CoRECT: A Framework for Evaluating Embedding Compression Techniques at Scale

Laura Caspari[1[0009−0002−6670−3211]], Michael Dinzinger[1[0009−0003−1747−5643]], Kanishka Ghosh Dastidar[1[0000−0003−4171−0597]], Christofer Fellicious[1[0000−0001−7487−7110]], Jelena Mitrović[1[0000−0003−3220−8749]], and Michael Granitzer[1,2[0000−0003−3566−5507]]

[1] University of Passau, Passau, Germany
{laura.caspari, michael.dinzinger}@uni-passau.de
[2] Interdisciplinary Transformation University Austria, Linz, Austria

Abstract. Dense retrieval systems have proven to be effective across various benchmarks, but require substantial memory to store large search indices. Recent advances in embedding compression show that index sizes can be greatly reduced with minimal loss in ranking quality. However, existing studies often overlook the role of corpus complexity – a critical factor, as recent work shows that both corpus size and document length strongly affect dense retrieval performance. In this paper, we introduce CoRECT (**Co**ntrolled **R**etrieval **E**valuation of **C**ompression **T**echniques), a framework for large-scale evaluation of embedding compression methods, supported by a newly curated dataset collection. To demonstrate its utility, we benchmark eight representative types of compression methods. Notably, we show that non-learned compression achieves substantial index size reduction, even on up to 100M passages, with statistically insignificant performance loss. However, selecting the optimal compression method remains challenging, as performance varies across models. Such variability highlights the necessity of CoRECT to enable consistent comparison and informed selection of compression methods. All code, data, and results are available on GitHub[3] and HuggingFace.[4]

Keywords: Dense Retrieval · Text Embedding · Vector Quantization · Matryoshka Representation Learning · Embedding Compression.

1 Introduction

The success and wide-spread adoption of large language models (LLMs) has caused a shift in the field of information retrieval, moving from sparse, keyword-based matching to dense retrieval methods. Although effective, such systems require large amounts of memory to store embedding-based indices, which can easily surpass the size of the underlying data. To overcome memory-related scalability constraints, researchers have developed a range of embedding compression

[3] https://github.com/padas-lab-de/CoRECT
[4] https://huggingface.co/datasets/PaDaS-Lab/CoRE

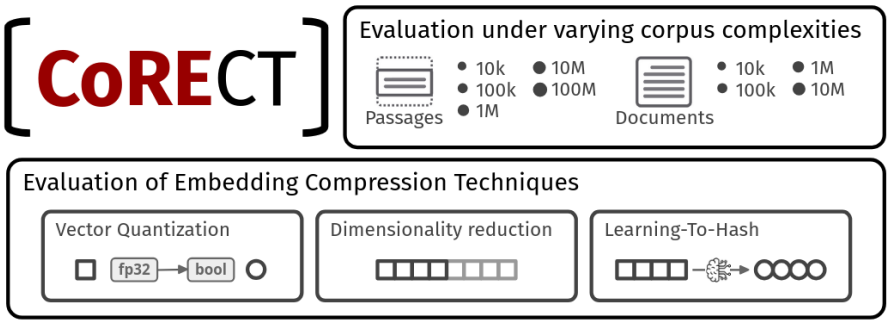

Fig. 1: The CoRECT framework.

techniques that achieve significant size reductions with little loss in retrieval quality [14,17,22,23,24,31,33,34,35,37]. While these methods show strong results on individual corpora, their systematic evaluation across a broader range of models and datasets is still lacking. In addition, they overlook how retrieval performance scales with corpus complexity, which may cause an overestimation of effectiveness on large and heterogeneous collections [19,26], particularly when vector compression limits embedding expressiveness.

In this work, we critically re-examine recent claims regarding the effectiveness of embedding compression methods, including vector quantization, dimensionality reduction, and Learning-To-Hash (LTH) techniques. We introduce **CoRECT**, a framework for systematically evaluating and comparing compression methods across diverse datasets. Our main objective is to investigate how these techniques perform under varying levels of corpus complexity – an aspect often overlooked in prior studies. To this end, we operationalize corpus complexity in two ways: (1) by scaling the corpus size from 10K to 100M passages and up to 10M documents, and (2) by comparing performance across both passage-level and document-level retrieval tasks (see Figure 1). To evaluate performance along these dimensions, we introduce a newly curated dataset collection named **Controlled Retrieval Evaluation (CoRE)**, constructed through targeted subsampling of MS MARCO v2[5] and comprising 76 human-judged TREC DL queries. CoRE serves as the primary evaluation dataset in CoRECT. To emphasize the robustness and generalizability of our findings, we further complement CoRE with datasets from BeIR [29]. Finally, to demonstrate the utility of CoRECT, we apply the framework to a diverse and representative set of embedding compression methods. As such, our main contributions are as follows:

1. The Controlled Retrieval Evaluation of Compression Techniques (CoRECT) framework, designed to systematically evaluate compression methods. To ensure robustness, we further extend our evaluation by including the domain-diverse corpora of the BeIR benchmark [29].
2. The dataset collection CoRE, the cornerstone of CoRECT, is used to assess the effect of corpus complexity on retrieval performance.

[5] `https://microsoft.github.io/msmarco/TREC-Deep-Learning.html`

3. We apply CoRECT on four different models using eight representative types of compression techniques in over 40 combinations and find that choosing the wrong compression method can significantly impact retrieval performance. Furthermore, we show that there is no single best method that fits all models.

2 Related Work

The following section reviews embedding compression methods, their evaluation, and related research on the influence of corpus complexity on performance.

Compression Methods. The compression techniques discussed below fall into three broad categories. The first group reduces the precision of embeddings by representing each dimension with a lower number of bits, in the following called Scalar and Binary Quantization (SBQ). This encompasses type-casting floating points, i.e. to float8 [21], or mapping them to discrete integer values, i.e. `uint8`. Its simplicity makes SBQ a popular compression approach supported by various libraries and vector databases, such as FAISS [8], OpenSearch[6] or Milvus[7], as well as recent embedding models through quantization-aware training [2,12].

The second group encompasses approaches for dimensionality reduction which remove certain dimensions from the embedding vector. The most common method in the context of data analysis is Principal Component Analysis (PCA), which can also be used to compress embedding vectors. More recently, Matryoshka Representation Learning (MRL) [17,35], which is integrated into model training, has found wide-spread adoption in contemporary embedding models [2,12,18,36,39].

The third and last group consists of regular hashing and Learning-to-Hash (LTH) methods. A simple baseline is Locality-Sensitive Hashing (LSH), whose numerous proposed variants and extensions make it a well-studied compression methods [3,6,7,11,13]. Regarding LTH methods, Product Quantization (PQ) [15] has emerged as a popular method with its own proposed expansions like Joint optimization of query encoding and Product Quantization (JPQ) [37] or Distill-VQ [33]. Yamada et al. [34] introduce the Binary Passage Retriever (BPR) to binarize embeddings generated by the Dense Passage Retriever [16]. Thakur et al. [28] enhance prior work by introducing domain adaptation modules, enabling methods like BPR and JPQ to perform effectively in zero-shot retrieval settings.

Evaluation of Compression Methods. Due to their popularity, SBQ methods have received growing attention in both academic studies and industry reports [14,22,23,24,31], demonstrating that high compression ratios can be achieved at minimal performance loss. Papers proposing new compression techniques tend to emphasize improvements over previous approaches [28,33,34,37], like improving over standard PQ. However, most of these evaluations are limited to a small number of datasets and methods and do not consider scalability. To the best of

[6] https://opensearch.org/
[7] https://milvus.io/

Table 1: List of tested Embedding Models; Metric on MMTEB: NDCG@10 in %

Model Name	#Params	#Dims	MRL	MMTEB
Jina V3 [27]	572M	1024	✓	55.76
Multilingual E5 (Large Instr.) [30]	560M	1024	✗	57.12
Snowflake-M V2 [36]	305M	768	✓	54.83
Snowflake-M V1 [20]	109M	768	✗	39.33

our knowledge, the only thorough study was conducted by Zhang et al. [38], who evaluate compression methods in the context of deep learning recommendation models. Additionally, they compare a small set of methods in a Retrieval Augmented Generation (RAG) setting on a single dataset, using exact match as a metric. However, due to their focus on recommendation, the chosen evaluation metrics and models are not representative of those commonly used in retrieval.

Effect of Corpus Complexity. Recent work suggests that corpus complexity can have a significant impact on downstream effectiveness [1,4,26]. Agrawal et al. [1] demonstrate that pretraining a language model on diverse and complex corpora improves their performance on downstream tasks. Reimers and Gurevych [26] show that dense retrieval models struggle as corpus size increases. However, the only compression method the authors consider is vector truncation. Finally, Chen et al. [4] examine the effect of retrieval granularity, showing that the choice of retrieval unit, i.e. passage- or document-level, has a significant impact on performance, albeit without considering embedding compression. In this paper, we focus on exactly these two aspects of corpus complexity: a) the corpus size, which we increase from 10k to 100M and b) retrieval granularity, where we compare passage and document-level retrieval.

3 Methodology

In the following sections, we introduce the CoRECT framework, including the CoRE dataset collection, and demonstrate its utility for conducting a comprehensive and comparative analysis of embedding compression methods. Table 1 lists the tested embedding models, including the number of parameters, dimensions and average performance on MMTEB [9]. While Jina V3 is MRL trained on six different cutoff levels (32, 64, 128, 256, 512 and 768), Snowflake V2 is trained only at cutoff 256.

3.1 Dataset Creation and Subsampling

The Controlled Retrieval Evaluation (CoRE) benchmark comprises two curated dataset collections – passages and documents – each derived from MS MARCO v2. Both collections are designed to enable controlled variation in the number

of non-relevant corpus items included in the retrieval task. Hence, CoRE varies along two key dimensions – corpus size and retrieval granularity – while keeping the underlying retrieval task and relevance judgments fixed. The passage collection (average length: 286 ± 111 characters) and the document collection (average length: $9\,010 \pm 15\,216$ characters) are annotated with high-quality, human-judged relevance labels from the TREC Deep Learning 2023 campaign. For passage retrieval, CoRE includes 65 queries across five corpus sizes (10K, 100K, 1M, 10M, and 100M). For document retrieval, it offers 55 queries across four corpus sizes (10K to 10M). Each query is associated with exactly 10 relevant passages or documents, ensuring comparability across corpus scales.

A central challenge in constructing smaller subsets of large corpora is preserving a high density of meaningful distractors, as naive random sampling often removes these and renders the retrieval task unrealistically easy. To address this issue, we adopt the intelligent subsampling strategy proposed by Fröbe et al. [10], mining distractor documents from pooled ranking lists submitted to the 2023 TREC Deep Learning track. To ensure quality, we discard the bottom 20% of runs based on effectiveness and apply Reciprocal Rank Fusion (RRF) [5] to merge the top-ranked documents from the remaining runs. For each query, the top 100 irrelevant or unjudged documents from this fused list are selected as high-quality distractors. To simulate a realistic and challenging evaluation scenario, each query is paired with exactly 100 mined distractors, besides the 10 relevant documents. The remainder of each corpus is filled with truly random documents that are neither relevant nor distractors under this definition.

3.2 Framework

The current implementation of the CoRECT framework evaluates compression methods on CoRE as well as the public BeIR datasets, tested with the four embedding models described above. The included compression techniques encompass eight distinct types, which will be discussed in detail in the next subsection.

Given a model and dataset, embeddings are generated in batches and stored for downstream analysis. Compression is then applied sequentially on each batch before performing retrieval using cosine similarity. Any thresholds or parameters specific to the compression method are learned per batch, with the same values applied to both query and document embeddings. For each query, CoRECT computes standard retrieval metrics such as NDCG, Recall, and MRR across multiple cutoff values k, and stores the results in JSON format.

The framework builds on widely used Python libraries, including Transformers [32] for model instantiation and the Hugging Face Hub for data loading. This modular design facilitates the integration of additional datasets, embedding models, and compression techniques. New models available through Transformers can be incorporated by extending the base interface class for model loading and embedding compression, while new retrieval datasets from Hugging Face

only require defining a custom loading function. Comprehensive instructions for extending the framework are provided in the project's README[8].

3.3 Compression Methods

In the following, we discuss the set of representative methods that have been initially included in the framework in more detail.

Floating Point Casting. A straightforward approach at embedding compression is downcasting embeddings from their native precision to a lower-precision floating-point format. We choose the native precision of each embedding model according to its reported Tensortype on Hugging Face. Using PyTorch [25], we cast embedding tensors to FP16, BF16, and FP8. As PyTorch provides multiple FP8 variants differing in mantissa and exponent length,[9] we employ two FP8 types with three- and two-bit mantissas, respectively.

Scalar and Binary Quantization (SBQ). To achieve higher compression ratios, we quantize embedding vectors to 8-, 4-, and 2-bit unsigned integer ranges using two quantization strategies. The first, Equal Distance Binning, computes the per-dimension minimum and maximum of the embeddings, divides this range into 2^x equidistant bins, and maps each floating-point value to its corresponding bin index. To reduce sensitivity to outliers, we clip values at the 2.5th and 97.5th percentiles before binning. The second method, Percentile Binning, determines bin boundaries based on percentiles so that each bin contains approximately the same number of values, thereby ensuring a more balanced quantization. For binary quantization, embeddings are mapped to 1-bit values using two thresholding schemes: zero and the per-dimension median.

Dimensionality Reduction. We employ two approaches to reduce the dimensionality of embedding vectors: truncation along Matryoshka Representation Learning (MRL) cutoff points and Principal Component Analysis (PCA). For truncation, only the first x dimensions of each embedding are retained, with x determined by the MRL training configurations of Jina V3 and Snowflake V2, enabling direct comparison with non-MRL-trained models such as E5 and Snowflake V1. Since the Snowflake models have a smaller embedding size than Jina V3 and E5 (768 vs. 1024), two distinct sets of cutoff points are defined to preserve comparable dimensionality ratios. MRL truncation is further combined with the floating-point casting and SBQ methods described above to obtain higher compression ratios. For PCA, the implementation from the FAISS library [8] is used. The sole hyperparameter – the output dimension of the compressed vector – is aligned with the cutoff values applied in MRL truncation.

[8] `https://github.com/padas-lab-de/CoRECT`
[9] `https://dev-discuss.pytorch.org/t/float8-in-pytorch-1-x/1815`

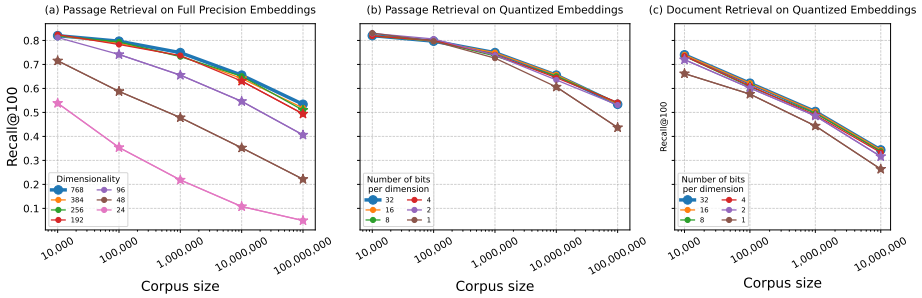

Fig. 2: Recall@100 for Snowflake V2 across increasing corpus sizes in different compression settings. Stars indicate significantly lower recall than the blue baseline (one-sided Wilcoxon test, p=0.05). Recall decreases with higher corpus size and granularity. Quantization (middle) outperforms vector truncation (left).

Hashing and Learning-to-Hash. The final group of methods includes Locality-Sensitive Hashing (LSH) and Product Quantization (PQ), both implemented using FAISS [8]. LSH compresses embedding vectors into binary codes, controlled by a hyperparameter that defines the bit length of the resulting code. Four configurations are evaluated, corresponding to compression ratios of 4×, 8×, 16×, and 32×. PQ, as implemented in FAISS, applies k-means clustering with L2 distance to partition each embedding into subvectors, encoding each subspace with a specified number of bits. Six combinations of subvector count and code size are compared. To maintain consistent compression ratios across models, two distinct hyperparameter sets are defined – one for 768-dimensional and one for 1024-dimensional embeddings.

4 Results

This section demonstrates the utility of the CoRECT framework by showcasing the analyses it enables. Using CoRECT, we systematically compare the embedding compression methods introduced in Section 3.3 and examine how their performance varies across models, datasets, corpus sizes, and retrieval granularities. These experiments highlight the framework's ability to support controlled, large-scale evaluation of compression techniques – insights that would be difficult to obtain without it. Comprehensive results for all models, datasets, and methods are available on our project page[10]

4.1 Impact of Corpus Complexity

Corpus Complexity Significantly Impacts Retrieval Performance. An important asset of our framework lies in its ability to evaluate the robustness of compression methods with respect to increasing corpus complexity. Figure 2

[10] `https://padas-lab-de.github.io/CoRECT/`

Table 2: Recall@100 (in %) with increasing corpus size. The best result per column is marked in bold, the second best is underlined. The degree of performance degradation when increasing corpus complexity varies between models, making some more suitable for application on large corpora than others.

	Passages			Documents		
	10k	1M	100M	10k	1M	10M
Jina V3	<u>83.85</u>	71.39	44.15	72.91	47.64	26.91
E5	**84.15**	**75.39**	50.77	**76.00**	47.64	31.64
Snowflake V2	82.00	<u>74.92</u>	**53.39**	<u>74.00</u>	<u>50.18</u>	<u>34.36</u>
Snowflake V1	80.31	71.69	<u>50.92</u>	<u>74.00</u>	**50.36**	**36.73**

shows Recall@100 for Snowflake V2 on CoRE, measuring the model's ability to retrieve relevant documents among the top 100 results as a first-stage retriever. It consists of three line charts: (a) and (b) for passage retrieval, and (c) for document retrieval. In Plot (a), embedding vectors are only truncated at increasingly lower dimensions, while in Plot (b), only Percentile Binning is applied. Plot (c) mirrors the structure of (b) for document retrieval. The blue line, representing the uncompressed baseline, serves as a reference in the three charts. Across all settings, the retrieval performance decreases significantly as the corpus grows, even for the uncompressed baseline. This trend reflects the increasing difficulty of the retrieval task with larger passage or document collections.

Scalability of Vector Truncation Depends on Compression Ratio. Plot (a) isolates the impact of vector truncation on performance, reaching a maximum compression ratio of 32 when only the first 24 dimensions of each vector are retained. While moderate truncation to 384 or 256 dimensions barely affects the recall scores, significant performance differences, albeit small, start to manifest at 192 dimensions (4× compression) for the larger corpora. At a compression ratio of 8, i.e. 96 dimensions, recall starts to drop sharply, even for the 100k corpus. While, here, we only present the results for Snowflake V2, as it is one of the MRL trained models, we observed even better results for Jina V3, as the vector can be truncated to 256 dimensions (4× compression) without significant performance differences. Furthermore, truncating to a compression ratio of 8 only leads to a slight decrease in performance. Naturally, for Snowflake V1 and E5, the two models without any MRL training, truncation performs worse, quickly leading to significant degradations of recall as corpus size increases. Notably, for both models taking only half of the vector dimensions already has a significant performance impact when the corpus size increases to 1M.

Quantization Outperforms Vector Truncation. Plot (b) reveals that scalar quantization barely impacts performance, while binarizing the vector has a significant impact on recall performance at larger corpus sizes. Comparing the performance of quantization to that of vector truncation in plot (a) at the same compression ratio, we find that quantization has a clear edge. Looking at the remaining models, whose results are not shown in the plot, we observe an

Table 3: Recall@100 (in %) for different corpus complexity with full and binary precision (no truncation). Choosing the wrong thresholding type can greatly decrease performance.

	Jina V3		E5		Snowflake V2		Snowflake V1	
	10k	100M	10k	100M	10k	100M	10k	100M
Full precision	83.85	44.15	84.31	50.92	82.00	53.39	80.31	50.92
Zero Thresh.	**83.39**	**41.39**	71.23	19.23	**82.62**	**49.39**	**78.92**	**32.46**
Median Thresh.	82.92	35.69	**86.31**	**39.54**	**82.62**	43.39	70.77	18.00

even clearer advantage for E5, where quantization clearly outperforms truncation. For Jina V3, the performance difference between quantization and vector truncation becomes smaller due to the model's ability to retain high recall even with large vector truncation. For both E5 and Jina V3 the quantized vectors retain most of the original's performance. In contrast, quantizing Snowflake V1 embeddings causes a large degradation in recall, even though SBQ still performs slightly better than truncation. Considering retrieval granularity, plot (c) shows similar trends to plot (b) for document retrieval, albeit with faster overall performance degradation. This pattern can also be observed for the remaining models, where quantization performance trends on passage level translate to document retrieval.

Model Robustness to Corpus Complexity Varies. Apart from allowing us to evaluate the robustness of compression methods with respect to corpus size and retrieval granularity, CoRE also enables us to study a model's general capability of dealing with these challenges. Interestingly, we can observe different degrees of performance degradation depending on the model. As shown in Table 2, while E5 performs best on the small 10k passage corpus followed by Jina V3, on the largest 100M corpus Snowflake V2 takes the lead with E5 and Jina V3 in the last two places. For document retrieval, E5 again takes the lead for 10K, while for the larger corpora Snowflake V1, the smallest model, demonstrates the best recall and Jina V3, the largest one, the worst. As the embedding models demonstrate different capability of retaining recall with increasing corpus complexity, we demonstrate the general importance of evaluating model performance on large-scale datasets.

4.2 Compression Robustness Across Datasets and Models

Choosing the Wrong Compression Method Harms Performance. When comparing the performance of methods with an equal compression ratio across models, we find that specific model-compression method combinations significantly reduce retrieval performance. Table 3 presents Recall@100 for each model, comparing full-precision baselines with two binary quantization methods: zero and median thresholding. Interestingly, zero thresholding outperforms median thresholding for three of the four models, particularly on the 100M corpus. The

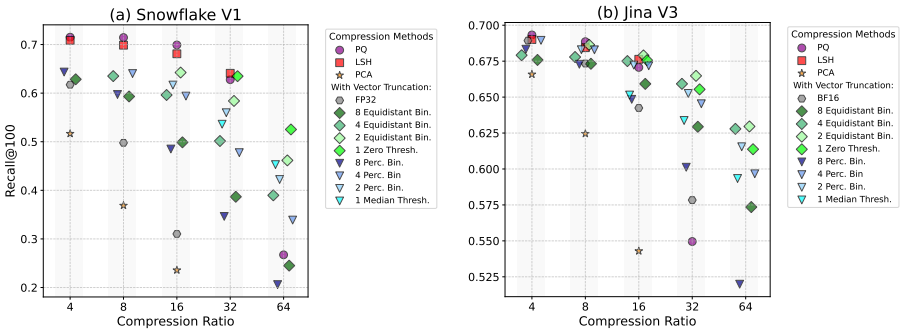

Fig. 3: Average Recall@100 across BeIR datasets for different compression methods. The x-values of some compression methods have been shifted slightly for better visibility. The light gray columns in the background mark the compression ratio to which the methods belong. Shapes indicate different types of compression techniques. Some methods have been combined with vector truncation to achieve the desired compression ratio as indicated in the legend. The top-performing measures vary depending on the model.

larger discrepancy at 100M is most likely a result of the median being estimated on a limited sample rather than the full dataset. Despite its simplicity, zero thresholding achieves remarkably strong performance, with E5 being the only model where this approach leads to a notable drop in recall. In this paper, we restrict ourselves to reporting these empirical observations; a deeper interpretation of the underlying causes is left for future work.

Large Compression Ratios Show Differences of Methods. Expanding our focus to evaluating the robustness of compression methods on a diverse set of corpora, we apply CoRECT on the BeIR datasets. Although the ranking of compression methods changes between datasets, these variations are largely attributable to minor performance differences among densely clustered methods. Figure 3 presents average Recall@100 across BeIR for different compression ratios. For Jina V3 (Plot b), the methods cluster closely at low compression rates, but clear performance differences emerge as the compression ratio increases. Quantization to 2 bits performs robustly across all ratios—a trend also observed for Snowflake V2 and E5. Within the SBQ family, combining moderate vector truncation with quantization yields high compression ratios while often improving retrieval performance compared to heavier quantization of the full vector. This approach works particularly well with Jina's MRL-training and, to a lesser extent, for Snowflake V2 and E5 at smaller truncation levels. An exception is the Snowflake V1 model, whose performance degrades sharply as compression increases (see Plot a). Although LSH and PQ perform relatively well at lower ratios, their effectiveness drops substantially at 32× compression. Across most settings, however, PQ and LSH still outperform scalar quantization, confirming their advantage at moderate compression levels.

Type of Retrieval Task Affects Compression Performance. So far, we only considered compression in the context of first-stage retrieval, using Recall@100 as a performance metric. However, when using retrieval without reranking, NDCG@10 becomes a more suitable metric, as the number of retrieved documents will be lower. Figure 4 shows that in contrast to the obvious performance degradation of Recall@100 in Figure 2, NDCG@10 remains far more stable with increasing corpus size. While the general observations detailed above also transfer to NDCG@10, Figure 4 shows that moderate vector truncation leads to almost no performance loss, while truncation to 192 dimensions only leads to a slight decrease on the 100M dataset. As such, when considering only the top-10 results, stronger compression is possible without significantly reducing retrieval performance. This demonstrates that the primary metric for evaluating compression methods should be chosen according to the retrieval context.

Overall, our results demonstrate that there is no single best compression method that fits all models. Rather, the optimal choice is model-dependent, necessitating a thorough evaluation and comparison of compression methods. Thus, comparing compression methods across diverse corpora allows us to a) identify methods that consistently perform well and b) exclude methods that are unsuitable for a particular model.

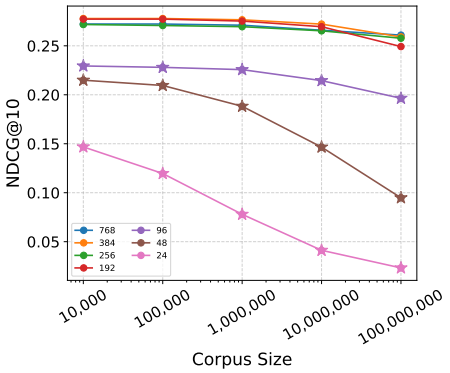

Fig. 4: NDCG@10 for Snowflake V2 on the CoRE passage corpora. In contrast to Recall@100, NDCG@10 remains stable with increasing corpus size.

5 Discussion and Future Work

Overall, the insights generated with the CoRECT framework enable us to derive certain guidelines for applying compression techniques in practice:

1. **Evaluate model performance on large datasets.** When choosing a model to generate millions of embeddings, it is essential to evaluate its performance on large corpora. As our analysis shows, some models' retrieval performance degrades quicker when increasing corpus size, which can lead to a significant decrease in performance when choosing the wrong model.
2. **Compare Compression Performance of Different Methods.** When choosing a compression method, comparing its performance to a diverse set of other methods at a fixed compression ratio is important, as our analysis has demonstrated that a method that performs well for one model does not necessarily do so for another. Selecting a method that is unsuitable for a specific model can greatly impact performance.
3. **Try Combining Compression Techniques.** For three out of our four models, combining quantization with moderate vector truncation achieved

the best retrieval performance at a fixed compression ratio. While the number of method combinations we considered is fairly limited, the results still highlight the potential benefit of mixing different methods.

While our framework aims to compare a wide variety of compression methods, a clear limitation lies in its current focus on SBQ along with techniques for dimensionality reduction, ignoring more recent LTH approaches described in Section 2. To demonstrate the initial capabilities of our framework, we focused on popular compression techniques that are not as train-intensive as recent LTH methods. However, we acknowledge that integrating such methods in the future would be a valuable extension of CoRECT, allowing the application and evaluation of these methods on modern embedding models. Furthermore, we only evaluate compression techniques based on their performance, ignoring other factors like their storage efficiency, computation time and retrieval speed.

6 Conclusion

We introduced CoRECT, a framework designed to robustly evaluate the performance of embedding compression methods with respect to (1) corpus complexity and (2) consistency across diverse corpora. To study the first aspect, we developed CoRE, a benchmark that enables controlled variation in the number of non-relevant passages or documents included in the retrieval task. To address the second aspect, we leverage the diverse BeIR datasets.

Applying CoRECT to a representative set of compression techniques – scalar and binary quantization, dimensionality reduction, locality-sensitive hashing, and product quantization, among others – across four different retrieval models reveals that there is no "one-size-fits-all" compression method that performs uniformly well across models and datasets. Crucially, these findings emerge only through CoRECT's systematic and controlled evaluation design, which is explicitly tailored to disentangle the effects of corpus complexity and model-specific behavior. Through this framework, we uncover model-dependent pitfalls, where certain compression techniques significantly degrade performance for one model while remaining effective for others.

Using CoRE, we further show that retrieval performance generally declines with increasing corpus complexity. While this trend is expected, the degree of degradation varies substantially across models, suggesting that some are inherently more robust to scaling than others. Overall, CoRECT demonstrates the necessity of model-specific, controlled evaluation when applying embedding compression techniques, enabling a more nuanced and reliable understanding of their real-world retrieval performance.

Acknowledgments

This work has received funding from the European Union's Horizon Europe research and innovation program under grant agreement No. 101070014 (OpenWebSearch.EU).

References

1. Agrawal, A., Singh, S.: Corpus complexity matters in pretraining language models. In: Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP). pp. 257–263 (2023)
2. AI, V.: voyage-3-large: the new state-of-the-art general-purpose embedding model. Voyage AI Blog (2025), https://blog.voyageai.com/2025/01/07/voyage-3-large/
3. Bawa, M., Condie, T., Ganesan, P.: Lsh forest: self-tuning indexes for similarity search. In: Proceedings of the 14th International Conference on World Wide Web. p. 651–660. WWW '05, Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1060745.1060840, https://doi.org/10.1145/1060745.1060840
4. Chen, T., Wang, H., Chen, S., Yu, W., Ma, K., Zhao, X., Zhang, H., Yu, D.: Dense X retrieval: What retrieval granularity should we use? In: Al-Onaizan, Y., Bansal, M., Chen, Y.N. (eds.) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 15159–15177. Association for Computational Linguistics, Miami, Florida, USA (Nov 2024). https://doi.org/10.18653/v1/2024.emnlp-main.845, https://aclanthology.org/2024.emnlp-main.845/
5. Cormack, G.V., Clarke, C.L.A., Buettcher, S.: Reciprocal Rank Fusion outperforms Condorcet and Individual Rank Learning Methods. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 758–759. SIGIR '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1571941.1572114, https://doi.org/10.1145/1571941.1572114
6. Dasgupta, A., Kumar, R., Sarlos, T.: Fast locality-sensitive hashing. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1073–1081. KDD '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2020408.2020578, https://doi.org/10.1145/2020408.2020578
7. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry. p. 253–262. SCG '04, Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/997817.997857, https://doi.org/10.1145/997817.997857
8. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.E., Lomeli, M., Hosseini, L., Jégou, H.: The faiss library (2024)
9. Enevoldsen, K., Chung, I., Kerboua, I., Kardos, M., Mathur, A., Stap, D., Gala, J., Siblini, W., Krzemiński, D., Winata, G.I., Sturua, S., Utpala, S., Ciancone, M., Schaeffer, M., Sequeira, G., Misra, D., Dhakal, S., Rystrøm, J., Solomatin, R., Ömer Çağatan, Kundu, A., Bernstorff, M., Xiao, S., Sukhlecha, A., Pahwa, B., Poświata, R., GV, K.K., Ashraf, S., Auras, D., Plüster, B., Harries, J.P., Magne, L., Mohr, I., Hendriksen, M., Zhu, D., Gisserot-Boukhlef, H., Aarsen, T., Kostkan, J., Wojtasik, K., Lee, T., Šuppa, M., Zhang, C., Rocca, R., Hamdy, M., Michail, A., Yang, J., Faysse, M., Vatolin, A., Thakur, N., Dey, M., Vasani, D., Chitale, P., Tedeschi, S., Tai, N., Snegirev, A., Günther, M., Xia, M., Shi, W., Lù, X.H., Clive, J., Krishnakumar, G., Maksimova, A., Wehrli, S., Tikhonova, M., Panchal, H., Abramov, A., Ostendorff, M., Liu, Z., Clematide, S., Miranda, L.J., Fenogenova, A., Song, G., Safi, R.B., Li, W.D., Borghini, A., Cassano, F., Su, H., Lin, J., Yen, H., Hansen, L., Hooker, S., Xiao, C., Adlakha, V., Weller, O., Reddy, S., Muennighoff, N.: Mmteb: Massive multilingual text embedding benchmark (2025), https://arxiv.org/abs/2502.13595

10. Fröbe, M., Parry, A., Scells, H., Wang, S., Zhuang, S., Zuccon, G., Potthast, M., Hagen, M.: Corpus subsampling: Estimating the effectiveness of neural retrieval models on large corpora. In: Hauff, C., Macdonald, C., Jannach, D., Kazai, G., Nardini, F.M., Pinelli, F., Silvestri, F., Tonellotto, N. (eds.) Advances in Information Retrieval. pp. 453–471. Springer Nature Switzerland, Cham (2025)
11. Gan, J., Feng, J., Fang, Q., Ng, W.: Locality-sensitive hashing scheme based on dynamic collision counting. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. p. 541–552. SIGMOD '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2213836.2213898, https://doi.org/10.1145/2213836.2213898
12. Günther, M., Sturua, S., Akram, M.K., Mohr, I., Ungureanu, A., Eslami, S., Martens, S., Wang, B., Wang, N., Xiao, H.: jina-embeddings-v4: Universal embeddings for multimodal multilingual retrieval (2025), https://arxiv.org/abs/2506.18902
13. Huang, Q., Feng, J., Zhang, Y., Fang, Q., Ng, W.: Query-aware locality-sensitive hashing for approximate nearest neighbor search. Proc. VLDB Endow. **9**(1), 1–12 (Sep 2015). https://doi.org/10.14778/2850469.2850470, https://doi.org/10.14778/2850469.2850470
14. Jina: Binary embeddings: All the ai, 3.125% of the fat, https://jina.ai/news/binary-embeddings-all-the-ai-3125-of-the-fat/
15. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(1), 117–128 (2011). https://doi.org/10.1109/TPAMI.2010.57
16. Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., tau Yih, W.: Dense passage retrieval for open-domain question answering (2020), https://arxiv.org/abs/2004.04906
17. Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., Farhadi, A.: Matryoshka representation learning. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 30233–30249. Curran Associates, Inc. (2022)
18. Lee, J., Chen, F., Dua, S., Cer, D., Shanbhogue, M., Naim, I., Ábrego, G.H., Li, Z., Chen, K., Vera, H.S., Ren, X., Zhang, S., Salz, D., Boratko, M., Han, J., Chen, B., Huang, S., Rao, V., Suganthan, P., Han, F., Doumanoglou, A., Gupta, N., Moiseev, F., Yip, C., Jain, A., Baumgartner, S., Shahi, S., Gomez, F.P., Mariserla, S., Choi, M., Shah, P., Goenka, S., Chen, K., Xia, Y., Chen, K., Duddu, S.M.K., Chen, Y., Walker, T., Zhou, W., Ghiya, R., Gleicher, Z., Gill, K., Dong, Z., Seyedhosseini, M., Sung, Y., Hoffmann, R., Duerig, T.: Gemini embedding: Generalizable embeddings from gemini (2025), https://arxiv.org/abs/2503.07891
19. Luan, Y., Eisenstein, J., Toutanova, K., Collins, M.: Sparse, dense, and attentional representations for text retrieval. Transactions of the Association for Computational Linguistics **9**, 329–345 (2021). https://doi.org/10.1162/tacl_a_00369, https://aclanthology.org/2021.tacl-1.20/
20. Merrick, L., Xu, D., Nuti, G., Campos, D.: Arctic-embed: Scalable, efficient, and accurate text embedding models (2024), https://arxiv.org/abs/2405.05374
21. Micikevicius, P., Stosic, D., Burgess, N., Cornea, M., Dubey, P., Grisenthwaite, R., Ha, S., Heinecke, A., Judd, P., Kamalu, J., Mellempudi, N., Oberman, S., Shoeybi, M., Siu, M., Wu, H.: Fp8 formats for deep learning (2022), https://arxiv.org/abs/2209.05433

22. Microsoft: Azure ai search october updates: Nearly 100x compression with minimal quality loss, https://techcommunity.microsoft.com/blog/azure-ai-services-blog/azure-ai-search-october-updates-nearly-100x-compression-with-minimal-quality-los/4265447
23. Mixedbread: 64 bytes per embedding, yee-haw, https://www.mixedbread.com/blog/binary-mrl
24. MongoDB: Binary quantization & rescoring: 96% less memory, faster search, https://www.mongodb.com/blog/post/binary-quantization-rescoring-96-less-memory-faster-search
25. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)
26. Reimers, N., Gurevych, I.: The curse of dense low-dimensional information retrieval for large index sizes. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) 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). pp. 605–611. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-short.77, https://aclanthology.org/2021.acl-short.77/
27. Sturua, S., Mohr, I., Akram, M.K., Günther, M., Wang, B., Krimmel, M., Wang, F., Mastrapas, G., Koukounas, A., Koukounas, A., Wang, N., Xiao, H.: jina-embeddings-v3: Multilingual embeddings with task lora (2024), https://arxiv.org/abs/2409.10173
28. Thakur, N., Reimers, N., Lin, J.: Injecting domain adaptation with learning-to-hash for effective and efficient zero-shot dense retrieval (2022). https://doi.org/10.48550/ARXIV.2205.11498, https://arxiv.org/abs/2205.11498
29. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I.: BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models (2021). https://doi.org/10.48550/ARXIV.2104.08663
30. Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., Wei, F.: Multilingual e5 text embeddings: A technical report. arXiv preprint arXiv:2402.05672 (2024)
31. Weaviate: 32x reduced memory usage with binary quantization, https://weaviate.io/blog/binary-quantization
32. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020), https://www.aclweb.org/anthology/2020.emnlp-demos.6
33. Xiao, S., Liu, Z., Han, W., Zhang, J., Lian, D., Gong, Y., Chen, Q., Yang, F., Sun, H., Shao, Y., Xie, X.: Distill-vq: Learning retrieval oriented vector quantization by distilling knowledge from dense embeddings. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 1513–1523. SIGIR '22, Association for Computing Machin-

ery, New York, NY, USA (2022). https://doi.org/10.1145/3477495.3531799, https://doi.org/10.1145/3477495.3531799
34. Yamada, I., Asai, A., Hajishirzi, H.: Efficient passage retrieval with hashing for open-domain question answering. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) 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). pp. 979–986. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-short.123, https://aclanthology.org/2021.acl-short.123/
35. Yoon, J., Sinha, R., Arik, S.O., Pfister, T.: Matryoshka-adaptor: Unsupervised and supervised tuning for smaller embedding dimensions. In: Al-Onaizan, Y., Bansal, M., Chen, Y.N. (eds.) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 10318–10336. Association for Computational Linguistics, Miami, Florida, USA (Nov 2024). https://doi.org/10.18653/v1/2024.emnlp-main.576, https://aclanthology.org/2024.emnlp-main.576/
36. Yu, P., Merrick, L., Nuti, G., Campos, D.: Arctic-embed 2.0: Multilingual retrieval without compromise (2024), https://arxiv.org/abs/2412.04506
37. Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Jointly optimizing query encoder and product quantization to improve retrieval performance. p. 2487–2496. CIKM '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3459637.3482358, https://doi.org/10.1145/3459637.3482358
38. Zhang, H., Zhao, P., Miao, X., Shao, Y., Liu, Z., Yang, T., Cui, B.: Experimental analysis of large-scale learnable vector storage compression. Proc. VLDB Endow. **17**(4), 808–822 (Dec 2023). https://doi.org/10.14778/3636218.3636234, https://doi.org/10.14778/3636218.3636234
39. Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., Zhou, J.: Qwen3 embedding: Advancing text embedding and reranking through foundation models. arXiv preprint arXiv:2506.05176 (2025)