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

Anonymous ACL submission

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. It has been argued that this is an inherent limitation of dense models. We disprove this claim by introducing the Salient Phrase Aware Retriever (SPAR), a dense retriever with the lexical matching capacity of a sparse model. In particular, we show that *a dense retriever* $\mathbf{\Lambda}$ *can be trained to imitate a sparse one*, and SPAR is built by augmenting a standard dense retriever with $\mathbf{\Lambda}$. When evaluated on five open-domain question answering datasets and the MS MARCO passage retrieval task, SPAR sets a new state of the art for dense and sparse retrievers and can match or exceed the performance of more complicated dense-sparse hybrid systems.

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 have 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?*". Similarly, Xiong et al. (2021) observe that ANCE fails to recognize "*active margin*" as a key phrase in the query "*What is an active margin*" and instead retrieves passages describing the financial term "*margin*". As a result, dense retrievers occasionally retrieve completely irrelevant results. In practice, therefore, hybrid retrieval systems are built that combine dense and sparse retrievers to enjoy the higher overall performance of dense retrievers while avoiding such failure cases. In fact, the predictions of DPR and BM25 are drastically different from each other, with only about 10% overlap between top-100 retrieved passages (discussed in §6.1.1). This suggests that a sparse retriever captures complementary information to a dense model, and its lexical matching capacity may further improve the performance of a dense retriever. For instance, new state-of-the-art performance on ODQA can be obtained using a hybrid system of DPR and a simple well-tuned BM25 (Ma et al., 2021).

Such a hybrid design, however, complicates the architecture of the retrieval system, increases latency, and takes more effort to maintain. Furthermore, as dense and sparse retrievers rely on different pieces of infrastructure for retrieval (text indexing for sparse retrievers and vector similarity search for dense ones), exact hybrid retrieval is computationally prohibitive. In reality, approximate hybrid search is performed such as using the one retriever to rerank the retrieval results of the other, leading to potentially inferior results.

In this work, we propose SPAR (Figure 1), a *dense* retriever with the lexical matching capacity

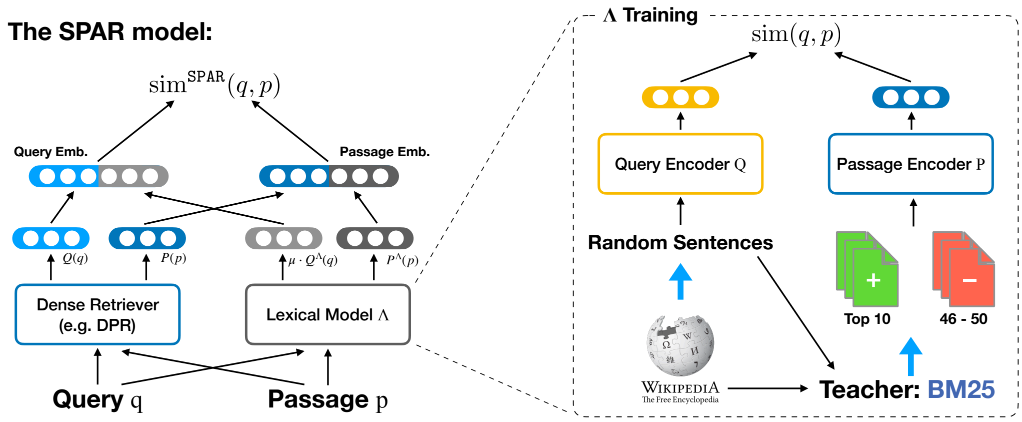

Figure 1: SPAR augments a dense retriever with a dense model $\mathbf{\Lambda}$ trained to imitate a sparse teacher retriever. $\mathbf{\Lambda}$ is trained using random sentences as queries with positive and negative passages generated from the teacher model. $\mathbf{\Lambda}$ is then combined with a dense retriever via vector concatenation to form a salient-phrase aware retriever (§4).

of a sparse model. SPAR has the high accuracy of a hybrid system without the need to maintain a complex pipeline of two separate retrieval architectures. In particular, we address an important yet largely unanswered research question: **Can we train a dense retriever to imitate a sparse one?** There have been theoretical and empirical studies suggesting a negative answer due to alleged intrinsic limitations of dense retrievers (Luan et al., 2021). For instance, Reimers and Gurevych (2021) show that dense retrievers prefer gibberish strings with random letters and spaces over the gold passage in up to 10% of cases, while it never happens to BM25. In addition, Sciavolino et al. (2021) find that DPR performs poorly compared to BM25 on simple entity-centric questions.

Contrary to these 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 model $\mathbf{\Lambda}$, and we build the SPAR model by combining $\mathbf{\Lambda}$ 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 in §4, by conducting an extensive study to answer two key questions: i) How to train $\mathbf{\Lambda}$ to imitate a sparse retriever (§4.1) and ii) How to best utilize $\mathbf{\Lambda}$ to build a salient-phrase aware dense retriever (§4.2).

We evaluate SPAR on five ODQA datasets (§5.1) as well as the MS MARCO (Bajaj et al., 2018) passage retrieval benchmark (§5.2), and show that it outperforms existing dense and sparse retrievers, matching or even beating the more complex hybrid systems. In addition, we conduct a series of analyses of $\mathbf{\Lambda}$ showcasing its lexical matching capability (§6.1). We also examine the generalization of SPAR showing strong **zero-shot** performance across datasets (§6.2), typically a weak point of dense retrievers, including on a new dataset of entity-centric questions (Sciavolino et al., 2021).

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) and to address vocabulary mismatch (Nogueira and Lin, 2019) and semantic mismatch (Gao et al., 2021), *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., 2021). 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); Liu et al. (2021) 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 a synthetic dataset of entity-centric questions (EntityQuestions) to highlight the failure of dense retrievers in matching key entities in the query. We perform a zero-shot evaluation on EntityQuestions in §6.2.1 and show that SPAR substantially outperforms DPR. The idea of using BM25 as weak supervision has been explored in IR (Dehghani et al., 2017), but their work predated dense retrieval, hence focusing on a different task (query-dependent ranking). As a result, the motivation and modeling are quite different from ours.

3 Preliminaries: Dense Retrieval

In this work, we adopt DPR (Karpukhin et al., 2020) as our dense retriever architecture for learning $\mathbf{\Lambda}$. We give a brief overview of DPR 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 the 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 loss: 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* (HN, irrelevant passages that are likely to be confused with positive ones) for more effective training[1].

We employ the DPR implementation from Oğuz et al. (2021), which supports efficient multi-node training as well as memory-mapped data loader, both important for the large-scale training of $\mathbf{\Lambda}$. We also adopt their validation metrics of mean reciprocal rank (MRR) on a surrogate corpus using one positive and one HN 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. At test time, the standard recall@k (R@k) metric is used for full evaluation, which is defined as the fraction of queries that has at least one positive passage retrieved in the top k.

4 The SPAR Model

In this section, we put forward our SPAR method, and explain how we arrive at our final approach by presenting pilot experiments on NaturalQuestions (NQ, Kwiatkowski et al., 2019), a popular ODQA dataset. A key ingredient of SPAR is the dense model $\mathbf{\Lambda}$ trained to imitate the prediction of a sparse retriever. As a proxy, we adopt MRR on a special validation set as a metric when training $\mathbf{\Lambda}$ to approximate how well $\mathbf{\Lambda}$ is imitating the sparse teacher model. In particular, we use questions from NQ dev as queries and the passage with the highest score from the sparse teacher model as positive, and thus a higher MRR on this set indicates a higher correlation between $\mathbf{\Lambda}$ and the teacher. We first investigate how to successfully train $\mathbf{\Lambda}$ (§4.1) and then study how to best leverage $\mathbf{\Lambda}$ to form a salient-phrase aware dense retriever (§4.2).

4.1 Training $\mathbf{\Lambda}$

In training $\mathbf{\Lambda}$, we are essentially distilling a sparse retriever into a bi-encoder model. There are many potential ways to do this, such as imitating the scores of the sparse retriever with the MSE loss or KL divergence, learning the passage ranking of the teacher while discarding the scores. After unsuccessful initial attempts with these methods, we instead adopt a very simple contrastive loss, inspired by DPR training, where passages ranked highly by the teacher model are taken as positives and those ranked lower as negatives.

In particular, using the top-k retrieved passages from the sparse retriever for a *given query* (§4.1.1),

[1]DPR uses BM25 to generate hard negatives.

we treat the top n_p as positives and the bottom n_n as hard negatives[2]. In our preliminary experiments, we find that $k = 50$ and $k = 100$ perform similarly, and n_n also does not affect the performance much as long as $n_n \geq 5$. Hence, we pick $k = 50$ and $n_n = 5$ in all experiments. On the other hand, n_p carries a significant impact on the success of $\mathbf{\Lambda}$ training, and will be discussed later in §4.1.2.

4.1.1 Training queries

Model	MRR
NQ $\mathbf{\Lambda}$	76.5
Wiki $\mathbf{\Lambda}$	92.4
Perturbed Wiki $\mathbf{\Lambda}$	92.5
PAQ $\mathbf{\Lambda}$	94.7

Table 1: $\mathbf{\Lambda}$ trained with various data regimes. MRR is calculated on questions from NQ dev, using one positive and one hard negative from BM25.

Now, we turn our attention to how the training queries are produced. Since a sparse retriever can be queried with arbitrary text, abundant choices are available. We consider two potentially helpful options, *in-domain* (queries similar to those in the downstream evaluation dataset) and *large-scale* training queries with an experiment on NQ. For in-domain queries, we use the questions from the NQ training set itself to generate the training data for $\mathbf{\Lambda}$. For large-scale queries, we randomly select sentences from each Wikipedia passage, resulting in a total of 37 million queries[3]. In addition, we experiment with questions from the PAQ dataset (Lewis et al., 2021b), a collection of 65 million synthetically generated *probably asked questions* for ODQA, which can be thought of as both in-domain and large-scale.

Table 1 shows how well $\mathbf{\Lambda}$ trained with each option can imitate the teacher BM25 model. The main takeaway is that the size of the query set is more consequential than the syntax of the queries, as both Wiki $\mathbf{\Lambda}$ and PAQ $\mathbf{\Lambda}$ outperform NQ $\mathbf{\Lambda}$ by a wide margin. Intuitively, large-scale training data is helpful for learning lexical matching as it gives the neural models more exposure to rare words and phrases. Finally, PAQ $\mathbf{\Lambda}$ further beats the Wiki $\mathbf{\Lambda}$, achieving the highest MRR score. However, as we shall see later, `SPAR` models built with Wiki and PAQ $\mathbf{\Lambda}$ achieve similar end performance as both are able to fully match the performance of a hybrid model, suggesting that carefully generated queries are not necessary for training $\mathbf{\Lambda}$. This makes our approach very appealing in practice as we can generate the training data of $\mathbf{\Lambda}$ cheaply, without relying on computationally intensive methods like PAQ.

[2]When using multiple positives for a query, one positive is randomly sampled at each training iteration of training.

[3]At least one sentence is sampled from each passage. We sample more sentences from the first few passages of each document following a pseudo-exponential distribution, as they tend to contain more answers. However, early experiment showed that sampling uniformly worked equally well.

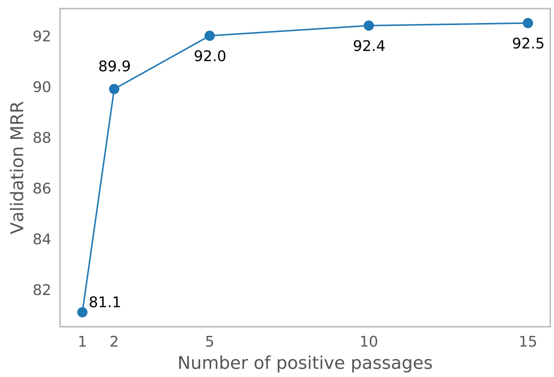

Figure 2: Validation MRR of the Wiki $\mathbf{\Lambda}$ using various numbers of positive passages.

Query Perturbation For Wiki $\mathbf{\Lambda}$, we initially hypothesized that query perturbation is needed to avoid exact phrase matches, which may improve the model robustness. We hence experimented with several perturbation strategies, including randomly dropping 15% of tokens, shuffling all tokens, and swap the order of the two chunks of text before and after a random anchor word. We found that all these perturbed variants, as well as a version that uses a mix of all perturbations (shown in Table 1), performed similarly to using the original sentences as queries. Therefore, we did not conduct any query perturbation in our final Wiki $\mathbf{\Lambda}$ model.

4.1.2 Number of positive passages

We now return to the selection of n_p, the number of positive passages per training query. As shown in Figure 2, using multiple positive passages 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 the gain is naturally diminishing as $\mathbf{\Lambda}$ is already imitating the teacher model closely. We use $n_p = 10$ for $\mathbf{\Lambda}$.

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

With a successfully trained $\mathbf{\Lambda}$, we now study how to build a salient-phrase aware retriever with it. We compare *implicit* and *explicit* use of $\mathbf{\Lambda}$, where the

Model	NQ	
	@20	@100
DPR	78.3	85.6
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**
2 DPRs (Weighted Concat)	79.8	86.4
DPR + BM25	80.9	87.9

Table 2: 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).

latter explicitly employs $\mathbf{\Lambda}$ in the final representation space when computing similarity. An example of implicit use of $\mathbf{\Lambda}$ is using it as initialization for DPR training with the hope that DPR can inherit the lexical matching capacity from $\mathbf{\Lambda}$. Unfortunately, this is ineffective as shown in Table 2.

Explicit use of $\mathbf{\Lambda}$ can be done either during training or inference time. The more straightforward inference-time utilization involves combining the vectors of the trained $\mathbf{\Lambda}$ with a separately trained dense retriever such as DPR at inference time. The vectors from the two models can be combined using various pooling methods such as summation or concatenation. Since both models are dense retrievers, this *post-hoc* combination can be easily done without affecting the architecture of retrieval index. Table 2 indicates that concatenation performs better, but summation has the benefit of not increasing the dimension of the final vectors.

Finally, instead of training the two models separately, 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 2.

We also compare with the ensemble (weighted concatenation of embeddings) of two DPR models trained with different random seeds, which has the same dimension and number of parameters as SPAR. While SPAR beats it substantially, this baseline does show non-trivial gains over DPR. We leave further investigation of the intriguing efficacy of *post-hoc* vector concatenation to future work.

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, 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:

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

μ is tuned on the validation set, as detailed in Appendix B. Note that our decision of adding μ to the query vectors can potentially support dynamic or query-specific weights without the need to change the index, but we leave this for future work.

Our final SPAR model is a general recipe 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 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 FAISS index and retrieval can be done in the same way as a standard dense retriever. Further implementation details are in Appendix B.

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.,

	NQ		SQuAD		TriviaQA		WebQ		TREC		**Average**	
Model	R@20	R@100	R@20	R@100	R@20	R@100	R@20	R@100	R@20	R@100	R@20	R@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												
(**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., 2021)

Table 3: R@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**, while the highest among dense retrievers is underlined.

2020), including the train, dev and test splits, and the Wikipedia passage collection.

Table 3 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).[4] SPAR outperforms all state-of-the-art retrievers in the literature, usually by wide margins, demonstrating the effectiveness of our approach. Notably, SPAR even performs better than some recent approaches with sophisticated multi-stage training that are more complicated and expensive (Qu et al., 2021).

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. In contrast, SPAR dramatically improves over previous models, achieving an improvement of 13.6 points in R@100 over xMoCo, the best dense retriever on SQuAD.

PAQ **Λ** matches the accuracy of the teacher BM25 model, while Wiki **Λ** 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 0.3% better on average than SPAR-Wiki (88.0 vs. 87.7). This enables us to go with the much cheaper Wiki option for training **Λ** without sacrificing much of the end performance.

[4]The *cross-dataset model generalization* section in Table 3 and 4 will be discussed in §6.2.

5.2 MS Marco Passage Retrieval

	MS MARCO Dev	
Model	MRR@10	R@1000
(**s**) BM25	18.7	85.7
(**s**) UniCOIL (Lin and Ma, 2021)	35.2	95.8
(**d**) MARCO BM25 **Λ**	17.3	83.1
(**d**) MARCO UniCOIL **Λ**	34.1	97.0
(**d**) ANCE (Xiong et al., 2021)	33.0	95.9
(**d**) TCT-ColBERT (Lin et al., 2021b)	35.9	97.0
(**d**) RocketQA (Qu et al., 2021)	37.0	97.7
(**h**) RocketQA + BM25	38.1	98.0
(**h**) RocketQA + UniCOIL	**38.8**	97.3
(**d**) SPAR (RocketQA + **Λ**=BM25)	37.9	98.0
(**d**) SPAR (RocketQA + **Λ**=UniCOIL)	38.6	**98.5**
Cross-dataset model generalization		
(**d**) Wiki BM25 **Λ**	15.8	78.8
(**d**) SPAR (RocketQA + **Λ**=Wiki BM25)	37.7	98.0

Table 4: SPAR results on MS MARCO passage retrieval. We consider several options for **Λ**, trained with different objectives (BM25 and UniCOIL) and different corpora (MSMARCO and Wikipedia). Full results are available in Table 9.

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. To highlight the versatility of our approach, we adopt two dense retrievers in SPAR, ANCE and RocketQA. (The full results with ANCE experiments and more metrics are shown in Table 9 in the

Appendix.) We further consider two sparse retrievers for training $\mathbf{\Lambda}$, BM25 and UniCOIL (Lin and Ma, 2021), a recent state-of-the-art (SoTA) sparse retriever, to study whether SPAR training can unpack the knowledge from a more advanced teacher model. Similar to the Wiki training queries, we create a MARCO corpus for training $\mathbf{\Lambda}$. Details can be found in Appendix B.

As demonstrated in Table 4, SPAR is able to augment RocketQA with the lexical matching capacity from either BM25 or UniCOIL, leading to a performance similar to the hybrid retriever and again outperforming all existing dense and sparse retrievers with a MRR@10 of 38.6 (see row group 5 and 4). The fact that SPAR works with not only DPR and BM25, but other SoTA dense and sparse retrievers makes SPAR a general solution for combining the knowledge of dense and sparse retrievers in a single dense model.

One intriguing observation is that SPAR works better when retrieving more candidates. SPAR is slightly worse than the hybrid models on MRR@10, but catches up on R@50 (Table 9), and even outperforms the hybrid ones when considering R@1000. This is also observed on ODQA experiments between R@20 and R@100.

Another 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 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 real hybrid model has to rely on approximation. We leave further investigation in these curious findings to future work.

6 Discussions

In this section, we dive deeper into the $\mathbf{\Lambda}$ model with a number of analyses on its behaviors.

6.1 Does $\mathbf{\Lambda}$ Learn Lexical Matching?

The first and foremost question is whether $\mathbf{\Lambda}$ actually learns lexical matching. We conduct a series of direct and indirect analyses to get more insights.

6.1.1 Rank Biased Overlap with BM25

We first directly compare the predictions of $\mathbf{\Lambda}$ against BM25 using rank biased overlap (RBO, Webber et al., 2010), a similarity measure of ranked lists. As shown in Table 5, 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. It also confirms that query perturbation does not make a difference in $\mathbf{\Lambda}$ training as the two variants of Wiki $\mathbf{\Lambda}$ perform almost identically.

RBO w/ BM25	NQ	SQuAD	Trivia
DPR	.104	.078	.170
Wiki $\mathbf{\Lambda}$	.508	.452	.505
Perturbed Wiki $\mathbf{\Lambda}$	.506	.453	.504
PAQ $\mathbf{\Lambda}$	.603	.478	.527

Table 5: 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.

6.1.2 Stress test: token-shuffled questions

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 6: Lexical matching stress test on NQ Dev, using token-shuffled questions. $\mathbf{\Lambda}$ maintains its performance on randomly 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 6 indicates that the performance of DPR drops significantly on this token-shuffled dataset, while the bag-of-word BM25 model remains unaffected by design. 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.

6.1.3 Hybrid SPAR + BM25 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 7, adding BM25 to SPAR only results in minimal gains, suggesting that the improvement SPAR made to DPR is indeed due to better lexical matching. The results indicate that SPAR renders BM25 almost completely redundant, supporting our main claim.

Model	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 7: The SPAR+BM25 model yields minimal gains over SPAR, indicating that SPAR does well in lexical matching and performs similarly to a hybrid model.

6.2 Generalization of $\boldsymbol{\Lambda}$ and SPAR

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

In the *cross-dataset model generalization* section of Table 3 and 4, we reported SPAR performance on ODQA using the $\boldsymbol{\Lambda}$ model built for MS MARCO and vice versa. In both directions, $\boldsymbol{\Lambda}$ has a high zero-shot performance, and the SPAR performance is close to that using in-domain $\boldsymbol{\Lambda}$. This suggests that $\boldsymbol{\Lambda}$ shares the advantage of better generalization of a sparse retriever, and it may not be always necessary to retrain $\boldsymbol{\Lambda}$ on new datasets.

6.2.1 Zero-shot performance on EntityQuestions

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

Table 8: 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) 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 8, 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. $\boldsymbol{\Lambda}$ transfers much better to this dataset than DPR, achieving a slightly lower performance than BM25. When $\boldsymbol{\Lambda}$ is combined with DPR in SPAR, the gap is once again bridged, and SPAR achieves a higher R@20 than the hybrid model, and overall an improvement of 17.4 points over DPR.

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 from any given sparse retriever. This is achieved by training a dense model $\boldsymbol{\Lambda}$ to imitate the behavior of the teacher sparse retriever, the feasibility of which remained unknown until this work. In the experiments, we show that SPAR outperforms previous state-of-the-art dense and sparse retrievers, matching or even beating more complex hybrid systems, on five open-domain question answering benchmarks and the MS MARCO passage retrieval dataset. Furthermore, SPAR is able to augment any dense retriever such as DPR, ANCE or RocketQA, with knowledge distilled from not only BM25, but more advanced sparse retrievers like UniCOIL, highlighting the generality of our approach. In addition, we show that $\boldsymbol{\Lambda}$ indeed learns lexical matching, and generalizes well to other datasets in a zero-shot fashion.

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 R@k improves relatively to the hybrid model as k increases, and why joint training is less effective than post-hoc vector concatenation.

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.

Petr Baudiš and Jan Šedivý. 2015. Modeling of the question answering task in the yodaqa system. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 222–228, Cham. Springer International Publishing.

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.

Zhuyun Dai and Jamie Callan. 2020. *Context-Aware Term Weighting For First Stage Passage Retrieval*, page 1533–1536. Association for Computing Machinery, New York, NY, USA.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 65–74, 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. 2021. 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.

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. *arXiv preprint*.

Jimmy Lin and Xueguang Ma. 2021. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques.

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. 2021. Challenges in generalization in open domain question answering. *arXiv preprint*.

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, and Jimmy Lin. 2021. A replication study of dense passage retriever. *arXiv preprint*.

Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen-tau 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.

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. 2021. Domain-matched pre-training tasks for dense retrieval. *arXiv preprint*.

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.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94*, pages 232–241, London. Springer London.

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 (EMNLP-2021)*.

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

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

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, Daxiang 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.

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)	<u>38.6</u>	<u>86.3</u>	<u>**98.5**</u>
Cross-dataset model generalization			
(**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 9: `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.

A Complete Results for MS MARCO Passage Retrieval

Table 9 shows the full results on MS MARCO, with an additional column reporting R@50 in addition to MRR@10 and R@1000. In Table 9, two state-of-the-art dense retrievers are considered in `SPAR`, namely ANCE and RocketQA. The conclusions drawn for RocketQA in Table 4 in the main paper stays the same for ANCE, further strengthening the generality of our approach.

B 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. However, early experiment showed that sampling uniformly worked almost 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 GPUs with a per-GPU batch size of 32 and a learning rate of 3e$-$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.