XTR meets ColBERTv2: Adding ColBERTv2 Optimizations to XTR

Riyaz Ahmed Bhat and Jaydeep Sen
IBM Research, India
riyaz.bhat@ibm.com, jaydesen@in.ibm.com

Abstract

XTR (Lee et al., 2023) introduced an efficient multi-vector retrieval method that addresses the limitations of the ColBERT (Khattab and Zaharia, 2020) model by simplifying retrieval into a single stage through a modified learning objective. While XTR eliminates the need for multistage retrieval, it doesn't incorporate the efficiency optimizations from ColBERTv2 (Santhanam et al., 2022), which improve indexing and retrieval speed. In this work, we enhance XTR by integrating ColBERTv2's optimizations, showing that the combined approach preserves the strengths of both models. This results in a more efficient and scalable solution for multi-vector retrieval, while maintaining XTR's streamlined retrieval process. We have released the code as an addition to the PrimeQA (PrimeQA, 2023) toolkit.

1 Introduction

Retrieval refers to the task of retrieving relevant documents from a larger corpus of documents, given a search query. Retrieval is one of the most active research fields in NLP owing to its many applications such as semantic search (Fazzinga and Lukasiewicz, 2010), Open-domain Question Answering (Voorhees and Tice, 2000; Chen and Yih, 2020), Retrieval Augmented Generation (RAG) (Cai et al., 2019; Lewis et al., 2020; Guu et al., 2020). Research in Retrieval technologies has been evolving through multiple paradigms which can broadly be divided into (1) Sparse Retrievers (Robertson and Zaragoza, 2009) (2) Dense Retrievers (Karpukhin et al., 2020; Chang et al., 2019; Guu et al., 2020; Xu et al., 2022; Khattab and Zaharia, 2020; Luan et al., 2021; Santhanam et al., 2022) and very recently (3) Differential Search Index based retrievers (Tay et al., 2022). Each paradigm of retrievers has its advantages and disadvantages stemming from the methodology adopted and the limitations in those. Therefore, in practical applications, we have seen hybrid approaches that employ different kinds of retrievers to build a robust pipeline for accurate retrieval.

Sparse retrievers rely on lexical overlap to retrieve relevant documents. They largely follow bag-of-words based similarity notions to score the documents using TF-IDF score. The most popular sparse retriever called BM25 (Robertson and Zaragoza, 2009) introduces robustness in using tf-idf scores for scoring documents. Sparse retrievers often employ an inverted index for word search which is very fast and easy to maintain. Although sparse retrievers are easy to use and interpretable, their accuracy is mainly limited by the need for relevant keyword overlaps for accurate document retrieval. Dense retrievers (Karpukhin et al., 2020; Chang et al., 2019; Guu et al., 2020; Xu et al., 2022; Khattab and Zaharia, 2020; Luan et al., 2021; Santhanam et al., 2022) try to address this problem by using neural models to encode words into an embedding space. Dense retrievers compute semantic similarity in the embedding space, where two different words if semantically similar, should have their embedding vectors close to each other and hence would produce a good similarity match.

Multi-vector retrievers like ColBERT (Khattab and Zaharia, 2020) are more effective than single-vector models like DPR (Karpukhin et al., 2020) because they can capture finer semantic details between queries and documents. This makes them better suited for handling complex queries and retrieving more relevant results, while single-vector models tend to miss subtle nuances due to their limited representational capacity. However, index management for multi-vector retrievers is resource-intensive and demands specialized techniques to reduce memory footprint. ColBERTv2 (Santhanam et al., 2022) tackles this challenge by implementing strategies such as token representation compression, an aggressive residual compression mechanism, and a denoised supervision approach, which

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 358–365
January 19–24, 2025.

help reduce the memory footprint without compromising retrieval performance. Despite these optimizations, ColBERT still suffers from slow inference due to its multi-stage retrieval process, which involves token similarity computation, gathering, and reranking. To simplify this process, XTR (Lee et al., 2023) has recently proposed limiting ColBERT's retrieval to just the token similarity stage by modifying the training objective.

The optimizations proposed by ColBERTv2 and XTR for multi-vector retrieval are complementary, and integrating them into a unified system can further improve retrieval performance. In this work, we propose exactly that, and propose **ColXTR**. We adopt the XTR training objective to train **ColXTR**, with some modifications. Unlike XTR, we introduce a projection layer to reduce the dimensionality of the encoded token vectors during training, thereby minimizing both query and index space costs. After training, we apply optimizations from ColBERTv2 and adapt XTR's inference which relies on missing token imputation to further enhance both indexing and retrieval efficiency.

Our contributions are as follows:

- We develop, **ColXTR**, a multi-vector retrieval model that integrates the strengths of both ColBERTv2 and XTR.
- We empirically show that our novel compression techniques proposed on top of XTR reduce the index size by 97%, thus making it a lightweight system for practical usage.

2 Related Work

Classical IR models like BM25 (Robertson and Zaragoza, 2009) etc retrieve a ranked set of documents based on their lexical overlap with the query tokens. Due to its simplicity and strong performance on many domain-specific datasets, it is still considered as a strong baseline. With the popularity of neural language models like BERT (Devlin et al., 2018) etc, the use of such neural language models to obtain continuous representations for words (tokens) or documents has become quite popular. These neural language model based IR systems can be broadly classified into two categories a) single vector and b) multi-vector approaches.

Single vector approaches obtain a single vector representation ($v \in \mathbf{R}^d$) for the query and the documents. Usually, cosine similarity is used to compare the representation of the query and the document and obtain a similarity score. The documents are ordered based on their similarity score and the top-k similar documents are retrieved. (Karpukhin et al., 2020) used separate encoders to encode both the query and the documents. They used BM25 to obtain negative passages for the contrastive loss. In addition, the authors also used in-batch negatives during training. (Chang et al., 2019; Guu et al., 2020; Xu et al., 2022) rely on data augmentation techniques like Inverse Cloze Tasks to create data for training.

Multi-vector approaches on the other hand obtain multiple vectors per query or documents. The reasoning behind this is that a single vector representation will be unable to capture the fine interactions between the query and the document needed to retrieve the relevant document. ColBERT (Khattab and Zaharia, 2020) obtains contextualized representation from BERT for every sub-word token in the query and the document separately. They calculate the similarity of each query token representation with all the document token representations and the maximum similarity score is noted. The final similarity score is the summation of all the maximum similarity scores per token obtained in the previous step. (Luan et al., 2021) use a single vector per query but multiple vectors to represent the documents. ColBERTv2 (Santhanam et al., 2022) adopt the late interaction of ColBERT (Khattab and Zaharia, 2020). While ColBERT obtains negative documents from a model like BM25, ColBERTv2 uses ColBERT to retrieve top-k documents for a given query. The retrieved query-documents pairs are passed through a cross-encoder to obtain a similarity score. The model is trained using KL-Divergence loss to distill the cross-encoder scores. Recently, XTR (Lee et al., 2023) proposed an efficient multi-vector retrieval method that addresses the limitation of the ColBERT model. More details about the XTR model are discussed in Section 3.1.

3 System Overview

Our system is built on XTR, and we use the same notations and expressions from the original paper to provide an overview.

3.1 XTR: Training and Inference

XTR was recently proposed to improve the training and inference efficiency of multi-vector retrieval models based on Colbert architecture. Unlike single-vector retrieval models that use one dense embedding for input text and determine similarity

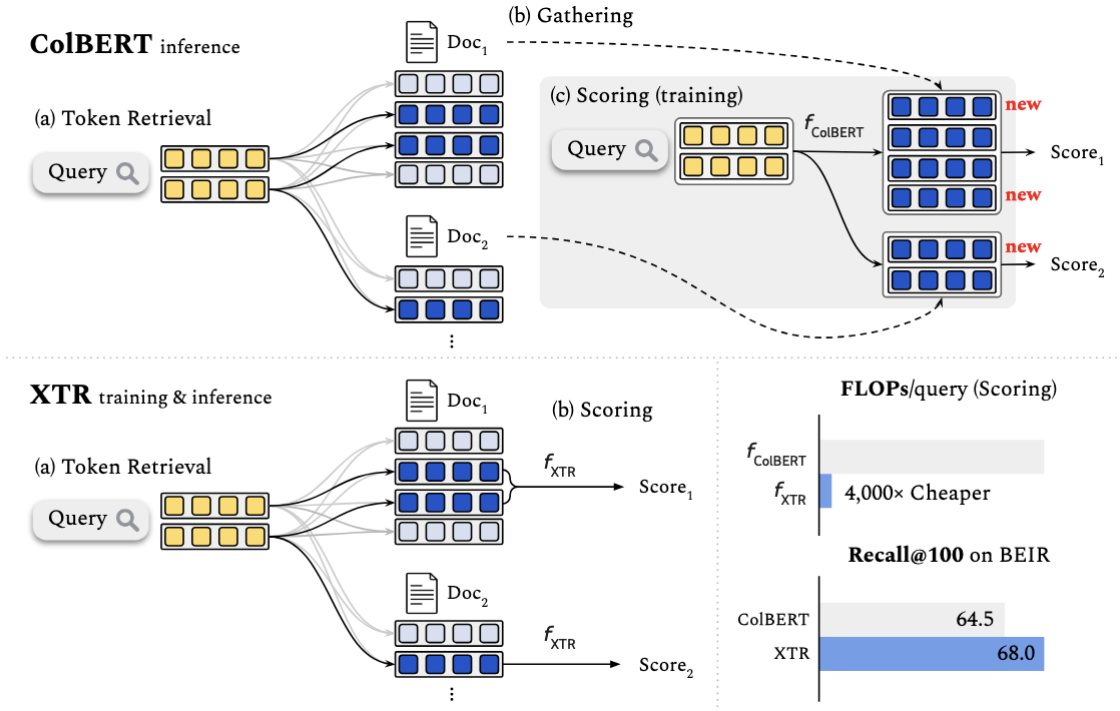

Figure 1: XTR retrieval (figure reused from original paper)

with a dot product, multi-vector retrieval models employ multiple dense embeddings for each query and document. These models usually utilize all contextualized word representations of the input text, enhancing overall model expressiveness. For a query $Q = q_{i=1}^{n}$ and a document $D = d_{j=1}^{m}$, where q_i and d_j represent d-dimensional vectors for query tokens and document tokens, multi-vector retrieval models determine the query-document similarity as follows:

$$f(Q,D) = \sum_{i}^{n} \max_{j \in |D|} q_i^T d_j = \sum_{i}^{n} \sum_{i}^{m} A_{ij} q_i^T d_j \tag{1}$$

$P_{ij} = q_i^T d_j and A \in \{0,1\}^{nxm}$ denotes the alignment matrix with A_{ij} being the token-level alignment between the query token vector q_i and the document token vector d_j. In ColBERT, sum-of-max operator sets $A_{ij} = \mathbb{1}_{[j=argmax_{j'}(P_{ij'})]}$, where the argmax is over tokens from a single document D, and $\mathbb{1}[*]$ is an indicator function.

Figure 1 demonstrates how XTR streamlines the retrieval process for multi-vector models such as ColBERT by using tokens retrieved in the initial phase to score documents directly, maintaining retrieval performance. This is accomplished by adjusting the training objective to simulate the token retrieval stage through a different alignment strategy, denoted as $\hat{A}$. Specifically, the alignment is defined as $\hat{A}_{ij} = \mathbb{1}[j \in \text{topk}_{j'}(P_{ij'})]$, where the top-k operator is applied over tokens from mini-batch documents, returning the indices of the k largest values. The modified equation is as follows:

$$f(Q,D) = \frac{1}{Z} \sum_{i}^{n} \max_{j \in |D|} \hat{A}_{ij} q_i^T d_j \tag{2}$$

Here, the normalizer Z denotes the count of query tokens that retrieve at least one document token from D. If all $\hat{A}_{ij} = 0$, Z is clipped to a small values causing $f(Q,D)$ to become 0. During training, the cross-entropy loss over in-batch negatives is used, expressed as:

$$\mathcal{L}_{CE} = -\log \frac{\exp f(Q, D^+))}{\sum_{b=1}^{B} \exp f(Q, D_b))} \tag{3}$$

Scoring Documents using Retrieved Tokens: During inference, multi-vector retrieval models first have a set of candidate documents $\hat{D}_{1:C}$ from the token retrieval stage:

$$\hat{D}_{1:C} = \hat{D} | d_j \in \hat{D} \wedge d_j \in top - k'(q*) \tag{4}$$

Here, $top-k(q*)$ represents a union of the top-k' document tokens (from the entire corpus) based on the inner product scores with each query vector. With n query token vectors, there are C ($\leq nk'$) candidate documents. Traditional methods load entire token vectors for each document and compute equation 1 for every query and candidate document pair. In contrast, XTR scores the documents solely based on retrieved token similarity. This significantly reduces computational costs during the scoring stage by eliminating redundant inner product computations and unnecessary (non-max) inner products. Moreover, the resource-intensive gathering stage, which involves loading all document token vectors for computing equation 1, is eliminated entirely.

Missing Similarity Imputation: During inference, k' document tokens are retrieved for each of the n query tokens. Assuming each document token belongs to a unique document, this results in $C = nk'$ candidate documents. In the absence of the gathering stage, there is a single token similarity to score each document. However, during training with either equation 1 or equation 2, each positive document has up to n (max) token similarities to average, which tends to converge to n as training progresses. Therefore, during inference, the missing similarity for each query token is imputed by treating each candidate document as if it were positive, with n token similarities. For every candidate document $\hat{D}$, the following scoring function is defined:

$$f(Q, \hat{D}) = \sum_{i}^{n} [\max_{j \in |D|} \hat{A}_{ij} q_i^T d_j (1 - \hat{A}_{ij}) m_i] \quad (5)$$

This is similar to equation 2, but it introduces $m_i \in \mathbb{R}$, estimating the missing similarity for each q_i. The definition of $\hat{A}$ is similar to the one in equation 2, except that it uses k' for the top-k operator. For each q_i, if $\hat{A}_{i*} = 0$ and $m_i \geq 0$, q_i considers the missing similarity m_i as the maximum value. Crucially, XTR eliminates the need to recompute any $q_i^T d_j$. When $\hat{A}_{ij} = 1$, the retrieval score from the token retrieval stage is already known, and when $\hat{A}_{ij} = 0$, there is no need to compute it as $\hat{A}_{ij} q_i^T d_j = 0$. Note that when every $\hat{A}_{ij} = 1$, the equation becomes the sum-of-max operator. Conversely, when no document tokens of $\hat{D}$ were retrieved for q_i (i.e., $\hat{A}_{i*} = 0$), the model falls back to the imputed score m_i. This provides an approximate sum-of-max result, as the missing similarity would have a score less than or equal to the score of the last retrieved token.

4 ColBERTv2: Indexing and Retrieval

XTR uses a basic MIPS library for indexing, resulting in a significant increase in space requirements. More precisely, it employs the ScaNN library (Guo et al., 2020; Sun, 2020) to store all contextualized vectors without compressing dimensionality. In contrast, our approach draws inspiration from ColBERTv2's (Santhanam et al., 2022) indexing strategy for multi-vector models, aiming to improve both space utilization and inference efficiency. ColBERTv2 achieves these enhancements by combining an aggressive residual compression mechanism with a denoised supervision strategy.

Figure 2 shows an overview of how ColBERTv2 attempts to do efficient index management with reduced storage. Contextualized vectors exhibit clustering in regions that capture highly specific token semantics, with evidence suggesting that vectors corresponding to each sense of a word cluster closely, demonstrating only minor variation due to context (Santhanam et al., 2022). Leveraging this regularity, a residual representation is introduced in ColBERTv2, significantly reducing the space requirements of late interaction models without any need for architectural or training changes. In this approach, given a set of centroids C, each vector v is encoded as the index of its closest centroid C_t and a quantized vector $\tilde{r}$ that approximates the residual $r = v - C_t$. During search, the centroid index t and residual $\tilde{r}$ are used to recover an approximate $\tilde{v} = C_t + \tilde{r}$. Each dimension of r is quantized into one or two bits to encode $\tilde{r}$.

4.1 Indexing

In the indexing stage, following ColBERTv2 we undertake a three-stage process for a given document collection, efficiently precomputing and organizing the embeddings for rapid nearest neighbor search.

- **Centroid Selection:** In the initial stage, we choose a set of cluster centroids C. These centroids serve a dual purpose in supporting both residual encoding and nearest neighbor search. To reduce memory usage, k-means clustering is applied to the embeddings produced by the T5

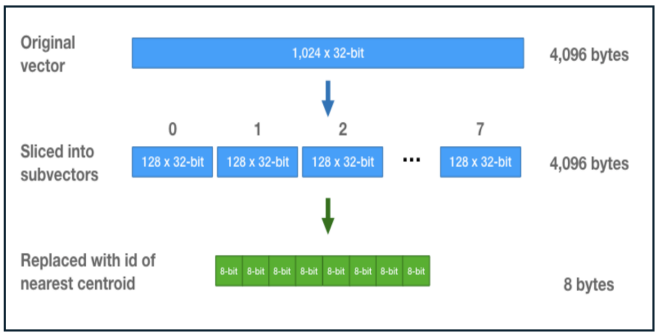

Figure 2: Overview of ColBERTv2 index optimization

encoder, considering only a sample of passages.

• **Passage Encoding:** With the centroids selected, every passage in the corpus undergoes encoding. This involves invoking the T5 encoder, compressing the output embeddings, and assigning each embedding to its nearest centroid while computing a quantized residual. The compressed representations are then saved to disk once a chunk of passages is encoded.

• **Index Inversion:** To facilitate rapid nearest neighbor search, the embedding IDs corresponding to each centroid are grouped together, and this inverted list is stored on disk. During search, this enables quick identification of token-level embeddings similar to those in a query.

4.2 Retrieval

During the retrieval phase, we use the trained **ColXTR** to encode a query, generating contextualized representations denoted as Q. Following this, we compute the inner product between these query representations and centroids ($Q^T C$), identifying the nearest centroids for each query token embedding. Leveraging the inverted list, we then identify the document token embeddings that are closer to these centroids.

Subsequently, we decompress these document token embeddings and calculate their inner product with the corresponding query vectors. The resulting similarity scores are organized based on document ids. In instances where a score for a particular query token and document token is missing, we impute it as per the equation 5. Finally, the documents are directly reranked using these similarity scores.

5 Experiments

We finetune the encoder of t5-base (Raffel et al., 2020) with XTR learning objective on MSMarco training set with a learning rate of 1-e3. XTR uses k_{train} parameter which we set to 320. We also employ a projection layer that compresses the encoded representations from 768 dimensions to 128 dimensions. The model is trained on a single A100 80GB GPU, with a batch size of 48. Moreover, we trained the model with hard negatives mined from BM25, one per positive query/document pair in a batch. The model is trained for 50K steps, and the best model based on the development set is used for the evaluation.

During retrieval, we use variable k depending on the size of the index. For smaller indexes (>1M documents), we set k to 500, while for larger ones, we increased it to 100,000. For each query token, we probed top 10 centroids.

5.1 Benchmark

We use datasets from BIER (Thakur et al., 2021) benchmark as our evaluation benchmark. BIER is a popular benchmark in IR community which is a collection of 18 datasets of varying domains as well as tasks. Because we build on top of XTR, we chose the same subset from BIER which was used for XTR benchmarking to be comparable. The datasets are as follows: (1) AR: ArguAna, (2) TO: Touché-2020, (3) FE: Fever,(4) CF: Climate-Fever, (5) SF: Scifact, (6) CV: TREC-COVID, (7) NF: NFCorpus, (8) NQ: Natural Questions, (9) HQ: HotpotQA, (10) FQ: FiQA-2018, (11) SD: SCIDOCS, (12) DB: DBPedia, (13) QU: Quora.

5.2 Evaluation Metric

We use Normalised Discounted Cumulative Gain (NDCG) as our evaluation metric, psrticularly, we report NDCG@10. As suggested in (Thakur et al.,

Datasets	AR	TO	FE	CF	SF	CV	NF	NQ	HQ	FQ	SD	DB	QU	Avg.
BM25	39.7	44.0	65.1	17.0	67.9	59.5	32.2	31.0	63.0	23.6	14.9	31.8	78.9	43.7
ColBERT	23.3	20.2	77.1	18.4	67.1	67.7	30.5	52.4	59.3	31.7	14.5	39.2	85.4	44.8
ColBERT v2	46.3	26.3	78.5	17.6	69.3	73.8	33.8	56.2	66.7	35.6	15.4	44.6	85.2	**49.9**
XTR	40.7	31.3	73.7	20.7	71.0	73.6	34.0	53.0	64.7	34.7	14.5	40.9	86.1	49.1
ColXTR	**49.3**	29.1	73.1	12.6	**71.9**	69.6	**34.3**	41.1	61.1	33.4	**15.7**	27.2	81.9	46.2

Table 1: **ColXTR** as Retriever

2021) NDCG is a robust metric to measure retrieval and reranker performance because it also considers the rank of the retrieved documents while computing the score and thus is a more informative metric than just recall.

5.3 Baselines

To compare the performance of **ColXTR**, we use several baselines, including BM25, the most popular sparse retriever. Additionally, we compare **ColXTR** with most relevant baselines such as ColBERT, ColBERTv2, the original XTR work.

5.4 Results

In this section, we review the experimental results and optimization benefits of **ColXTR** in detail.

As shown in Table 1, we see BM25, as expected, scores lower than other retrievers due to its reliance on lexical overlap. ColBERT improves upon BM25, while XTR further boosts performance over ColBERT. ColBERTv2, benefiting from distillation training, achieves the highest scores overall. Our system, **ColXTR**, performs better than ColBERT but falls short of XTR and ColBERTv2. This drop in accuracy can be attributed to the lack of hard negative mining, lack of distillation training, and the use of compressed embeddings. While ColBERTv2 is computationally expensive at inference, and XTR poses challenges in index management, **ColXTR** adopts XTR-style training with ColBERT-style compressed representations to make it more lightweight while maintaining comparable performance.

5.5 ColXTR Optimization Impact

Here we discuss the implications of the design choices and optimizations we have incorporated in **ColXTR** in making it a lightweight system, easy to deploy and manage.

In **ColXTR** we try to combine the different set of optimizations proposed in XTR and ColBERT together. We follow the XTR style retrieval with missing token imputation during inference, without the expensive gathering stage of ColBERT. As reported in XTR (Lee et al., 2023), this makes the inference 400x times faster than ColBERT.

On the other hand, XTR uses full token representations for indexing, and retaining the original size of 768, as in XTR, would result in a significantly larger memory footprint and overhead. Instead of that, we applied ColBERT like approach where we learn to compress the representation to lower dimensions and make further optimizations with residual compression. This reduces the index size by a huge margin, almost a shrink of 97%, making the index management much cheaper and easier.

Datasets	Faiss HNSW Flat Index(in GB)	ColBERT index(in GB)
NQ	860	25
NFCorpus	2.4	0.091
TREC COVID	67	3
Touché 2020	481	7

Table 2: Comparison of Faiss HNSW Flat indices and ColBERT indices in terms of size, with both using embedding dimensions of 128.

In Table 2, we give some empirical numbers to establish how the ColBERTv2 optimizations we discussed in Section 4 help in reducing the index size. Considering the first dataset, NQ, as an example, we can see it offers almost upto 97% shrinkage over the original index. On an average, we see the index size reduced by 98% across 4 datasets, which validates the need for our optimizations in designing **ColXTR** making the index management and deployment much cheaper and easier.

6 Conclusion

We have proposed **ColXTR**, an optimized multi-vector retrieval model built on top of t5-base that combines the best of both worlds: ColBERTv2 like index optimization and runtime optimizations from XTR for speedy inference. We posit this is a need of the hour for meeting industry needs for scalability with practical resource constraints. We empirically show that the lightweight training and inference pipeline for **ColXTR** provides competitive and in some cases even better performance than state-of-the-art retrieval models, while reducing the index footprint almost by 97%. We believe **ColXTR** can potentially become a default choice for using neural retrievers in industry.

References

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019. Retrieval-guided dialogue response generation via a matching-to-generation framework. 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 1866–1875, Hong Kong, China. Association for Computational Linguistics.

Wei-Cheng Chang, X Yu Felix, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2019. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*.

Danqi Chen and Wen-tau Yih. 2020. Open-domain question answering. In *Proceedings of the 58th annual meeting of the association for computational linguistics: tutorial abstracts*, pages 34–37.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Bettina Fazzinga and Thomas Lukasiewicz. 2010. Semantic search on the web. *Semantic Web*, 1(1-2):89–96.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

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.

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.

Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Y Zhao. 2023. Rethinking the role of token retrieval in multi-vector retrieval. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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.

PrimeQA. 2023. Primeqa: Toolkit for open domain qa. `https://github.com/primeqa/primeqa`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Philip Sun. 2020. Announcing scann: Efficient vector similarity search. *Google AI Blog*.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843.

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

Ellen M. Voorhees and Dawn M. Tice. 2000. The TREC-8 question answering track. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece. European Language Resources Association (ELRA).

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2022. Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3557–3569.