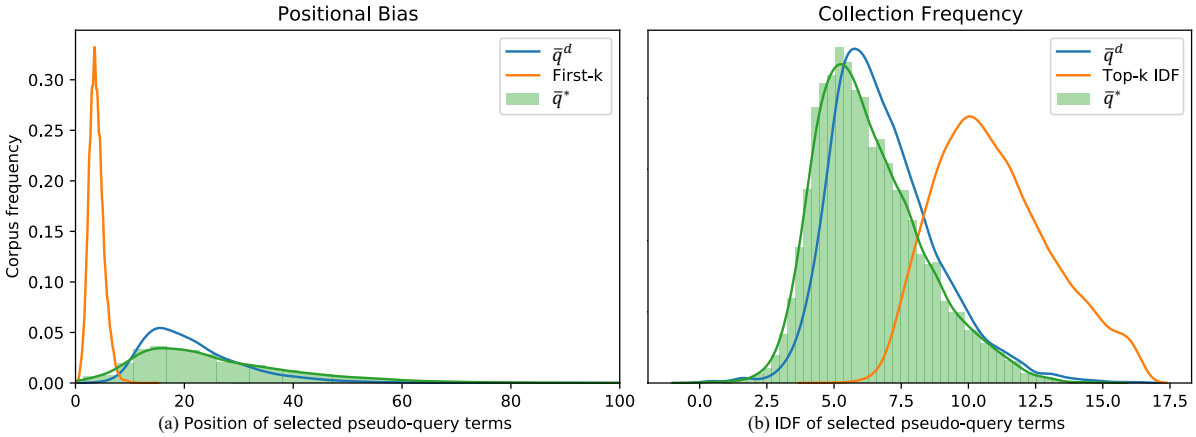

Figure 2: Histograms of (a) positions and (b) IDF scores for selected pseudo-query terms, on documents in MSMARCO Dev Passage Ranking dataset.

set $k = 36$ considering that the average length of documents in TREC-COVID is in-between that of MSMARCO Passage of using $k = 24$ and MS-MARCO Document of using $k = 48$.

Most of the dense retrievers including PQA-ColBERT underperform BM25 (presented with red-colored performance), showing the shared weakness on the generalization ability given the train-test difference. Meanwhile, titles in documents provide strong inductive bias in many datasets (Jin et al., 2002) including TREC-COVID, and our pseudo-query extractor is flexible enough to leverage such prior, in a way of giving the priority to title terms over other terms. Specifically, when we force our extractor to retain title terms and then extract top-k pseudo-query terms within the remaining budget (*i.e.*, k), denoted by "+ prioritizing title", our PQA-ColBERT shows the best performance, significantly outperforming BM25.

On the other hand, we stress that solely relying on titles is insufficient, given missing salient terms out of the titles. As an ablation study, we compare ColBERT-First[7] (denoted by *Abl* in Table 7) which preserves title terms as much as possible, since a title is placed at the beginning of a document, and uses the same index size to ours: Though maximally enjoying title terms, ColBERT-First still underperforms BM25 and even our PQA-ColBERT that do not leverage such prior, indicating that many salient terms exist also at the latter part of documents. In contrast, PQA-ColBERT successfully captures such terms, being synergetic to the prior on titles.

[7]The presented performance of ColBERT-First in this paper is different from that presented by Lassance et al. (2022) since different language models and different k values are used. In this ablation study, we use the same language model and k to PQA-ColBERT, to have a fair comparison between the two instead of competing on the state-of-the-art performance.

5 Conclusion

We studied enhancing document representation of bi-encoder, by balancing the benefit of the single-vector (of memory-efficient indexing) and the multi-vector (of preserving all relevant query terms) representation. Specifically, inspired by relevance model, we supervise a pseudo-query extractor by using query logs of relevant query-document pairs in the training dataset, to extract query-relevant terms at indexing time. Our proposed model is validated on MSMARCO and RealQ benchmarks, for both passage and document retrieval tasks. Our model achieves comparable ranking performance to multi-vector retriever and even to cross-encoder ranker, with significantly better memory-efficiency for indexing and lower latency for inference.

Acknowledgements

This work was supported by [search engine training optimization using deep learning] funded by NAVER corporation.

Limitations

Targeting real-world retrieval, we employ MS-MARCO and RealQ datasets which provide large-scale relevance annotations on diverse queries. However, both datasets have annotation biases inherent in most of the retrieval datasets.

Because of the extremely large number of documents in a corpus, relevance annotations are given only to a few documents, while the others are often assumed to be non-relevant. For example, the user clicks as a relevance proxy often suffer from the ranking bias (also known as position bias), where clicks are often only examined on the top-ranked (or top-positioned) documents by a system (Joachims et al., 2017). For MSMARCO, relevance annotations are biased to the documents that have exact matching terms to queries (Xiong et al., 2021). Such annotation biases have been a bottleneck for both training (Prakash et al., 2021; Kim et al., 2022) and evaluation (Arabzadeh et al., 2022).

To alleviate such biases, one can design the annotation process in an interactive way between the annotators and a target retriever, such that the pool of documents presented to the annotators dynamically change as the system evolves (Wang et al., 2021a), which we leave as future work.

References

Negar Arabzadeh, Alexandra Vtyurina, Xinyi Yan, and Charles LA Clarke. 2022. Shallow pooling for sparse labels. *Information Retrieval Journal*, pages 1–21.

Leif Azzopardi and Vishwa Vinay. 2008. Retrievability: An evaluation measure for higher order information access tasks. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 561–570.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951.

Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. 2021. Intra-document cascading: Learning to select passages for neural document ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1349–1358.

Sebastian Hofstätter, Markus Zlabinger, Mete Sertkan, Michael Schröder, and Allan Hanbury. 2020. Fine-grained relevance annotations for multi-task document ranking and question answering. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3031–3038.

Rong Jin, Alex G Hauptmann, and Cheng Xiang Zhai. 2002. Title language model for information retrieval. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–48.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2017. Accurately interpreting clickthrough data as implicit feedback. In *Acm Sigir Forum*, volume 51, pages 4–11. Acm New York, NY, USA.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. *ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT*, page 39–48. Association for Computing Machinery, New York, NY, USA.

Jihyuk Kim, Minsoo Kim, and Seung-won Hwang. 2022. Collective relevance labeling for passage retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4141–4147, Seattle, United States. Association for Computational Linguistics.

Carlos Lassance, Maroua Maachou, Joohee Park, and Stéphane Clinchant. 2022. Learned token pruning in contextualized late interaction over bert (colbert). In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2232–2236, New York, NY, USA. Association for Computing Machinery.

Victor Lavrenko and W Bruce Croft. 2017. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. ACM New York, NY, USA.

Ruohan Li, Jianxiang Li, Bhaskar Mitra, Fernando Diaz, and Asia J Biega. 2022. Exposing query identification for search transparency. In *Proceedings of the ACM Web Conference 2022*, pages 3662–3672.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

Jeremy Pickens, Matthew Cooper, and Gene Golovchinsky. 2010. Reverted indexing for feedback and expansion. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1049–1058.

Prafull Prakash, Julian Killingback, and Hamed Zamani. 2021. Learning robust dense retrieval models from incomplete relevance labels. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1728–1732.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Hongyin Tang, Xingwu Sun, Beihong Jin, Jingang Wang, Fuzheng Zhang, and Wei Wu. 2021. Improving document representations by generating pseudo query embeddings for dense retrieval. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5054–5064, Online. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. Trec-covid: Constructing a pandemic information retrieval test collection. 54(1).

Huazheng Wang, Yiling Jia, and Hongning Wang. 2021a. Interactive information retrieval with bandit feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2658–2661.

Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021b. Pseudo-relevance feedback for multiple representation dense retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '21, page 297–306, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

A Appendices

A.1 Dataset construction process for RealQ

To evaluate models on open-domain, realistic documents, we newly construct dataset, namely **RealQ**[8], from commercial search engines X [9]. The construction process is as follows[10].

We first sampled about 1M queries randomly from the recent 3 month Web search logs of X, satisfying the following constraints:

- A query should consist of more than 2 words.
- It has to appear at least 10 times in each month.
- It should not be a navigational query.

The first constraint was introduced to exclude a single-term query which does not have any contextual word of a query term. The remaining two

[8]The dataset was created for commercial purpose.

[9]Name annonymized for blind reviewing

[10]All documents are anonymized by the search engine.

constraints intend to rule out other factors than semantic relevance, such as recency or types of information, which affect in determining relevance between a query and a document as much as possible. A pre-built statistical search intent classifier, which is being deployed in X Web search, is used to identify a navigational query. Then, we extracted top 3 most clicked documents for each of sampled queries from the same log data and regards them as positive documents. To filter noisy clicks, we consider only a satisfactory click, determined by several heuristics considering dwell time and other user behavior pattern after clicks.

To sample negative documents, we first retrieve top 1,000 documents from X search engine, which indexes multi-billions of Web pages. To increase the relevance gap between positive, clicked documents and negative ones, we use simple BM25 as a ranking scheme instead of X's original ranking model utilizing a vast amount of different ranking signals. Then, 20 documents are randomly selected from the collected 1,000 documents and finally regarded as a negative samples for the query. In the case of test queries, we sampled 100 negative documents, instead of 20 documents, to enable to measure a recall@K metrics at a sufficiently large K. Note that it inevitably incurs a risk that many false-negative documents can be included in the dataset. However, we assume that such potentially false-negative documents would be less relevant than a positive document which received satisfactory clicks from users.

For the document preprocessing, we identified and extracted a title and body text using a HTML parser for each of samples, and regarded their concatenation as a text content of a document. In this step, a document which contains less than 20 words or a root page of a web site was filtered out in both positive and negative samples. As a result, we collect total 21,829,598 (query, document) pairs for 1,135,453 queries for training, and 10,935,449 pairs for 139,274 queries for testing. In the experiments, we used 20% of samples randomly chosen from both positive and negative pairs, to reduce training time required to learn a ColBERT model. The average number of positive documents for each query is 2.32. We present the detailed statistics on our dataset in Table 8.

	MS MARCO	**RealQ**
# of train set queries	532,761	1,145,579
# of evaluation set queries	6,980	169,213
# of documents for indexing	8,841,823	8,475,162
Average # of words in d	56.25	382.88
Average # of words in q	6.37	3.15

Table 8: Statistics on benchmark dataset MSMARCO and newly constructed dataset RealQ. q and d denote query and document respectively. The number of words in q and d is measured based on whitespaces. For RealQ, The queries and documents are sampled from real-world search engine for experiments.

		MS MARCO	RealQ
GPUs	Training	2 x RTX3090	4 x V100
	Indexing	8 x RTX3090	1 x V100
Elapsed time	Training	2 days	4 days
	Indexing	1 hour	9 hours
# parameters	ColBERT	109,580,544	18,263,424
	Pseudo-query extractor	131,457	131,457

Table 9: Computational budgets and the total number of parameters

A.2 Implementation details

For MSMARCO, we initialized the encoder with *bert-base-uncased* checkpoint from Transformers (Wolf et al., 2020) library. The maximum sequence length of document and query are 180 and 32, respectively. We trained our model for 500K steps using AdamW (Loshchilov and Hutter, 2019) optimizer with learning rate 3e-6 and batch size 36.

For RealQ, the encoder was initialized with pretrained ELECTRA (Clark et al., 2020) with 12 layers and 4 attention heads. The model was pretrained on web corpus in dominant language used in X. The maximum sequence length of document and query are set to 180 and 16, respectively. Then we train our model with a batch size 560 for 30 epochs.

A.3 Computational budgets

In Table 9, we present the computational budgets used in our experiments with the number of total parameters.

A.4 Qualitative Analysis on Pseudo-Query Extractor

In Figure 3, we present a relevant query-document pair from MSMARCO Dev Passage Ranking dataset, along with selected terms within the document by different strategies.

Query	Is ammonium salt soluble in water?
Passage	**ammonium** cation is found in a variety of **salts** such as ammonium carbonate, ammonium chloride, … most simple **ammonium salts are very soluble in water**. an exception is ammonium hexachloroplatinate, … the ammonium salts of nitrate and especially perchlorate are highly explosive, in these cases ammonium …
Selected terms	
First-k	**ammonium** cation is found in a variety of **salts** such as ammonium carbonate, ammonium chloride, … most simple **ammonium salts are very soluble in water**. an exception is ammonium hexachloroplatinate, … the ammonium salts of nitrate and especially perchlorate are highly explosive, in these cases ammonium …
Top-k IDF	**ammonium** cation is found in a variety of **salts** such as ammonium carbonate, ammonium chloride, … most simple **ammonium salts are very soluble in water**. an exception is ammonium hexachloroplatinate, … the ammonium salts of nitrate and especially perchlorate are highly explosive, in these cases ammonium …
Ours	**ammonium** cation is found in a variety of **salts** such as ammonium carbonate, ammonium chloride, … most simple **ammonium salts are very soluble in water**. an exception is ammonium hexachloroplatinate,… the ammonium salts of nitrate and especially perchlorate are highly explosive, in these cases ammonium …

Figure 3: A relevant query-document pair from MSMARCO Dev Passage Ranking dataset (the first two rows), followed by extracted top-24 pseudo-query terms (underlined) by different strategies (the last rows). The **bold-faced** terms are query-relevant terms and red-colored terms are missing relevant terms from the selection.

According to writing conventions, salient contents often appear at the beginning of the document (Hofstätter et al., 2020). By leveraging such tendency, First-k strategy successfully selects two query-relevant terms, "ammonium" and "salts". However, this obviously misses useful terms positioned beyond the leading terms, e.g., "soluble". Another useful feature for the extraction is collection frequency (IDF) which often indicates the importance of a term within a corpus. For example, Top-k IDF strategy selects discriminative, query-relevant terms such as "ammonium" and "hexachloroplainate". Note that, "hexachloroplainate" enables the document to be matched to other relevant yet unannotated queries, e.g., "Does ammonium hexachloroplainate soluble in water?". Nevertheless, Top-k IDF fails to capture "salt" and "soluble", as these may appear frequently in other documents, producing low IDF values. Different from the two strategies which use predefined features, our pseudo-query extractor learns, during training, useful features including the two aforementioned features, capturing all query-relevant terms such as "ammonium", "salt", and "soluble".

A.5 Quantitative Analysis on Pseudo-Query Extractor

By devising our pseudo-query extractor, our goal is to maximally preserve relevant query terms given a fixed budget for the document indexing (*i.e.*, k). In Figure 4, by using MSMARCO Dev Passage Ranking dataset, we evaluate how much the document-level relevance can be preserved using the selected pseudo-query terms, compared to using all terms

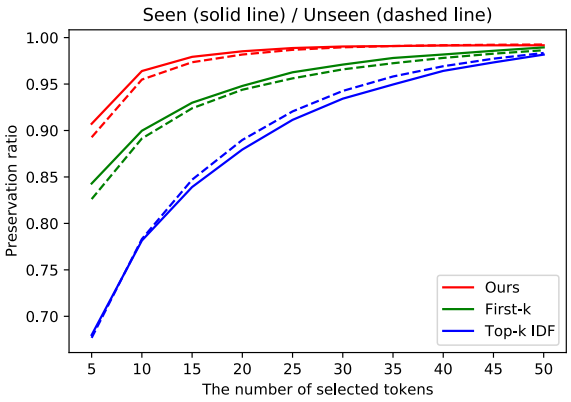

Figure 4: The number of selected tokens (x-axis) versus preservation ratio $\frac{\text{rel}(q,\bar{q}^d)}{\text{rel}(q,d)}$ (y-axis), on relevant q-d pairs in MSMARCO Dev Passage Ranking dataset. "Seen" ("Unseen") denote passages with (without) observed query logs in the training dataset.

for computing relevance. Specifically, we measure the degree of preservation on relevance scores from the pre-trained ColBERT using all tokens in each document (i.e., $\text{rel}(q, d)$ in Eq 4) to the scores using selected pseudo-query tokens (i.e., $\text{rel}(q, \bar{q}^d)$ in Eq 7), depending on the number of selected tokens. To analyze the generalizability of pseudo-query extractors, we present results on seen and unseen documents that have and do not have observed query logs in the training dataset, respectively. As baselines to our extractor, we compare with the two query-agnostic selection methods extracting leading k terms and top-IDF terms, denoted by First-k and Top-k IDF in Figure 4, respectively. Given the goal of achieving both memory-efficiency and

the ranking effectiveness, promising curves should bow towards the top-left corner.

Our pseudo-query extractor much quickly converges to the upper bound than the others. For example, our extractor preserves 95% of the relevance score rel(q, d) using less than 10 tokens (13% of entire document tokens on average), while baseline extractors use approximately 20 (ColBERT-First) and 30 (ColBERT-IDF) tokens to achieve the same preservation ratio. Furthermore, the gap between performance on seen and unseen documents is marginal, indicating that ours generalize well to unseen documents.