BM25 Query Augmentation Learned End-to-End

Anonymous ACL submission

Abstract

Given BM25's enduring competitiveness as an information retrieval baseline, we investigate to what extent it can be even further improved by augmenting and re-weighting its sparse query-vector representation. We propose an approach to learning an augmentation and a re-weighting end-to-end, and we find that our approach improves performance over BM25 while retaining its speed. We furthermore find that the learned augmentations and re-weightings transfer well to unseen datasets.

1 Introduction

Despite the enormous progress in neural information retrieval (IR) techniques over the last several years (Karpukhin et al., 2020; Xiong et al., 2020; Dai and Callan, 2019, *inter alia*), simply using BM25 (Robertson et al., 1995; Crestani et al., 1998) remains a strong baseline approach (Thakur et al., 2021; Sciavolino et al., 2021). This is especially the case if it is important that retrieval be fast, or that the retrieval method not consume too much memory or disk space. Given the effectiveness of this rather simple baseline, it is natural to wonder whether BM25 might become an even *more* competitive baseline with a bit of additional engineering.

One straightforward approach to improving BM25 while retaining its favorable properties is to do query augmentation — that is, to augment the query with additional tokens, thereby improving the quality of the retrieved documents. Notably, Nogueira and Cho (2017) propose to learn query augmentations using reinforcement learning (RL) techniques, with recall-at-K as the reward. While query augmentations are indeed discrete (consisting of word tokens), and therefore certainly congenial to RL methods, we show that augmentations can be much more simply learned end-to-end, in the course of minimizing a standard contrastive loss.

We find that our augmentation approach improves performance, while allowing for retrieval that is as fast, and sometimes even faster, than BM25. Moreover, our approach can be realized by simply modifying the query and IDF vectors (respectively), which allows for doing retrieval with standard tools, such as Pyserini (Lin et al., 2021). In particular, unlike many recent sparse approaches to retrieval (e.g., SPLADE (Formal et al., 2021)) we do not need to re-encode the documents.

Finally, we find that our approach transfers well between datasets. This is an encouraging finding, since transfer can be challenging for neural IR methods, while it is typically not a challenge for BM25. Code for reproducing all models and experiments is available at `hidden.for.submission`.

2 Augmenting BM25

Recall that BM25 (Robertson et al., 1995; Crestani et al., 1998) scores the compatibility between a query q, viewed as a set of tokens, and a document d, also a set. Given a collection $\mathcal{D}$ of N documents, and assuming a vocabulary $\mathcal{V}$, let $\mathbf{v} \in \mathbb{R}^{|\mathcal{V}|}$ be the document collection's inverse document-frequency (IDF) vector, defined as

$$v_i = \log(\frac{N - |\{d \in \mathcal{D} \mid w_i \in d\}| + 0.5}{|\{d \in \mathcal{D} \mid w_i \in d\}| + 0.5} + 1),$$

where $|\{d \in \mathcal{D} \mid w_i \in d\}|$ counts the number of documents in the collection in which word type w_i appears. Also let $\mathbf{f}(d) \in \mathbb{R}^{|\mathcal{V}|}$ be a document d's BM25 term-frequency vector, defined as

$$\mathrm{f}(d)_i = \frac{\mathrm{count}(w_i, d)(k+1)}{\mathrm{count}(w_i, d) + k(1 - b + \frac{b|d|}{M})},$$

where $\mathrm{count}(w_i, d)$ counts the number of times word type w_i appears in document d, k and b are hyperparameters, and M is the average length of the documents in the collection. Finally, let $\mathbf{bow}(q) \in \{0,1\}^{|\mathcal{V}|}$ be the binary bag-of-words representation of q, where $\mathrm{bow}(q)_i$ is 1 if $w_i \in q$, and is 0 otherwise. The BM25 score between q and d can then be written as

$$\mathrm{BM25}(q,d) = (\mathbf{v} \odot \mathbf{bow}(q))^\top \mathbf{f}(d),$$

where $\odot$ is element-wise multiplication.

Augmenting differentiably To augment BM25's query vector, one can produce an additional set of tokens $\hat{q}$, and then score documents with $(\mathbf{v} \odot \mathbf{bow}(q \cup \hat{q}))^\top \mathbf{f}(d)$. While $\hat{q}$ is of course discrete, we observe that if we are willing to rescale $\mathbf{v}$ element-wise by some additional vector $\mathbf{c} \in \mathbb{R}^{|\mathcal{V}|}$, we have

$$\mathbf{c} \odot \mathbf{v} \odot \mathbf{bow}(q \cup \hat{q}) = \mathbf{v} \odot (\mathbf{bow}(q) + \mathbf{a})$$

for $\mathbf{a} \in \mathbb{R}^{|\mathcal{V}|}$, so long as $a_i = c_i \mathrm{bow}(q \cup \hat{q})_i - \mathrm{bow}(q)_i$. Moreover, we have that $a_i \neq 0$ only if $\mathrm{bow}(q \cup \hat{q})_i = 1$. Thus, we can predict $\mathbf{a}$ as our *non-discrete* augmentation vector, and score documents with

$$(\mathbf{v} \odot (\mathbf{bow}(q) + \mathbf{a}))^\top \mathbf{f}(d),$$

and this will be equivalent to scoring with $(\mathbf{c} \odot \mathbf{v} \odot \mathbf{bow}(q \cup \hat{q}))^\top \mathbf{f}(d)$ for some $\mathbf{c}$.

In practice, we use a model to produce both the $\mathbf{a}$ vector as well as an additional element-wise weighting vector $\mathbf{w}$, as parameterized functions of the query q. This leads to the following final scoring function:

$$\mathrm{score}(q,d) = (\mathbf{w}(q) \odot \mathbf{v} \odot (\mathbf{bow}(q) + \mathbf{a}(q)))^\top \mathbf{f}(d), \tag{1}$$

where we have written $\mathbf{w}(q)$ and $\mathbf{a}(q)$ to emphasize that these are (parameterized) functions of the query. This scoring function is differentiable with respect to both $\mathbf{a}(q)$ and $\mathbf{w}(q)$, and we can therefore train the models producing these vectors with a standard objective, described below. We also describe below how to regularize $\mathbf{a}(q)$ to ensure it is sparse.

Retrieval Since a_i is nonzero only when q or $\hat{q}$ contains w_i, at retrieval time we can extract the augmented set $q \cup \hat{q}$ from $\mathbf{a}$ to be our augmented query. If we then define a new IDF vector $\mathbf{v}'$ with elements $v'_i = w_i v_i (\mathrm{bow}(q)_i + a_i)$, we can use standard BM25 implementations to retrieve the highest-scoring documents using $\mathbf{v}'$ as the IDF vector, and this will be equivalent to retrieving documents under $\mathbf{c} \odot \mathbf{v} \odot \mathbf{bow}(q \cup \hat{q})$ for some $\mathbf{c}$.[1]

[1]Some implementations of BM25, such as Pyserini's, allow rescaling IDF values directly, which is what we do.

Parameterization We feed a linearized q into a pretrained BERT-like (Devlin et al., 2019) encoder, after first prepending to it a [CLS] token. Let $\mathbf{enc}(q)_0 \in \mathbb{R}^E$ be the encoder's representation of the [CLS] token, and let $\mathbf{enc}(q)_i \in \mathbb{R}^E$ be the encoder's representation of the i-th token in the query, for $i \geq 1$. We then parameterize $\mathbf{a}$ and $\mathbf{w}$ as follows:

$$\mathbf{a}(q) = \mathrm{ReLU}(\mathbf{W}\,\mathbf{enc}(q)_0)$$

$$w(q)_i = \mathrm{ReLU}(\mathbf{u}^\top \mathbf{enc}(q)_i),$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times E}$ and $\mathbf{u} \in \mathbb{R}^E$.

Training We train end-to-end, using a standard contrastive loss:

$$\mathcal{L}_{rank} = -\log \frac{\exp(\mathrm{score}(q, d^+))}{\sum_{d' \in \mathcal{D}^- \cup \{d^+\}} \exp(\mathrm{score}(q, d'))},$$

where d^+ is a positive document provided by the training dataset, and $\mathcal{D}^-$ consists of hard negatives mined by BM25 as well as in-batch negatives, following the approach of Karpukhin et al. (2020).

Sparse regularization Note that in practice, BM25's retrieval speed depends on how many distinct word types appear in the query. In order to promote retrieval efficiency, we regularize $\mathbf{a}(q)$ to ensure it is sparse. We found it beneficial to encourage sparsity *more* for frequent words, which are less discriminative. We therefore weight the $L1$ regularization per word by a monotonic function of its document frequency. In particular, we use:

$$\mathcal{L}_{reg} = \mathrm{sqrt}(\mathbf{h})^\top \mathbf{a}(q),$$

where $h_i = \frac{|\{d \in \mathcal{D} | w_i \in d\}|}{N}$, and where the square-root is applied elementwise. Our final training loss is then $\mathcal{L} = \mathcal{L}_{rank} + \lambda \mathcal{L}_{reg}$.

3 Experiments

Datasets We evaluate the retrieval performance of our proposed method on Natural Questions (NQ; Kwiatkowski et al., 2019), EntityQuestions (Sciavolino et al., 2021), and MSMARCO passage ranking task (Bajaj et al., 2016). We use TriviaQA (Joshi et al., 2017) and EntityQuestions to test out-of-distribution retrieval.

Experimental Details We initialized all models with distilbert-base-uncased (Sanh et al., 2019), using the Hugging Face implementation (Wolf et al., 2020). On NQ, models were trained with the AdamW optimizer (Kingma and Ba, 2015; Loshchilov and Hutter, 2018), using a learning rate of $3e^{-4}$, a batch size of 144, and a value for λ of 0.1. We trained the models for 45 epochs with 1 hard negative per sample. We used the same settings for training on EntityQuestions but only trained for 10 epochs. On MSMARCO, we used $3e^{-5}$ for the learning rate, 144 for the batch size, and 0.025 for λ. Models were trained for 2 epochs with 4 hard negatives per sample. We trained and evaluated our models on a single A6000 GPU. Training took about 70 minutes on NQ and 30 minutes on MSMARCO. Documents were tokenized by BERT's WordPiece tokenizer and indexed by Pyserini (Lin et al., 2021). We also used Pyserini for retrieval at test time.

Tokenization We emphasize that the performance of standard BM25 depends on the tokenization used. Since we use a distilbert model, we must tokenize queries and documents with a WordPiece tokenizer (Kudo, 2018; Devlin et al., 2019). Because this is not the default tokenization employed by Pyserini, we report baseline BM25 numbers using both tokenizations. We refer to BM25 with the default Pyserini tokenization as "BM25 (Pyserini)" and BM25 with the WordPiece tokenization as "BM25 (Ours)."

3.1 Retrieval Performance

We report results and latencies on NQ, EntityQuestions, and MSMARCO in Table 1. We measure per-query latency using wall-clock time on the same machine. On NQ, our method improves 12.1 percentage points in top-5 retrieval accuracy over the vanilla BM25 while adding only 43 milliseconds in latency. Compared to GAR (Mao et al., 2021), an alternative query augmentation method that autoregressively predicts the target document given a query, our method retrieves much more quickly and performs only slightly worse.

On MSMARCO and EntityQuestions, our method consistently improves over the baseline BM25. Notably, our method achieves lower latency than BM25 on MSMARCO since our weighting allows skipping some terms in the query by setting corresponding $w(q)_i$ to 0. On EntityQuestions, a dataset designed to demonstrate the inconsistency of dense retrievers, our method outperforms both BM25 and DPR.

NQ	Acc@5	Acc@20	Latency
BM25 (Pyserini)	0.436	0.629	0.099s
BM25 (Ours)	0.430	0.589	0.103s
GAR+BM25	0.609	0.744	5min
DPR	0.668	0.781	30min
SEAL	0.613	0.762	35min
Ours	0.557	0.694	0.146s
EntityQuestions	Acc@5	Acc@20	Latency
BM25 (Pyserini)	0.616	0.720	0.060s
BM25 (Ours)	0.526	0.637	0.094s
DPR	-	0.684	-
Ours	0.693	0.798	0.669s
MSMARCO	NDCG@10	R@100	Latency
BM25 (Pyserini)	0.228	0.658	0.020s
BM25 (Ours)	0.217	0.623	0.031s
SPLADE	0.433	-	1.764s
Ours	0.251	0.687	0.030s

Table 1: Retrieval results on NQ, EntityQuestions, and MSMARCO. Non-BM25 baselines include GAR+BM25 (Mao et al., 2021), DPR (Karpukhin et al., 2020), and SEAL (Bevilacqua et al., 2022) on NQ, DPR on EntityQuestions, and SPLADE (Formal et al., 2021) on MSMARCO. Results and latencies for non-BM25 methods are taken from their respective papers. Latency for DPR on NQ is reported by Mao et al. (2021); the latency number is unavailable for DPR on EntityQuestions. Latency for SPLADE is measured with the Pyserini implementation.

3.2 Transfer Results

One of the major concerns relating to language-model-assisted retrieval is that it may not generalize well out-of-distribution. To test our method in a transfer setting, we select the best model trained on NQ, and test it on TriviaQA and EntityQuestions without further fine-tuning. We show the results of this experiment in Table 3, where we see that our method generalizes to both datasets, and improves from the baseline by 2-3 percentage points. At the same time, it is clear from comparing the results of BM25 (Pyserini) with BM25 (Ours) that the Pyserini tokenization is helpful for these datasets. We anticipate being able to further improve given a pretrained model using the preferred tokenization, which we leave to future work.

3.3 Ablation Study

In Table 2 we present the results of ablating various aspects of our approach on the NQ dataset. Our full setting achieves the best balance between accuracy and efficiency. Compared to a uniform $L1$ penalty, our $L1$ penalty weighted by document frequency achieves lower latency while predicting augmen-

	Accuracy@5	Accuracy@20	Latency	Augmentation Length
Full Setting	0.557	0.694	0.146	12.334
- w/o Weighted $L1$	0.562	0.704	0.268	15.211
- w/o Weight	0.545	0.683	0.269	19.165
- w/o BM25 Scoring	0.487	0.635	0.225	31.861
- w/o BM25 Scoring & Weighted $L1$	0.525	0.670	0.377	21.794

Table 2: Ablation experiments on NQ. From top to bottom, we consider our approach with a uniform weighting of the $L1$ penalty (rather than by word frequency), without the elementwise weight vector $\mathbf{w}$, using just a bag-of-words rather than BM25-style query and document representations, and with both uniform $L1$ and bag-of-words representations.

	Accuracy@5	Accuracy@20
TriviaQA		
BM25 (Pyserini)	0.677	0.773
BM25 (Ours)	0.636	0.742
Ours	0.662	0.755
EntityQuestions		
BM25 (Pyserini)	0.616	0.720
BM25 (Ours)	0.526	0.637
Ours	0.542	0.656

Table 3: Results of our approach trained on NQ and then transferred to TriviaQA (top) and EntityQuestions (bottom).

tations of similar lengths. This suggests that the model is augmenting with rarer terms, thereby reducing the total number of documents in the inverted index and increasing speed. We also see that the additional weighting $\mathbf{w}$ is helpful. Finally, we check whether using BM25-style query and document vectors, rather than simple bag-of-words vectors, is important, by replacing the scoring function (1) with the following:

$$(\mathbf{w}(q) \odot (\mathbf{bow}(q) + \mathbf{a}(q)))^\top \mathbf{bow}(d).$$

We observe that this also decreases performance.

4 Related Work

While recent dense retrievers have shown strong retrieval performance (Reimers and Gurevych, 2019; Karpukhin et al., 2020), their high latencies limit their application to first-stage retrieval. To improve efficiency, late-interaction approaches have been proposed (Khattab and Zaharia, 2020; Gao et al., 2021; Formal et al., 2021). Here, documents are first retrieved with an inverted-index and then scored by aggregating term embeddings pre-computed while indexing. Although such methods reduce retrieval latencies, the need to store dense representations of documents significantly increases index sizes (Thakur et al., 2021).

Document expansion methods, such as Doc2query (Nogueira et al., 2019; Nogueira and Lin, 2019), allow indexing and retrieval using standard BM25. Nogueira and Lin (2019) use a language model to predict possible queries given a document; augmented documents are then constructed by appending these queries. By adding to the document, this approach addresses the term mismatching issue affecting sparse retrieval methods. However, this approach also requires running a language model over every document, which is expensive, especially if new documents are added incrementally. It is also potentially infeasible to do this when documents are very long. In contrast, we restrict our method to only perform neural operations on the queries, which are usually much shorter than documents.

To the best of our knowledge, only a few recent methods meet this requirement of only modifying queries. Nogueira and Cho (2017) use reinforcement learning to predict discrete augmentations. GAR (Mao et al., 2021) and SEAL (Bevilacqua et al., 2022) train language models to generate target documents or n-grams. Our approach differs from these methods by optimizing for BM25 retrieval in an end-to-end fashion, and in being significantly faster.

5 Conclusion

We propose a novel approach for learning to augment BM25 end-to-end with a language model. Our method improves over BM25 on three different datasets while retaining its efficiency. Additionally, we show that such improvements are able to generalize out-of-distribution. With its simple formulation, our method can be easily integrated into existing sparse retrieval frameworks. And we believe it might serve as a stronger sparse baseline for future work in retrieval.

Limitations

As mentioned in Sec. 3.2, tokenization methods heavily influence retrieval performance. This is a limitation both of BM25 and of our modification of it. In its current form, there are no straightforward solutions that allow our method to augment queries with words rather than the subword tokens of the pretrained tokenizer. Thus, in situations where word-tokenization is important for BM25 (some of which appear in Table 1), using our method would require pre-training a word- rather than subword-based model, which may be difficult.

Relatedly, since our method makes fundamental use of a pretrained model as the backbone, it suffers from the same problems afflicting pretrained models, including susceptibility to misinformation and bias, and requiring significant computational resources.

Ethics Statement

As our approach attempts to improve retrieval technology, the ethical considerations are similar to those of other retrieval technologies, especially those utilizing large pretrained language models. In particular, there is always a risk that the documents retrieved by our approach will be influenced by errors or biases in the underlying model, and it is necessary to ensure this does not happen before deployment. Because our augmentations are token-based, rather than based on dense representations, it should be slightly easier to manually check whether augmentations are problematic. We also emphasize that our approach is relatively undemanding computationally, which we believe to be a positive feature.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. In *Advances in Neural Information Processing Systems*.

Fabio Crestani, Mounia Lalmas, Cornelis J Van Rijsbergen, and Iain Campbell. 1998. "is this document relevant?... probably" a survey of probabilistic models in information retrieval. *ACM Computing Surveys (CSUR)*, 30(4):528–552.

Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*.

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.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086*.

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.

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*, pages 39–48.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

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*, 7:453–466.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval 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 4089–4100, Online. Association for Computational Linguistics.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 574–583, Copenhagen, Denmark. Association for Computational Linguistics.

Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. *Online preprint*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

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)*.

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

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

A Dataset Statistics

Dataset	Train	Dev	Test
Natural Questions	58,880	8,757	3,610
EntityQuestions	176,560	22,068	22,075
TriviaQA	60,413	8,837	11,313
MSMARCO	502,939	6,980	-

Table 4: Number of train/dev/test queries on each dataset. On MSMARCO, we follow the same approach as the previous work that reports the dev set performance of BEIR (Thakur et al., 2021).

B License

All packages and datasets used in our study are released with Apache-2.0 or MIT licenses.