Simple and Effective Unsupervised Redundancy Elimination to Compress Dense Vectors for Passage Retrieval

Xueguang Ma*, **Minghan Li***, **Kai Sun, Ji Xin,** and **Jimmy Lin**

David R. Cheriton School of Computer Science
University of Waterloo

{x93ma, m692li, k49sun, ji.xin, jimmylin}@uwaterloo.ca

Abstract

Recent work has shown that dense passage retrieval techniques achieve better ranking accuracy in open-domain question answering compared to sparse retrieval techniques such as BM25, but at the cost of large space and memory requirements. In this paper, we analyze the redundancy present in encoded dense vectors and show that the default dimension of 768 is unnecessarily large. To improve space efficiency, we propose a simple unsupervised compression pipeline that consists of principal component analysis (PCA), product quantization, and hybrid search. We further investigate other supervised baselines and find surprisingly that unsupervised PCA outperforms them in some settings. We perform extensive experiments on five question answering datasets and demonstrate that our best pipeline achieves good accuracy–space trade-offs, for example, $48\times$ compression with less than 3% drop in top-100 retrieval accuracy on average or $96\times$ compression with less than 4% drop. Code and data are available at http://pyserini.io/.

1 Introduction

Dense passage retrieval (DPR; Karpukhin et al., 2020) improves end-to-end retrieval accuracy in open-domain question answering (QA) by representing queries and documents in a low-dimensional, dense vector space. However, the vastly increased space and memory demands for storing and loading the dense vectors call for effective compression methods (Izacard et al., 2020). For example, the size of the original DPR vector (flat) index on the Wikipedia corpus is about 61 GB, while its sparse counterpart—BM25 inverted index—only uses 2.4 GB. The staggering increase of around $25\times$ in space requirements only yields an average gain of 2.5% in top-100 accuracy across five datasets (Ma et al., 2021), indicating potential redundancy in the dense representations.

In this work, we quantify redundancy within the dense vectors encoded by the DPR model using explained variance ratio and mutual information. Figure 1 shows that the original 768 dimensions is unnecessarily large as both the mutual information and explained variance ratio start to plateau at around 256 dimensions. Based on this observation, we further propose a simple yet effective pipeline for dense retrieval compression that includes principal component analysis (PCA), product quantization (PQ), and hybrid search to reduce index size while retaining effectiveness.

We also compare other compression options, including supervised dimensionality reduction, where we fine-tune a linear projection layer on top of the pre-trained DPR model using relevance labels. Surprisingly, we find that PCA achieves top-100 retrieval accuracy that is better than the two supervised dimensionality reduction techniques for 256 and 128 dimensions, while supervised techniques outperform (unsupervised) PCA for 64 dimensions. Our techniques support different accuracy–space trade-offs, but one sweet spot manages to compress the dense vectors by $96\times$ with less than 4% drop in top-100 accuracy on average across five standard QA datasets. Finally, we incorporate our pipeline with the BM25 inverted index for sparse–dense hybrid search, where we can achieve $16\times$ compression without any accuracy drop compared to the original DPR results.

2 Background and Related Work

Sparse retrieval methods such as BM25 (Robertson and Zaragoza, 2009; Yang et al., 2017) have established strong baselines in open-domain QA (Chen et al., 2017; Yang et al., 2019). Recently, dense retrieval emerges as a promising alternative (Karpukhin et al., 2020; Zhan et al., 2020; Xiong et al., 2021; Hofstätter et al., 2020; Lin et al.,

* Equal contribution

2020) in end-to-end question answering, but at the cost of increased space requirements. Efforts have been made towards developing memory efficient baselines (Izacard et al., 2020), but the topic still remains under-explored. In the following, we briefly introduce how dense retrieval works during training and inference.

Given a collection of passages and a QA task, DPR (Karpukhin et al., 2020) adopts a bi-encoder structure where encoders $f_Q(\cdot)$ and $f_D(\cdot)$ are independent BERT (Devlin et al., 2019) models that encode questions/passages into dense vectors. The relevance between the question q and passage d is defined by the dot product between their corresponding vectors as $v_q^\top v_d$, where $v_q = f_Q(q)$ and $v_d = f_D(d)$. The relevance score is used to rank the passages during retrieval with nearest neighbor search techniques. During training, given a question q, a positive passage d^+ that contains the answer for q, and m negative passages $d_1^-, d_2^-, ...d_m^-$, the training objective is:

$$\begin{aligned}
&\mathcal{L}(q, d^+, d_1^-, d_2^-, \cdots, d_m^-) \\
&= -\log p(D = d^+ \mid Q = q) \\
&= -\log \frac{\exp(v_q^\top v_{d^+})}{\exp(v_q^\top v_{d^+}) + \sum_{i=1}^{m} \exp(v_q^\top v_{d_i^-})}, \qquad (1)
\end{aligned}$$

where $p(D = d^+ \mid Q = q)$ can be seen as a classifier given the question q and evaluated at d^+. Normally, the `[CLS]` output of the BERT model is used as the dense representation and its default dimension is 768.

3 Compressing Dense Representations

As mentioned above, DPR's encoders produce dense vectors of 768 dimensions by default, which we will show is unnecessarily large below. In this section, we discuss how to quantify the redundancy in the encoded vectors and how to improve DPR's space efficiency by reducing this redundancy.

3.1 Quantifying Redundancy

We use two metrics to quantify the redundancy in dense vectors: *explained variance ratio* of the principal components and *mutual information* between the question vectors and passage vectors. The explained variance ratio of PCA is defined as:

$$\frac{\sum_{i=1}^{m} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2}, \qquad (2)$$

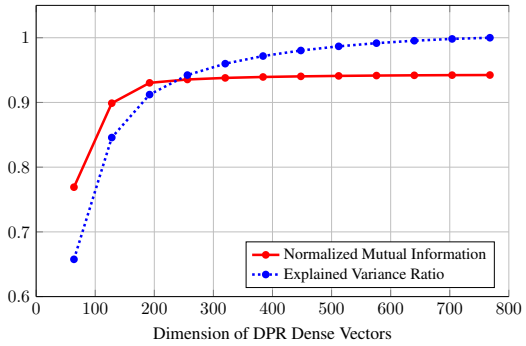

Figure 1: Quantifying redundancy of encoded vectors using explained variance ratio and normalized mutual information. Mutual information plateaus beyond roughly 200 dimensions, indicating redundancy in the remaining dimensions of DPR-768.

where σ_i^2 is the variance corresponding to the i^{th} largest eigenvalue, n is the original dimension, and m is the reduced dimension. This ratio tells us how much variance is retained by preserving the first m eigenvectors of the dense representations. Another way to evaluate redundancy is using the mutual information between the dense representation of questions Q and the retrieved passages D, which can be approximated using the classifier in Eq. (1):

$$\begin{aligned}
&\mathcal{I}(Q; D) \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{p(D|Q=q_i)}[\ln p(D \mid Q = q_i)] \\
&\quad - \mathbb{E}_{p(D)} \left[\ln \frac{1}{N} \sum_{i=1}^{N} p(D \mid Q = q_i) \right], \qquad (3)
\end{aligned}$$

where $\{q_i\}_{i=1}^N$ is the training/dev/test questions. This quantity is upper-bounded by $\ln N$ and the normalized mutual information is $\mathcal{I}(Q; D)/\ln N$.

Figure 1 shows the explained variance and mutual information of compressed vectors with different dimensions reduced by PCA. As we can see, ∼90% variance and ∼99% mutual information is held by the first ∼200 dimensions. However, as the dimension further decreases, useful information is discarded at a higher rate and the dense representation starts to degrade visibly. The figure illustrates that a dimension of ∼200 could be a sweet spot in accuracy–space trade-offs. We also find in later experiments that a dimension of 256 indeed achieves the best balance among other choices as shown in Figure 2, which will be discussed in Section 5.3.

3.2 Dense Vector Compression

We explore three different types of dimensionality reduction techniques given dense representations from a pre-trained DPR model:

Supervised Approach We apply a linear transformation to the pre-trained dense vector representations, and fine-tune this linear layer with relevance labels according to Eq. (1). Independent linear layers W_q and W_p are added to the question encoder and passage encoder, respectively. To make this compression technique a plug-and-play component, we only optimize the linear layer while freezing the rest of the networks. In addition, we add an orthogonality regularization term $||W_p W_q^\top - I||_2$ to the original loss function in Eq. (1), where I is the identity matrix. Such regularization encourages the weights in W_p and W_q to be orthogonal while retaining the most information in the original dense vectors.

Unsupervised Approach A popular technique, principle component analysis (PCA), can effectively reduce the dimensionality of high dimensional vectors while retaining most of the variance within the original representation. We fit a linear PCA transformation using the combination of all question and passage vectors to learn a compressed representation based on a pre-trained DPR model. During inference, the same transformation is applied to each question and the relevance score is the dot product between the compressed question and passage vectors.

Product Quantization On top of the supervised and unsupervised dimensionality reduction techniques described above, we further leverage product quantization (PQ), which decomposes the original d-dimensional vector into s sub-vectors. Each sub-vector is quantized using k-means and eventually stored with t bits (Jégou et al., 2011). For example, the original 768 dimension dense vector occupies 768×32 bits. By dividing it into 192 sub-vectors of 8 bits, the storage space becomes 192×8 bits, which is $1/16$ of the original size. On average, space is reduced from 32 bits to 2 bits per dimension.

4 Experimental Setup

Datasets and Metrics We evaluate the top-k retrieval accuracy of our compression methods on five QA datasets examined in the original DPR paper (Karpukhin et al., 2020): NQ, TriviaQA, WQ, CuratedTREC, and SQuAD. The top-k retrieval accuracy is defined as the fraction of questions that have at least one correct answer span in the top-k retrieved passages. Following previous work, we use $k \in \{20, 100\}$. We use the combination of NQ, TriviaQA, WQ, and CuratedTREC to train our models, following the same setting as DPR.

Model Training For DPR, instead of the original Facebook implementation, we use the implementation of Gao et al. (2021), which takes advantage of gradient caching to save GPU memory usage and mixed precision training to speed up the learning process. We find that using a learning rate of 10^{-6} and training the model for 40 epochs achieve better effectiveness than the default DPR setting. We refer to the original DPR model, which has 768 dimensional output vectors, as DPR-768. Other hyperparameters are identical to default DPR (Karpukhin et al., 2020).

For the compression methods, we consider the reduced dimensions $d \in \{256, 128, 64\}$ according to Figure 1. For the supervised approach, the linear layer is trained for one epoch with a learning rate of 10^{-3} while freezing the backbone DPR model (i.e., BERT), and we refer to these models as Linear-d. For comparison, we train models with identical architecture to Linear-d, but without freezing the BERT model, and refer to them as DPR-d.

For the unsupervised PCA approach, we fit the PCA transformation with question and passage embeddings produced by the original DPR-768 model. The question embeddings are from the original embeddings of questions in the training set. A total of 160k passage embeddings are randomly sampled from the passage embeddings of the entire corpus (i.e., original dense index).

We utilize the PQ features from Faiss (Johnson et al., 2021). The number of codewords for each sub-vector is fixed at 256 (i.e., using 8 bits). Following Izacard et al. (2020), we change the number of sub-vectors for dense embeddings such that the average memory of each dimension is reduced. Combined with dimensionality reduction methods, we decrease the occupied space of each dimension from 32 bits to 1 or 2 bits.

Sparse–Dense Hybrid Search Ma et al. (2021) have shown that hybrid search significantly improves the retrieval accuracy of DPR. Therefore, we fuse the DPR's and BM25's retrieval results following their same rule, where the final hybrid

Method	NQ		TriviaQA		WQ		CuratedTREC		SQuAD	
	top-20	top-100	top-20	top-100	top-20	top-100	top-20	top-100	top-20	top-100
			1× compression		**size: 61 GB**		**latency: 7570 ms**			
DPR-768	79.4	87.0	78.5	84.5	75.3	83.0	88.2	94.4	58.3	72.4
			3× compression		**size: 21 GB**		**latency: 2540 ms**			
PCA-256	**77.2**	**85.5**	**76.5**	**83.4**	**73.5**	**81.9**	**87.3**	**93.8**	**53.7**	**69.2**
Linear-256	76.2	85.2	75.6	83.0	71.9	81.3	86.3	93.7	51.0	67.1
DPR-256	71.8	84.1	72.1	81.2	71.8	80.6	86.6	91.8	47.2	63.3
			6× compression		**size: 11 GB**		**latency: 1130 ms**			
PCA-128	**75.3**	**84.3**	**74.9**	**82.7**	**72.4**	**81.3**	**86.3**	**93.5**	**51.4**	**67.1**
Linear-128	74.7	84.2	74.4	82.7	71.3	80.6	86.0	93.2	48.7	65.4
DPR-128	72.0	82.6	69.9	79.9	68.5	78.9	83.6	91.5	44.4	60.9
			12× compression		**size: 5.1 GB**		**latency: 625 ms**			
PCA-64	63.6	77.2	66.7	78.4	65.5	76.8	81.1	89.8	42.2	59.8
Linear-64	**69.7**	**81.2**	**69.3**	**80.0**	**66.3**	**77.8**	**83.4**	**91.8**	**42.4**	**59.9**
DPR-64	68.3	80.4	65.7	77.6	66.2	77.5	82.1	91.1	39.5	56.6
BM25	62.9	78.3	76.4	83.2	62.4	75.5	80.7	89.9	71.1	81.8

Table 1: Top-$\{20, 100\}$ retrieval accuracy of different dimensionality reduction methods evaluated at $d = \{256, 128, 64\}$ on five benchmark QA datasets. The BM25 index size is about 2.4 GB.

score is calculated by $\text{score}_{\text{dense}} + \alpha \cdot \text{score}_{\text{sparse}}$. We set $\alpha = 1$, which equally weights dense and sparse scores for all hybrid search cases as it is a good default option for hybrid search in general.

5 Results

In this section, we characterize the trade-offs between retrieval accuracy and space requirements using different combinations of our proposed techniques. All experiments are implemented using the Pyserini IR toolkit (Lin et al., 2021).

5.1 Dimensionality Reduction

Table 1 shows the top-$\{20, 100\}$ accuracy of the three dimensionality reduction techniques presented in Section 3.2 evaluated at $d = \{256, 128, 64\}$ on five benchmark QA datasets. DPR-768 achieves the best accuracy, which serves as the upper bound for compression. Overall, the top-100 accuracy decreases as we reduce the dense vectors to fewer dimensions. However, we see that PCA-256 only has a $0.6 \sim 1.5\%$ drop in accuracy on {NQ, TriviaQA, WQ, CuratedTREC} while achieving $3\times$ compression. Model quality degrades more on SQuAD, since our models are trained on the combination of the other four datasets.

We further evaluate retrieval latency for different dimensionality levels in Table 1. Although latency is not the focus of this work, it is still worth noticing that dimensionality reduction also reduces retrieval latency as it speeds up dot-product calculations. For example, reducing dimensionality from 768 to 256 can reduce retrieval latency by three times. The latency is measured by query encoding time + brute-force retrieval time using a machine with Intel Xeon Platinum 8160 2.10GHz CPU using Faiss FlatIP indexes. Both query encoding and retrieval are performed with a single CPU thread.

Across different dimensions, the unsupervised PCA method often works the best in terms of the trade-off between accuracy and compression rate. It is surprising that unsupervised PCA outperforms the other two supervised methods at dimensions 256 and 128. Although the supervised methods might be further improved with carefully-tuned hyperparameters, training DPR can be computationally expensive, while PCA is the more robust and economical method to achieve comparable results. Another surprising finding is that Linear-d generally outperforms DPR-d, which means that freezing the backbone DPR model and fine-tuning only the linear layer seem to work better than training the entire model end to end to generate compressed representations. However, this finding may be simply due to poor hyperparameter selection.

5.2 Product Quantization

Product quantization (PQ) can often aggressively reduce the index size while largely preserving retrieval effectiveness. For example, PQ2 (meaning each dimension occupies 2 bits on average after PQ) reduces the size of DPR-768's vectors by $16\times$ with only 0.7% loss in top-100 retrieval accuracy on average. Table 2 shows the top-$\{20, 100\}$ re-

Method	Comp.	Size	Latency	NQ		TriviaQA		WQ		CuratedTREC		SQuAD	
				top-20	top-100	top-20	top-100	top-20	top-100	top-20	top-100	top-20	top-100
DPR-768	1×	61 G	7570ms	79.4	87.0	78.5	84.5	75.3	83.0	88.2	94.4	58.3	72.4
+ PQ2	16×	3.8 G	2360ms	77.9	86.3	77.0	84.2	73.4	82.4	87.9	93.9	56.1	70.9
+ PQ1	32×	1.9 G	1080ms	73.7	83.5	74.7	82.7	71.1	81.0	86.3	92.8	52.4	68.4
PCA-256	3×	21 G	2540ms	77.2	85.5	76.5	83.4	73.5	81.9	87.3	93.8	53.7	69.2
+ PQ2	48×	1.3 G	765ms	74.8	84.1	74.5	82.6	72.2	81.0	88.2	92.7	51.7	67.5
+ PQ1	96×	642 M	382ms	67.6	79.4	68.0	79.2	66.5	78.7	82.7	90.1	45.1	61.4
PCA-128	6×	11 G	1130ms	75.3	84.3	74.9	82.7	72.4	81.3	86.3	93.5	51.4	67.1
+ PQ2	96×	642 M	416ms	72.3	82.9	72.2	81.7	71.0	80.1	85.3	91.8	48.5	64.9
+ PQ1	192×	321 M	266ms	62.6	76.4	64.3	77.2	64.8	77.1	80.0	89.2	41.4	58.7
Linear-64	12×	5.1 G	625ms	69.7	81.2	69.3	80.0	66.3	77.8	83.4	91.8	42.4	59.9
+ PQ2	192×	321 M	249ms	62.5	77.2	65.9	78.0	63.7	75.8	79.1	90.3	39.7	57.9
+ PQ1	384×	161 M	200ms	44.0	64.2	51.7	70.0	50.0	68.2	71.8	85.2	30.7	50.0

Table 2: Top-{20, 100} retrieval accuracy of three dimensionality reduction methods with different PQ settings at $d = \{256, 128, 64\}$.

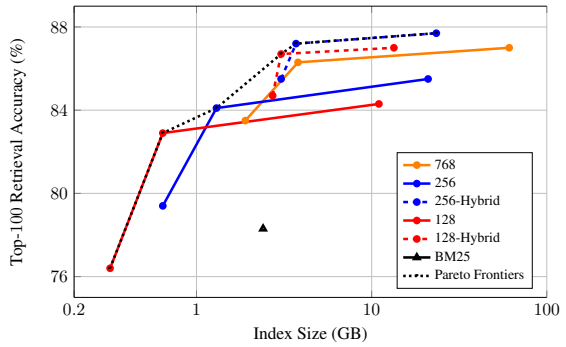

Figure 2: Trade-offs between retrieval accuracy and index size using different combination of our compression techniques.

trieval accuracy of our dimensionality reduction methods with different PQ settings. Combined with product quantization, the dimensionality reduction methods achieve a much higher compression rate with, in some cases, only a modest loss in retrieval accuracy. In addition, we find that PQ2 outperforms PQ1 on most datasets and dimensions, as PQ1 suffers more than twice the accuracy loss compared to PQ2. It appears that compression to 1 bit per dimension is too aggressive and represents a poor trade-off. If we restrict the retrieval accuracy drop to within 4% on average, we can compress the dense vectors by up to $96\times$, reducing the original DPR index from 61 GB to mere hundreds of MB on the Wikipedia corpus.

5.3 Hybrid Search

Figure 2 shows the trade-off between retrieval accuracy and index size with different combinations of dimensionality reduction, product quantization, and hybrid search. On each curve, the points from left to right represent PQ1, PQ2, and w/o PQ, respectively. Sparse retrieval with the BM25 inverted index is shown as the black triangle. The dashed lines represent sparse–dense hybrid retrieval; these lines include the size of the BM25 inverted index. In the plot, up and to the left represents better: higher accuracy and smaller indexes.

As an example, the pipeline consisting of PCA-256, PQ2, and HS reduces the (total) index size from 61 GB to 3.7 GB (57 GB or roughly $16\times$ smaller) and even yields 0.2% gain in top-100 accuracy compared to DPR-768. The dotted black line in Figure 2 shows the Pareto Frontier, which can be understood as the best achievable accuracy for a particular restriction on index size. Overall, we see that the DPR-768 (orange) line does *not* lie on the frontier, which means that some combination of our techniques is strictly more accurate and smaller than the original DPR representations.

6 Conclusions

This paper analyzes the redundancy within dense representations from DPR, a popular dense retrieval model. We propose a simple yet effective compression pipeline that enables trade-offs between space and accuracy, which drastically reduces index size while preserving end-to-end retrieval accuracy at reasonable levels.

Acknowledgements

This research was supported in part by the Canada First Research Excellence Fund and the Natural Sciences and Engineering Research Council (NSERC) of Canada; computational resources were provided by Compute Canada.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

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, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 316–321, Online. Association for Computational Linguistics.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv:2010.02666*.

Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Sebastian Riedel, and Edouard Grave. 2020. A memory efficient baseline for open domain question answering. *arXiv:2012.15156*.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:117–128.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

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

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 International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362. ACM.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv:2010.11386*.

Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy Lin. 2021. A replication study of dense passage retriever. *arXiv:2104.05740*.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

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 *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256. ACM.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized text embeddings for first-stage retrieval. *arXiv:2006.15498*.