Salient Phrase Aware Dense Retrieval: Can a Dense Retriever Imitate a Sparse One?

Xilun Chen, Kushal Lakhotia[†], Barlas Oğuz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta and **Wen-tau Yih**
Meta AI
{xilun,barlaso,anchit,plewis,stanvp,mehdad,sonalgupta,scottyih}@meta.com
[†]lakhotia.kushal@gmail.com

Abstract

Despite their recent popularity and well-known advantages, dense retrievers still lag behind sparse methods such as BM25 in their ability to reliably match salient phrases and rare entities in the query and to generalize to out-of-domain data. It has been argued that this is an inherent limitation of dense models. We rebut this claim by introducing the Salient Phrase Aware Retriever (SPAR)[1], a dense retriever with the lexical matching capacity of a sparse model. We show that *a dense Lexical Model* $\mathbf{\Lambda}$ *can be trained to imitate a sparse one*, and SPAR is built by augmenting a standard dense retriever with $\mathbf{\Lambda}$. Empirically, SPAR shows superior performance on a range of tasks including five question answering datasets, MS MARCO passage retrieval, as well as the EntityQuestions and BEIR benchmarks for *out-of-domain* evaluation, exceeding the performance of state-of-the-art dense and sparse retrievers.

1 Introduction

Text retrieval is a crucial component for a wide range of knowledge-intensive NLP systems, such as open-domain question answering (ODQA) models and search engines. Recently, dense retrievers (Karpukhin et al., 2020; Xiong et al., 2021) have gained popularity and demonstrated strong performance on a number of retrieval tasks. Dense retrievers employ deep neural networks to learn continuous representations for the queries and documents, and perform retrieval in this dense embedding space using nearest neighbor search (Johnson et al., 2019). Compared to traditional sparse retrievers that rely on discrete bag-of-words representations, dense retrievers can derive more semantically expressive embeddings, thanks to its end-to-end learnability and powerful pre-trained encoders. This helps dense retrievers to overcome several inherent limitations of sparse systems such as *vocabulary mismatch* (where different words are used for the same meaning) and *semantic mismatch* (where the same word has multiple meanings).

On the other hand, while existing dense retrievers excel at capturing semantics, they sometimes fail to match the *salient phrases* in the query. For example, Karpukhin et al. (2020) show that DPR, unlike a sparse BM25 retriever (Robertson and Walker, 1994), is unable to catch the salient phrase "*Thoros of Myr*" in the query "*Who plays Thoros of Myr in Game of Thrones?*". In addition, dense retrievers struggle to *generalize to out-of-domain test data* compared to training-free sparse retrievers such as BM25. For instance, Sciavolino et al. (2021) find that DPR performs poorly compared to BM25 on simple entity-centric questions, and Thakur et al. (2021) introduce a new BEIR benchmark to evaluate the zero-shot generalization of retrieval models showing that BM25 outperforms dense retrievers on most tasks.

With dense and sparse retrievers each having their own distinctive pros and cons, researchers have long aspired to develop retriever models that combine the strengths of both. This, however, has proven challenging as dense and sparse retrievers are supported by drastically different algorithms and data structures (inverted index (Bialecki et al., 2012) for sparse and approximate nearest neighbor search (Johnson et al., 2019) for dense). Most existing research towards this goal extends sparse retrievers with improved representations from neural models (Lin and Ma, 2021). These methods, nonetheless, still rely on exact matching on a bag of tokens, which arguably cannot fully leverage the representational power of the pre-trained encoders.

The opposite route of *building better dense retrievers with the strengths of sparse models* is much less explored. In fact, there have been theoretical and empirical studies suggesting that such drawbacks of dense retrievers may be a result of in-

[1]The code and models of SPAR are available at: https://github.com/facebookresearch/dpr-scale/tree/main/spar.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 250–262

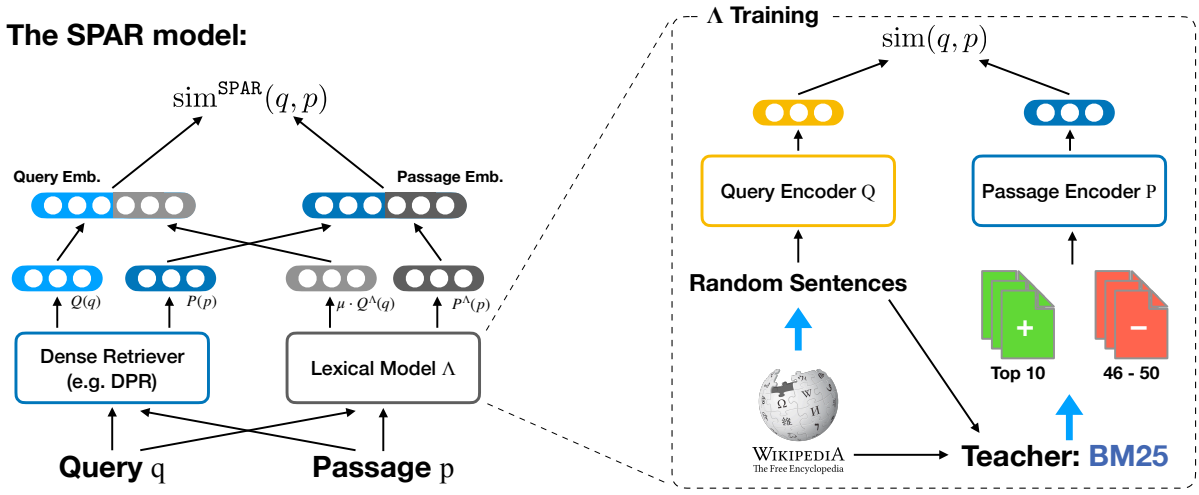

Figure 1: SPAR augments a dense retriever with a dense Lexical Model **Λ** trained to imitate a sparse teacher retriever. **Λ** is trained using random sentences as queries with positive and negative passages produced by the teacher. **Λ** is then combined with a dense retriever via vector concatenation to form a salient-phrase aware retriever.

herent limitations (Luan et al., 2021; Reimers and Gurevych, 2021). In this work, we embark on this underexplored research direction by proposing SPAR (Fig. 1), a *dense* retriever with the lexical matching capacity and out-of-domain generalization of a sparse model. In particular, we address an important and yet largely unanswered research question: **Can we train a dense retriever to imitate a sparse one?** Contrary to previous findings, we show that it is indeed possible to mimic a given sparse retriever (e.g., BM25 or UniCOIL (Lin and Ma, 2021)) with a dense Lexical Model **Λ**, and we build the SPAR model by combining **Λ** with a standard dense retriever (e.g., DPR or ANCE). Despite the long-standing dichotomy between sparse and dense retrievers, we arrive at a simple yet elegant solution of SPAR by conducting an extensive study to answer two key questions: i) How to train **Λ** to imitate a sparse retriever (§4.1) and ii) How to best utilize **Λ** to build a salient-phrase aware dense retriever (§4.2).

We evaluate SPAR on five ODQA datasets (§5.1) as well as on the MS MARCO (Bajaj et al., 2018) passage retrieval benchmark (§5.2), and show that it outperforms existing dense and sparse retrievers. We also examine the out-of-domain generalization of SPAR showing strong **zero-shot** performance across datasets (§5.3), including on the BEIR benchmark (Thakur et al., 2021) and a recently released dataset of entity-centric questions (Sciavolino et al., 2021). In addition, we conduct analyses of **Λ** showcasing its lexical matching capability (§6).

2 Related Work

Sparse retrievers date back for decades and successful implementations such as BM25 (Robertson and Walker, 1994) remain popular to date for its lexical matching capacity and great generalization. Despite the rapid rise of dense retrievers in recent years, development in sparse retrievers remain active, partly due to the limitations of dense retrievers discussed in §1. Various methods have been proposed to improve term weight learning (Dai and Callan, 2020; Mallia et al., 2021), address vocabulary mismatch (Nogueira and Lin, 2019) and semantic mismatch (Gao et al., 2021a), *inter alia*. While most of these methods have been incompatible with dense retrievers, our SPAR method provides a route for incorporating any such improvement into a dense retriever.

Dense retrievers employ pre-trained neural encoders to learn vector representations and perform retrieval by using nearest-neighbor search in this dense embedding space (Lee et al., 2019; Karpukhin et al., 2020). Subsequent works have developed various improvements, including more sophisticated training strategies and using better hard negatives (Xiong et al., 2021; Qu et al., 2021; Maillard et al., 2021; Oğuz et al., 2022). Such improvements are also complementary to the SPAR approach, which can potentially leverage these more powerful dense retrievers as shown in §5.2.

A few recent studies focus on the limitations of current dense retrievers. Lewis et al. (2021a) and Liu et al. (2022) study the generalization issue of dense retrievers in various aspects, such as

the overlap between training and test data, compositional generalization and the performance on matching novel entities. Thakur et al. (2021) introduce a new BEIR benchmark to evaluate the zero-shot generalization of retrieval models showing that BM25 outperforms dense retrievers on most tasks. A different line of research explores using multiple dense vectors as representations which achieves higher accuracy but is much slower (Khattab and Zaharia, 2020). Lin et al. (2021b) further propose a knowledge distillation method to train a standard dense retriever with similar performance of the multi-vector ColBERT model. More recently, Sciavolino et al. (2021) create EntityQuestions, a synthetic dataset of entity-centric questions to highlight the failure of dense retrievers in matching key entities in the query. We evaluate the generalization of SPAR on BEIR and EntityQuestions in §5.3.

Hybrid retrievers directly combine sparse and dense retrievers, and have been the most commonly used approach to overcome the limitations of a dense or sparse retriever (Gao et al., 2021b; Ma et al., 2022) before this work. A hybrid system retrieves two separate sets of candidates using a dense and a sparse retriever and rerank them using the hybrid retrieval score. Compared to a hybrid retriever, SPAR offers the same performance with a much simpler architecture. We compare SPAR with the hybrid models in more details in §4.3.

3 Preliminaries: Dense Retrieval

In this work, we adopt DPR (Karpukhin et al., 2020) as our dense retriever architecture for learning the Lexical Model $\mathbf{\Lambda}$. We give a brief overview of DPR in this section and refer the readers to the original paper for more details.

DPR is a bi-encoder model with a *query encoder* and a *passage encoder*, each a BERT transformer (Devlin et al., 2019), which encodes the queries and passages into d-dimensional vectors, respectively. Passage vectors are generated offline and stored in an index built for vector similarity search using libraries such as FAISS (Johnson et al., 2019). The query embedding is computed at run time, which is used to look up the index for k passages whose vectors are the closest to the query representation using dot-product similarity.

DPR is trained using a contrastive objective: given a query and a positive (relevant) passage, the model is trained to increase the similarity between the query and the positive passage while decreasing the similarity between the query and negative ones. It is hence important to have *hard negatives* (irrelevant passages that are likely confused with positive ones) for more effective training[2].

We employ the DPR implementation from Oğuz et al. (2022), which supports efficient multi-node training as well as memory-mapped data loader, both important for the large-scale training of $\mathbf{\Lambda}$. For model training, we also adopt their validation metrics of mean reciprocal rank (MRR) on a surrogate corpus using one positive and one hard negative from each query in the development set. Assuming a set of N dev queries, this creates a mini-index of $2N$ passages, where the MRR correlates well with full evaluation while being much faster.

4 The SPAR Model

In this section, we present SPAR, our salient phrase aware dense retriever. As illustrated in Figure 1, the basic idea of SPAR is to first train a dense Lexical Model $\mathbf{\Lambda}$ such that it produces similar predictions to a sparse teacher retriever. $\mathbf{\Lambda}$ is then combined with a regular dense retriever via vector concatenation. Although the high-level idea of SPAR is fairly straightforward, details of the model training process dictate its success in practice, and thus require a careful experimental study. To find the best configuration of SPAR, we conduct pilot experiments using the validation set of NaturalQuestions (NQ, Kwiatkowski et al., 2019) following the ODQA setting (Lee et al., 2019). BM25 is adopted as the teacher model for training $\mathbf{\Lambda}$. Below we describe the key results on how to successfully train $\mathbf{\Lambda}$ (§4.1) and on how to best leverage $\mathbf{\Lambda}$ in SPAR (§4.2).

4.1 Training the Lexical Model $\mathbf{\Lambda}$

There are many options to train $\mathbf{\Lambda}$ to imitate the predictions of a sparse retriever, such as mimicking the scores of the sparse retriever with the MSE loss or KL divergence, or learning the passage ranking of the teacher while discarding the scores. After unsuccessful initial attempts with these methods, we instead choose a very simple approach inspired by the DPR training. Recall that to train a DPR model, a *passage corpus* and a set of *training queries* are needed, where each query is associated with one or more *positive* and *hard negative* passages. To create such data for training $\mathbf{\Lambda}$, we use the sparse teacher retriever to produce the positive and negative passages. In particular, for any given query,

[2]DPR uses BM25 to generate hard negatives.

we run the teacher model to retrieve the top K passages and use the top n_p passages as positives and the bottom n_n as negatives. After the training data is generated, $\mathbf{\Lambda}$ can be trained using the same contrastive loss as DPR. To find the best strategy for training $\mathbf{\Lambda}$, we experiment with different training queries and different values of K, n_p and n_n.

We use a similar process to create the *validation* data, and adopt the MRR metric (§3) for evaluating whether $\mathbf{\Lambda}$ behave similarly to the teacher model. In particular, we use the questions from the NQ validation set as the target queries and whether a passage is relevant is again determined by the teacher. For validation, n_p is set to 1, which means only the passage ranked the highest by the teacher is considered positive. As a result, a higher MRR on this validation data indicates that the predictions of $\mathbf{\Lambda}$ are more similar to the teacher.

Training queries As the teacher model can be run on arbitrary queries to generate positive and negative passages, we are not restricted to any annotated data for training $\mathbf{\Lambda}$. We experiment with different approaches for constructing such queries, focusing on two potentially helpful properties: i) how similar the queries are to the downstream evaluation queries and ii) the quantity of queries as large-scale training data gives the neural models more exposure to rare words and phrases. Specifically, we consider three query sets: NQ questions, Wiki sentences and PAQ questions. NQ questions consists of 59k questions in the training set of NQ, which have the benefit of being similar to the evaluation queries. Wiki sentences are a large collection of 37 million sentences sampled randomly from the Wikipedia passage corpus. (See Appendix A for how the queries are sampled.) Finally, PAQ questions are from the PAQ dataset (Lewis et al., 2021b), a collection of 65 million synthetically generated *probably asked questions* based on Wikipedia, which have both desirable properties.

Based on our experiments, NQ $\mathbf{\Lambda}$ achieves a MRR of 76.5%, while Wiki $\mathbf{\Lambda}$ attains 92.4% and PAQ $\mathbf{\Lambda}$ reaches 94.7%. This indicates that the size of the data certainly plays an important role, as both Wiki $\mathbf{\Lambda}$ and PAQ $\mathbf{\Lambda}$ outperform NQ $\mathbf{\Lambda}$ by a wide margin. PAQ $\mathbf{\Lambda}$ achieves the highest MRR among the three options, for it combines the benefits of NQ and Wiki $\mathbf{\Lambda}$, with large-scale queries that are also similar to the downstream evaluation data. However, such large-scale synthetic queries are very expensive to obtain and not available for all domains, so we focus our attention on the much cheaper Wiki $\mathbf{\Lambda}$ option, given that the gap between their performance is small.

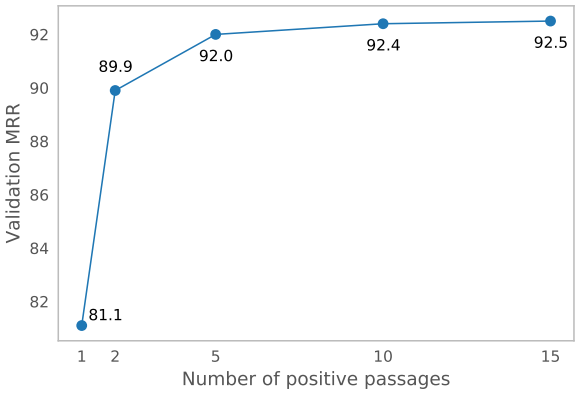

Figure 2: Validation MRR of the Wiki $\mathbf{\Lambda}$ using various numbers of positive passages.

Number of positive and negative passages We also experimented with the numbers of positive and negative passages per training query. The model performance was not sensitive to the total number of passages retrieved from the teacher model (K) as well as the number of negative passages (n_n). However, the number of positives (n_p) is more important. As shown in Figure 2, using more than one positive passage is critical to successful training, as $n_p = 2$ significantly improves the validation metrics over $n_p = 1$. Further increase in n_p remains helpful, but with a diminished return. As a result, we use $n_p = 10$ in our final model.

4.2 Building SPAR with $\mathbf{\Lambda}$

With a successfully trained dense lexical model $\mathbf{\Lambda}$, we next experiment with various approaches to build a salient-phrase aware retriever with it. All models are trained on NQ training set and we report the Acc@20 and Acc@100 metrics on the test set, which evaluates whether any of the top 20/100 retrieved passages contain the answer to the input question. We consider two baseline models, DPR (NQ-single) and the hybrid DPR+BM25 model[3]. Because SPAR is built by augmenting DPR with $\mathbf{\Lambda}$, it should perform better than the DPR alone. Moreover, if our dense lexical model is effective, then the final model should achieve a comparable performance as DPR+BM25.

The first method we test is to initialize DPR with the model weights of $\mathbf{\Lambda}$ (Initialization) for DPR

[3]We implement DPR+BM25 with Pyserini (Lin et al., 2021a); see §4.3 for more discussions on hybrid models.

training, with the hope that DPR can inherit the lexical matching capacity from $\mathbf{\Lambda}$. However, as shown in Table 1, the results are only slightly better than the original DPR on Acc@100. We next test directly combining the vectors of DPR and $\mathbf{\Lambda}$, using either summation (Weighted Sum) or concatenation (Weighted Concat), where the weights are tuned using the validation data. In our experiments, we find that both methods perform well, achieving higher Acc@20 and Acc@100 scores than DPR+BM25. Although concatenation performs better, summation has the benefit of not increasing the dimension of the final vectors. Since both DPR and $\mathbf{\Lambda}$ are dense retrievers, this *post-hoc* combination can be easily done while still using a single FAISS index.

The good performance of Weighted Concat is encouraging, but also triggers two questions. First, does the good performance come from the longer embedding size? To answer this question, we include the ensemble (weighted concatenation of embeddings) of two independently trained DPR models (2 DPRs) as an additional baseline. Although it has the same dimension and number of parameters as Weighted Concat, its performance is substantially lower. Second, can we train a better concatenation model instead of simply combining the vectors of two separately trained models at test time? We experiment with a joint training approach, in which we concatenate the DPR embeddings with that of a trained $\mathbf{\Lambda}$ during DPR training. The similarity and loss are computed with the concatenated vector, but we freeze $\mathbf{\Lambda}$, and only train the DPR encoders as well as the scalar weight for vector concatenation. The idea is to make DPR "aware" of the lexical matching scores given by $\mathbf{\Lambda}$ during its training in order to learn a SPAR model. This can also be viewed as training DPR to correct the errors made by $\mathbf{\Lambda}$. Somewhat surprisingly, however, this strategy does not work well compared to post-hoc concatenation as shown in Table 1. We hence adopt the Weighted Concat method in our final model.

Concatenation Weight Tuning When concatenating the vectors from two dense retrievers, they may be on different scales, especially across varied datasets. It is hence helpful to add a weight μ to balance the two models during concatenation. We *add the weight to the query embeddings* so that the offline passage index is not affected by a change of weight. Specifically, for a query q and a passage p, a dense retriever with query encoder Q and passage encoder P, as well as a $\mathbf{\Lambda}$ model with Q^Λ and P^Λ,

Acc@k on NQ	@20	@100
DPR	78.3	85.6
2 DPRs (Weighted Concat)	79.8	86.4
DPR + BM25 Hybrid	80.9	87.9
SPAR (Initialization)	78.1	86.3
SPAR (Joint Training)	79.2	86.6
SPAR (Weighted Sum)	81.3	88.0
SPAR (Weighted Concat)	**82.2**	**88.3**

Table 1: Comparison of various methods of leveraging the Wiki $\mathbf{\Lambda}$ in SPAR: used as initialization for DPR training; combining two trained models with weighted vector sum or concatenation; vector concatenation during DPR training (joint training).

the final query vector in SPAR is $\left[Q(q), \mu Q^\Lambda(q)\right]$ while the passage vector being $\left[P(p), P^\Lambda(p)\right]$. The final similarity score is equivalent to a linear combination of the two model scores:

$$\begin{aligned}\text{sim}^{\text{SPAR}}(q,p) &= \left[Q(q), \mu Q^\Lambda(q)\right]^\intercal \left[P(p), P^\Lambda(p)\right] \\ &= \text{sim}(q,p) + \mu \cdot \text{sim}^\Lambda(q,p) \qquad (1)\end{aligned}$$

Note that unlike hybrid retrievers, the similarity function of SPAR is an **exact** hybrid of $\mathbf{\Lambda}$ and the base retriever, achievable in a single FAISS index search, thanks to the fact that both are dense retrievers. In addition, our decision of adding μ to the query vectors can potentially support dynamic or query-specific weights without the need to change the index, which we leave for future work.

Our final SPAR model is a general framework for augmenting any dense retriever with the lexical matching capability from any given sparse retriever. We first train $\mathbf{\Lambda}$ using queries from random sentences in the passage collection and labels generated by the teacher model with 10 positive and 5 hard negative passages. We then combine $\mathbf{\Lambda}$ and the base dense retriever with weighted vector concatenation using weights tuned on the development set. The passage embeddings can still be generated offline and stored in a single FAISS index and retrieval can be done in the same way as a standard dense retriever. Further implementation details can be found in Appendix A.

4.3 Comparing SPAR with Hybrid Retrievers

While most research focuses on improving either dense or sparse retrievers to overcome their drawbacks, people in practice also build hybrid retrievers that directly incorporate both. In addition to addressing an open research question probing the limits of a dense retriever, SPAR also has several *practical* advantages over such hybrid models, which is discussed in this section.

Model	NQ		SQuAD		TriviaQA		WebQ		TREC		**Average**	
	@20	@100	@20	@100	@20	@100	@20	@100	@20	@100	@20	@100
(**s**) BM25	62.9	78.3	71.1	81.8	76.4	83.2	62.4	75.5	80.7	89.9	70.7	81.7
(**d**) Wiki **Λ**	62.0	77.4	67.6	79.4	75.7	83.3	60.4	75.0	79.8	90.5	69.1	81.1
(**d**) PAQ **Λ**	63.8	78.6	68.0	80.1	76.5	83.4	63.0	76.4	81.0	90.5	70.5	81.8
(**d**) DPR-multi	79.5	86.1	52.0	67.7	78.9	84.8	75.0	83.0	88.8	93.4	74.8	83.0
(**d**) xMoCo[1]	82.5	86.3	55.9	70.1	80.1	85.7	**78.2**	84.8	89.4	94.1	77.2	84.2
(**d**) ANCE[2]	82.1	87.9	-	-	80.3	85.2	-	-	-	-	-	-
(**d**) RocketQA[3]	82.7	88.5	-	-	-	-	-	-	-	-	-	-
(**h**) DPR + BM25[4]	82.6	88.6	**75.1**	**84.4**	**82.6**	86.5	77.3	84.7	90.1	95.0	**81.5**	87.8
(**d**) SPAR-Wiki	**83.0**	**88.8**	73.0	83.6	**82.6**	86.7	76.0	84.4	89.9	95.2	80.9	87.7
(**d**) SPAR-PAQ	82.7	88.6	72.9	83.7	82.5	**86.9**	76.3	**85.2**	**90.3**	**95.4**	80.9	**88.0**
Cross-dataset model generalization (Discussed in §5.3)												
(**d**) SPAR-MARCO	82.3	88.5	71.6	82.6	82.0	86.6	77.2	84.8	89.5	94.7	80.5	87.4

[1] (Yang et al., 2021) [2] (Xiong et al., 2021) [3] (Qu et al., 2021) [4] (Ma et al., 2022)

Table 2: Acc@20 and 100 for Open-Domain Question Answering. Model types are shown in parentheses (**d**: dense, **s**: sparse, **h**: hybrid). The highest performance is in bold, and the highest among dense retrievers is underlined.

Architectural Complexity A major advantage of SPAR is its simplicity as a dense retriever. Hybrid retrievers need to build and search from two separate indices with different libraries (e.g. FAISS for dense and Lucene for sparse), and the retrieved passages from the dense and the sparse index then need to be aggregated to form the final results. SPAR, on the other hand, can be deployed and used in the same way as any standard dense retriever such as DPR without added complexity. Passage embeddings can be pre-computed and stored in a single FAISS index, and only a single lookup in the FAISS index is needed for retrieval. Furthermore, most dense retrieval optimizations such as Product Quantization, HNSW can also be applied to SPAR. Last but not least, it is prohibitive to perform an exact hybrid search in hybrid models and challenging to even devise an efficient approximation (Wu et al., 2019). SPAR retrieval, however, is equivalent to an **exact** hybrid of **Λ** and the base retriever (Eqn. 1), without the need for approximation.

Retrieval Speed and Index Size SPAR has two variants (§4.2), where the *weighted concat* variant is optimized for accuracy while the *weighted sum* variant has higher efficiency. With the weighted sum variant, the index size and retrieval speed stays the same as the base dense retriever, making it superior than a hybrid model.

For the weighted concat variant, SPAR index search takes 20ms/query using a HNSW index (compared to DPR's 10ms). BM25, in comparison, has a latency of 55ms using the Anserini toolkit (Hofstätter et al., 2021), and a hybrid DPR+BM25 retriever has a latency of 58ms assuming DPR and BM25 retrieval can be done in parallel. For the index size, however, a hybrid model may have an advantage thanks to the small index footprint of BM25. SPAR's index for MS MARCO is 52GB when using weighted concat, which is twice as large as DPR. A hybrid DPR+BM25 model, instead, has an index size of 27GB as BM25 only takes up 700MB space.

5 Experiments

5.1 Open-Domain Question Answering

Datasets We evaluate on five widely used ODQA datasets (Lee et al., 2019): NaturalQuestions (NQ, Kwiatkowski et al., 2019), SQuAD v1.1 (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), WebQuestions (WebQ, Berant et al., 2013) and CuratedTREC (TREC, Baudiš and Šedivý, 2015). We follow the exact setup of DPR (Karpukhin et al., 2020), including the train, dev and test splits, and the Wikipedia passage collection, as well as the accuracy@k (Acc@k) metric for evaluation, which is defined as the fraction of queries that has at least one positive passage retrieved in the top k.

Table 2 presents the main results on ODQA. For SPAR models, we report two variants both trained with BM25 as teacher, using the Wiki and PAQ training queries respectively (§4.1). The MARCO **Λ** is to test the model generalization of SPAR, and will be discussed in §5.3. SPAR outperforms all state-of-the-art retrievers in the literature, usually by wide margins, demonstrating the effectiveness of our approach.

Model	MS MARCO Dev Set		
	MRR@10	R@50	R@1000
(**s**) BM25	18.7	59.2	85.7
(**s**) UniCOIL (Lin and Ma, 2021)	35.2	80.7	95.8
(**d**) MARCO BM25 $\mathbf{\Lambda}$	17.3	56.3	83.1
(**d**) MARCO UniCOIL $\mathbf{\Lambda}$	34.1	82.1	97.0
(**d**) ANCE (Xiong et al., 2021)	33.0	79.1	95.9
(**d**) TCT-ColBERT (Lin et al., 2021b)	35.9	-	97.0
(**d**) RocketQA (Qu et al., 2021)	37.0	84.7	97.7
(**h**) ANCE + BM25	34.7	81.6	96.9
(**h**) RocketQA + BM25	38.1	85.9	98.0
(**h**) ANCE + UniCOIL	37.5	84.8	97.6
(**h**) RocketQA + UniCOIL	**38.8**	**86.5**	97.3
(**d**) SPAR (ANCE + $\mathbf{\Lambda}$=MARCO BM25)	34.4	81.5	97.1
(**d**) SPAR (RocketQA + $\mathbf{\Lambda}$=MARCO BM25)	37.9	85.7	98.0
(**d**) SPAR (ANCE + $\mathbf{\Lambda}$=MARCO UniCOIL)	36.9	84.6	98.1
(**d**) SPAR (RocketQA + $\mathbf{\Lambda}$=MARCO UniCOIL)	38.6	86.3	**98.5**
Cross-dataset model generalization (Discussed in §5.3)			
(**d**) Wiki BM25 $\mathbf{\Lambda}$	15.8	50.8	78.8
(**d**) SPAR (ANCE + $\mathbf{\Lambda}$=Wiki BM25)	34.4	81.5	97.0
(**d**) SPAR (RocketQA + $\mathbf{\Lambda}$=Wiki BM25)	37.7	85.3	98.0

Table 3: SPAR results on MS MARCO passage retrieval. We consider several options for $\mathbf{\Lambda}$, trained with different objectives (BM25 and UniCOIL) and different corpora (MSMARCO and Wikipedia). For ANCE and RocketQA, we use the released checkpoints and our evaluation scripts. We matched public numbers in most cases, but we were unable to reproduce the R@50 and R@1000 reported by RocketQA.

Another appealing result comes from SQuAD, a dataset on which all previous dense retrievers fail to even get close to a simple BM25 model. As the SQuAD annotators are given the Wikipedia passage when they write the questions, the lexical overlap between the questions and the passages is hence higher than other datasets. The poor performance of dense retrievers on SQuAD confirms that dense retrieval struggles at lexical matching. On the other hand, SPAR dramatically improves over previous models, achieving an improvement of 13.6 points in Acc@100 over the best existing dense retriever.

PAQ $\mathbf{\Lambda}$ matches the accuracy of the teacher BM25 model, while Wiki $\mathbf{\Lambda}$ performs slightly worse. The performance gap, however, is smaller in the final SPAR model. Both approaches are able to match the performance of the hybrid model, and SPAR-PAQ is only 0.3% better on average than SPAR-Wiki. This enables us to go with the much cheaper Wiki option for training $\mathbf{\Lambda}$ without sacrificing much of the end performance.

5.2 MS Marco Passage Retrieval

In this section, we report our experiments on the MS MARCO passage retrieval dataset (Bajaj et al., 2018), a popular IR benchmark with queries from the Bing search engine and passages from the web. Following standard practice, we evaluate MRR@10, Recall@50 and Recall@1000.

To highlight the versatility of our approach, we adopt two base dense retrievers in SPAR, ANCE and RocketQA. We further consider two sparse retrievers for training $\mathbf{\Lambda}$, BM25 and UniCOIL (Lin and Ma, 2021), a recent SoTA sparse retriever, to study whether SPAR training can imitate a more advanced teacher model. Similar to the Wiki training queries, we create a MARCO corpus for training $\mathbf{\Lambda}$. As the MS MARCO passage collection has fewer passages than Wikipedia, we use all sentences instead of sampling, resulting in a total of 28 million queries. We also report the performance of the Wiki $\mathbf{\Lambda}$ for model generalization (see §5.3).

Table 3 illustrates the results, where the sections correspond to sparse retrievers, the Lexical Models, state-of-the-art dense retrievers, various hybrid models, and finally SPAR and the model generalization experiments. As demonstrated in Table 3, SPAR is able to augment ANCE and RocketQA with the lexical matching capacity from either BM25 or UniCOIL, leading to a performance close to the hybrid retriever, and again outperforming all existing dense and sparse retrievers with a MRR@10 of 38.6. The fact that SPAR works with not only DPR and BM25, but other SoTA dense

	TC	NF	NQ	HQ	FQ	AA	T2	Qu	CQ	DB	SD	Fe	CF	SF	**Avg.**
BM25	65.6	32.5	32.9	60.3	23.6	31.5	**36.7**	78.9	29.9	31.3	15.8	75.3	21.3	66.5	43.0
docT5query	71.3	32.8	39.9	58.0	29.1	34.9	34.7	80.2	32.5	33.1	16.2	71.4	20.1	67.5	44.4
ANCE	65.4	23.7	44.6	45.6	29.5	41.5	24.0	85.2	29.6	28.1	12.2	66.9	19.8	50.7	40.5
ColBERT	67.7	30.5	52.4	59.3	31.7	23.3	20.2	85.4	35.0	39.2	14.5	77.1	18.4	67.1	44.4
Contriever (Izacard et al., 2022)	59.6	32.8	49.8	63.8	32.9	44.6	23.0	86.5	34.5	41.3	16.5	75.8	23.7	67.7	46.6
GTR-large (Ni et al., 2021)	55.7	32.9	54.7	57.9	**42.4**	52.5	21.9	89.0	38.4	39.1	15.8	71.2	**26.2**	63.9	47.2
GTR-large (our reproduction)	56.3	31.4	55.1	57.8	41.1	50.9	22.0	88.5	36.2	39.5	15.5	56.6	19.8	54.0	44.6
SPAR (ANCE + BM25 $\mathbf{\Lambda}$)	68.4	27.7	47.3	53.6	32.1	45.0	28.1	86.7	33.2	32.1	14.1	72.6	23.2	59.6	44.5
SPAR (ANCE + UniCOIL $\mathbf{\Lambda}$)	**76.4**	31.8	51.1	63.0	32.3	47.2	30.3	86.9	35.8	36.1	15.5	80.2	23.5	62.6	48.1
SPAR (GTR + BM25 $\mathbf{\Lambda}$)	60.7	32.3	55.7	60.9	40.9	51.4	23.5	89.3	37.4	41.1	16.3	58.0	20.5	57.3	46.1
SPAR (GTR + UniCOIL $\mathbf{\Lambda}$)	73.7	**34.0**	**57.0**	66.2	39.8	**52.6**	30.3	**89.6**	**39.1**	41.9	**17.2**	66.5	21.4	62.8	49.4
SPAR (Contriever + BM25 $\mathbf{\Lambda}$)	63.0	33.7	51.3	66.4	34.1	45.9	24.9	87.5	35.8	**42.8**	16.9	76.9	24.8	**69.5**	48.1
SPAR (Contriever + UniCOIL $\mathbf{\Lambda}$)	73.5	33.8	53.1	**67.6**	33.7	48.8	27.5	87.0	36.5	42.2	17.1	**81.2**	24.5	68.3	**49.6**

Table 4: Zero-shot results on BEIR (Thakur et al., 2021). All SPAR models, including the concatenation weights, are trained / tuned on MS MARCO. Dataset Legend: TC=TREC-COVID, NF=NFCorpus, NQ=NaturalQuestions, HQ=HotpotQA, FQ=FiQA, AA=ArguAna, T2=Touché-2020, Qu=Quora, CQ=CQADupStack, DB=DBPedia, SD=SCIDOCS, Fe=FEVER, CF=Climate-FEVER, SF=SciFact.

and sparse retrievers makes SPAR a general solution for combining the knowledge of dense and sparse retrievers in a single dense model.

One interesting phenomenon in both experiments is that while $\mathbf{\Lambda}$ by itself achieves a slightly lower performance than the teacher sparse retriever, the final SPAR model can reach or beat the hybrid model when combined with the same dense retriever. One possible reason why SPAR outperforms the hybrid model may be that SPAR is able to perform an *exact* "hybrid" of two retrievers since both are dense models (Eqn. 1), while the hybrid model relies on approximation. We leave further investigation in this curious finding to future work.

5.3 Out-of-Domain Generalization of SPAR

We now focus on another important topic regarding the generalization of SPAR. We have shown that Wiki $\mathbf{\Lambda}$ achieves a similar performance to PAQ $\mathbf{\Lambda}$, making it often unnecessary to rely on sophisticatedly generated queries for training $\mathbf{\Lambda}$. A more exciting finding is that $\mathbf{\Lambda}$ also has great **zero-shot** generalization to other datasets.

In the last section of Table 2 and 3, we reported SPAR performance on ODQA using the $\mathbf{\Lambda}$ model built for MS MARCO and vice versa. In both directions, $\mathbf{\Lambda}$ has a high zero-shot accuracy, and SPAR's performance is close to that using in-domain $\mathbf{\Lambda}$. This suggests that $\mathbf{\Lambda}$ shares the advantage of better generalization of a sparse retriever, and it may not be always necessary to retrain $\mathbf{\Lambda}$ on new datasets.

5.3.1 Zero-shot performance on BEIR

We further evaluate zero-shot transfer of SPAR on the BEIR benchmark (Thakur et al., 2021), which consists of a diverse set of 18 retrieval tasks gacross 9 domains[4]. In particular, following the standard setup, all models are trained on MS MARCO and tested on the BEIR benchmarks. Therefore, we adopt the MARCO $\mathbf{\Lambda}$ models, and combine them with various dense retrievers trained on MS MARCO to form SPAR models. As shown in Table 4, SPAR achieves a new state-of-the-art overall performance, and performs the best on 11 out of 14 datasets. Regardless of the choice of the base dense retriever, adding either the BM25 or UniCOIL $\mathbf{\Lambda}$ can consistently and substantially boost the performance of the base retriever, even for very recent SoTA models such as Contriever (Izacard et al., 2022) and GTR (Ni et al., 2021).

5.3.2 SPAR on EntityQuestions

Acc@k on EQ	Acc@20	Acc@100
(**d**) DPR	56.6	70.1
(**s**) BM25	70.8	79.2
(**h**) DPR + BM25	73.3	**82.3**
(**d**) Wiki $\mathbf{\Lambda}$	68.4	77.5
(**d**) PAQ $\mathbf{\Lambda}$	69.4	78.6
(**d**) SPAR	73.6	81.5
(**d**) SPAR-PAQ	**74.0**	82.0

Table 5: Zero-shot performance on the EntityQuestions (Sciavolino et al., 2021) dataset. We report micro-average instead of macro-average in the original paper.

A concurrent work (Sciavolino et al., 2021)

[4] 4 tasks were omitted in our evaluation for license reasons.

also identifies the lexical matching issue of dense retrievers, focusing specifically on entity-centric queries. They create a synthetic dataset containing simple entity-rich questions, where DPR performs significantly worse than BM25. In Table 5, we evaluate SPAR on this dataset in a zero-shot setting without any re-training, other than tuning the concatenation weight on the development set. The result further confirms the generalization of SPAR. $\mathbf{\Lambda}$ transfers much better to this dataset than DPR, achieving a slightly lower performance than BM25. When $\mathbf{\Lambda}$ is combined with DPR, SPAR achieves a higher Acc@20 than the hybrid model, and overall an improvement of 17.4 points over DPR.

6 Does Λ Learn Lexical Matching?

In this section, we verify whether $\mathbf{\Lambda}$ actually learns lexical matching with a series of analyses.

6.1 Rank Biased Overlap with BM25

RBO w/ BM25	NQ	SQuAD	Trivia
DPR	.104	.078	.170
Wiki $\mathbf{\Lambda}$	.508	.452	.505
PAQ $\mathbf{\Lambda}$	.603	.478	.527

Table 6: Rank Biased Overlap (RBO, Webber et al., 2010) between BM25 and various dense retrievers on the dev set. We use the standard $p = 0.9$ in RBO.

We first directly compare the predictions of $\mathbf{\Lambda}$ against BM25. As shown in Table 6, the prediction of DPR and BM25 are dramatically different from each other, with a RBO of only 0.1, which is a correlation measure between partially overlapped ranked lists. In contrast, $\mathbf{\Lambda}$ achieves a much higher overlap with BM25 of around 0.5 to 0.6.

6.2 Token-shuffled queries

Model	Original		Shuffled		Δ	
	@20	@100	@20	@100	@20	@100
DPR	77.4	84.7	69.4	80.1	8.0	4.6
BM25	62.3	76.0	62.3	76.0	0.0	0.0
Wiki $\mathbf{\Lambda}$	60.9	74.9	60.8	74.9	0.1	0.0
PAQ $\mathbf{\Lambda}$	62.7	76.4	62.6	76.2	0.1	0.2

Table 7: Lexical matching stress test on NQ Dev, using token-shuffled questions. $\mathbf{\Lambda}$ is order agnostic and maintains its performance on shuffled queries.

Next, we inspect the lexical matching capacity of $\mathbf{\Lambda}$ in an extreme case where the order of tokens in each question is randomly shuffled. Table 7 indicates that the performance of DPR drops significantly on this token-shuffled dataset, while the bag-of-word BM25 model remains completely unaffected. On the other hand, both Wiki $\mathbf{\Lambda}$ and PAQ $\mathbf{\Lambda}$ remain highly consistent on this challenge set, showing great robustness in lexical matching.

6.3 Hybrid SPAR + BM25 model

Acc@k on NQ	@20	@100
(**d**) DPR	79.5	86.1
(**h**) DPR + BM25	82.6	88.6
(**d**) SPAR-Wiki	83.0	88.8
(**h**) SPAR-Wiki + BM25	82.9	88.9
(**d**) SPAR-PAQ	82.7	88.6
(**h**) SPAR-PAQ + BM25	82.7	88.8

Table 8: The SPAR+BM25 model yields minimal gains over SPAR, indicating that SPAR does well in lexical matching and performs similarly to a hybrid model.

To confirm that SPAR improves DPR's performance by enhancing its lexical matching capability, we add the real BM25 to SPAR to create a hybrid model. As demonstrated in Table 8, adding BM25 to SPAR only results in minimal gains, which indicates that SPAR renders BM25 almost completely redundant and supports our main claim.

7 Conclusion

In this paper, we propose SPAR, a salient-phrase aware dense retriever, which can augment any dense retriever with the lexical matching capacity and out-of-domain generalization from a sparse retriever. This is achieved by training a dense Lexical Model $\mathbf{\Lambda}$ to imitate the behavior of the teacher sparse retriever, the feasibility of which remained unknown until this work. We show that SPAR outperforms previous state-of-the-art dense and sparse retrievers, matching or even exceeding more complex hybrid systems, on various in-domain and out-of-domain evaluation datasets.

For future work we plan to explore if a dense retriever can be trained to learn lexical matching directly without relying on a teacher model. This way, we can avoid imitating the errors of the sparse retriever, and devise new ways of training dense retrievers that can potentially surpass hybrid models. Moreover, there are several intriguing findings in this work that may warrant further study, such as why SPAR's Acc@k improves relatively to the hybrid model as k increases, and why joint training is less effective than post-hoc vector concatenation.

Limitations

There is a trade-off between accuracy and efficiency when considering the two variants of SPAR (Section 4.2). The Weighted Concat variant, the main focus of this paper, gives higher accuracy but results in longer query and passage embeddings. This in turn increases the index size and the retrieval time. On the other hand, the Weighted Sum variant does not increase the embedding size, but achieves a lower accuracy compared to the Weighted Concat variant as shown in Table 1.

As mentioned in Section 7, SPAR relies on a sparse teacher model to learn the Lexical Model$\mathbf{\Lambda}$. It is an intriguing direction for future work to explore whether we can learn SPAR from scratch without the help of a teacher retriever.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *ArXiv e-prints 1611.09268*.

Petr Baudiš and Jan Šedivý. 2015. Modeling of the question answering task in the yodaqa system. In *Proceedings of the 6th International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction - Volume 9283*, CLEF'15, page 222–228, Berlin, Heidelberg. Springer-Verlag.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Andrzej Bialecki, Robert Muir, and Grant Ingersoll. 2012. Apache lucene 4. In *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval, OSIR@SIGIR 2012, Portland, Oregon, USA, 16th August 2012*, pages 17–24. University of Otago, Dunedin, New Zealand.

Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1533–1536, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021a. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042, Online. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021b. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*, volume 12656 of *Lecture Notes in Computer Science*, pages 146–160. Springer.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 113–122, New York, NY, USA. Association for Computing Machinery.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

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

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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. In *Proceedings of the 43rd International ACM SIGIR Conference on Research*

and Development in Information Retrieval, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021a. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, Online. Association for Computational Linguistics.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021b. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Jimmy Lin and Xueguang Ma. 2021. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *ArXiv e-prints 2106.14807*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021a. *Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations*, page 2356–2362. Association for Computing Machinery, New York, NY, USA.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021b. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online. Association for Computational Linguistics.

Linqing Liu, Patrick Lewis, Sebastian Riedel, and Pontus Stenetorp. 2022. Challenges in generalization in open domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2014–2029, Seattle, United States. Association for Computational Linguistics.

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.

Xueguang Ma, Kai Sun, Ronak Pradeep, Minghan Li, and Jimmy Lin. 2022. Another look at dpr: Reproduction of training and replication of retrieval. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I*, page 613–626, Berlin, Heidelberg. Springer-Verlag.

Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wentau Yih, Barlas Oguz, Veselin Stoyanov, and Gargi Ghosh. 2021. Multi-task retrieval for knowledge-intensive tasks. 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 1098–1111, Online. Association for Computational Linguistics.

Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1723–1727, New York, NY, USA. Association for Computing Machinery.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. Technical report, University of Waterloo.

Barlas Oğuz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Wen tau Yih, Sonal Gupta, and Yashar Mehdad. 2022. Domain-matched pre-training tasks for dense retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2021. The curse of dense low-dimensional information retrieval for large index sizes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 605–611, Online. Association for Computational Linguistics.

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, page 232–241, Berlin, Heidelberg. Springer-Verlag.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148, Online and Punta Cana, Dominican Republic. 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)*.

William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems*, 28(4).

Xiang Wu, Ruiqi Guo, David Simcha, Dave Dopson, and Sanjiv Kumar. 2019. Efficient inner product approximation in hybrid spaces. *arXiv e-print 1903.08690*.

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*.

Nan Yang, Furu Wei, Binxing Jiao, Daxing Jiang, and Linjun Yang. 2021. xMoCo: Cross momentum contrastive learning for open-domain question answering. 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 6120–6129, Online. Association for Computational Linguistics.

Appendix A Implementation Details

For the Wiki training queries for $\mathbf{\Lambda}$, we randomly sample sentences from each passage (following the DPR passage split of Wikipedia) following a pseudo-exponential distribution while guaranteeing at least one sentence is sampled from each passage. The pseudo-exponential distribution would select more sentences in the first few passages of each Wikipedia document, as they tend to contain more answers, resulting in a collection of 37 million sentences (queries) out of 22M passages. We did not extensively experiment with sampling strategies, but one preliminary experiment suggested that sampling uniformly may have worked equally well. For the MS MARCO passage collection, we use all sentences as training queries without sampling, leading to a total of 28M queries out of 9M passages.

We train $\mathbf{\Lambda}$ for 3 days on 64 V100 GPUs with a per-GPU batch size of 32 and a learning rate of $3\mathrm{e}{-5}$ (roughly 20 epochs for Wiki $\mathbf{\Lambda}$ and MARCO $\mathbf{\Lambda}$, and 10 epochs for PAQ $\mathbf{\Lambda}$). The remaining hyper-parameters are the same as in DPR, including the BERT-base encoder and the learning rate scheduler. For Wiki and PAQ $\mathbf{\Lambda}$, we use NQ dev as the validation queries, and MS MARCO dev for MARCO $\mathbf{\Lambda}$. For the dense retrievers used in SPAR, we directly take the publicly released checkpoints without re-training to combine with $\mathbf{\Lambda}$. We use Pyserini (Lin et al., 2021a) for all sparse models used in this work including BM25 and UniCOIL.

For tuning the concatenation weights μ, we do a grid search on $[0.1, 1.0]$ (step size 0.1) as well as their reciprocals, resulting in a total of 19 candidates ranging from 0.1 to 10. The best μ is selected using the best R@100 for ODQA (§5.1) and MRR@10 for MS MARCO (§5.2) on the development set for each experiment.