Quantization Beyond Uniform Bit Allocation

K. S. Sreeramji*
Indian Institute of Science
Bengaluru, India
sreeramjiks@iisc.ac.in

Sabyasachi Basu
Microsoft Research
Bengaluru, India
sabyasachi.basu@microsoft.com

Ravishankar Krishnaswamy
Microsoft Research
Bengaluru, India
rakri@microsoft.com

Kirankumar Shiragur
Microsoft Research
Bengaluru, India
kshiragur@microsoft.com

Yujia Wang
Microsoft STCA
Suzhou, China
yujiawang@microsoft.com

ABSTRACT

Quantization is a fundamental technique to handle the growing sizes of embeddings generated by modern models. Existing quantization schemes are largely embedding agnostic and allocate bits uniformly across dimensions. However, recent models produce embeddings with significant geometric structure. In this work, we investigate whether a variable bit allocation scheme can improve quantization quality under a fixed memory budget. We propose a simple variable bit allocation framework that partitions an embedding into contiguous buckets and allocates storage non-uniformly across them. Using a greedy allocation strategy, we instantiate this framework for both Product Quantization (PQ) and Scalar Quantization (SQ). We perform a series of experiments on embeddings known to have the Matryoshka property (MRL), and consistently observe that non-uniform allocations outperform uniform baselines at identical storage budgets. The largest improvements occur in the low-bit regime, where uniform allocation is particularly inefficient for MRL embeddings. At the same compression rates, variable allocation improves recall by up to 8% for PQ and up to 18% for SQ. Our results suggest a new direction for structure-aware compression and indexing techniques for large-scale retrieval systems.

VLDB Workshop Reference Format:
K. S. Sreeramji, Sabyasachi Basu, Ravishankar Krishnaswamy, Kirankumar Shiragur, and Yujia Wang. Quantization Beyond Uniform Bit Allocation. VLDB 2026 Workshop: 2nd Workshop on Vector Databases (VecDB).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/jam1729/DiskANN/tree/variable_quantization.

1 INTRODUCTION

Nearest neighbor search is a crucial component of nearly all forms of information retrieval in large ML systems that use embeddings: from search [16, 22], to recommendations [8]. However, due to the curse of dimensionality, nearest neighbor search becomes infeasible [18, 30]; even the more popular approximate versions of nearest neighbor search used in practice [9, 14, 24, 33, 34] face challenges related to scalability [11]. Modern embeddings routinely have thousands of dimensions; a single 3000 dimensional vector has a disk footprint of 12 kilobytes, implying a total footprint of several terabytes for billion scale datasets. Quantization provides some reprieve in these situations: these techniques aim to reduce the storage and memory footprints by replacing each full vector with a quantized code that approximates the geometric structure of the dataset [9, 20]. Reducing the data type's footprint from a 32 bit float representation to a custom 1 bit binary representation reduces the footprint by a factor of 32, albeit at the cost of some loss in embedding quality.

Several quantization schemes have shown excellent performance on real-world data and are routinely used in large scale industrial systems. Of these, Product Quantization (PQ) [20, 25] and Scalar Quantization (SQ) [1, 9, 40] are perhaps the most well-known and widely used.

Methods such as this stand out for not just their impressive performance, but also their relative conceptual simplicity.

Furthermore, many modern embeddings [2, 7, 28, 29, 31, 37, 38, 42] are known to exhibit the *Matryoshka property* (MRL) [21], where the leading dimensions capture a disproportionate fraction of the structure of the embeddings. This implies that even if the full vector itself is quite large, truncating it to a fraction of its length and retaining only the first few dimensions can preserve the vector's quality sufficiently well for most downstream tasks, including retrieval. In fact, prior work observes that across many real-world datasets, retaining just the top 15% of dimensions achieves over 70% recall from modern embeddings that have the MRL property (Figure 4 of [21]).

So far, post training quantization schemes have been largely embedding agnostic. We ask the following question at this juncture:

> Can a variable bit allocation scheme allow for better quantization of vectors at the same compression rate?

Specifically, in the setting of MRL embeddings, this manifests as an interesting hypothesis: given the outsized importance of the initial dimensions, a strategy that assigns more bits to the initial dimensions may outperform existing techniques that provide uniform (or amortized uniform) bit budgets to all dimensions.

*Work done during internship at Microsoft Research.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

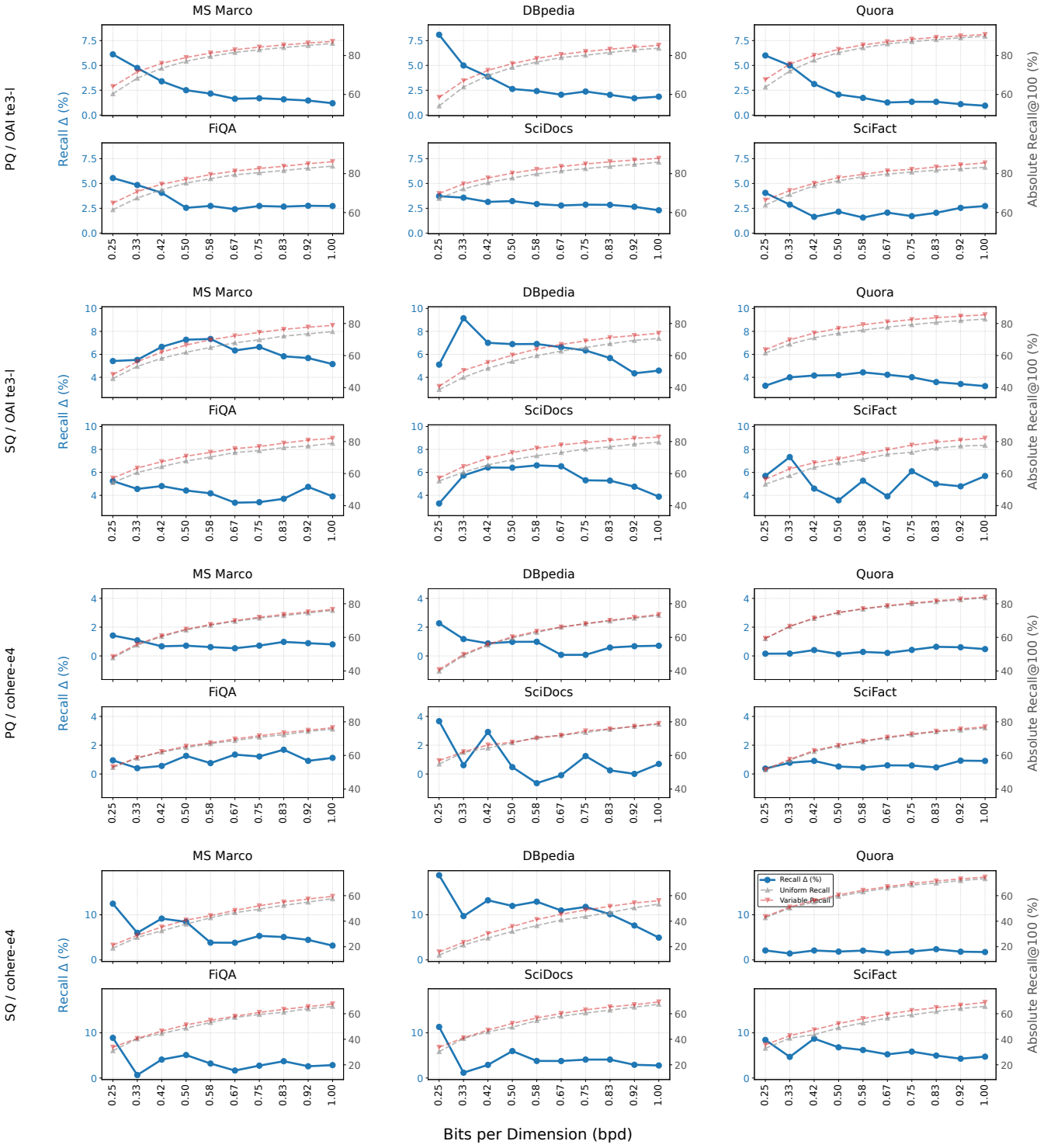

Figure 1: A comprehensive account of the gains in recall across different datasets. We compare gains on embeddings generated using OpenAI's `text-embedding-3-large` and Cohere's `embed-v4` models, using both Product Quantization (PQ) and Scalar Quantization (SQ). In addition to the relative recall improvement, we also present the absolute 100-recall@100 achieved by the uniform and variable allocation schemes.

This paper affirms this hypothesis. We show that variable allocation strategies allow us to beat standard uniform allocation quantization schemes at the same bit budgets. All embeddings used in our experiments are known to have the MRL property (refer to Figure 2). While our allocation scheme does not explicitly make use of the Matryoshka property, the "optimized" bit allocation scheme strongly reflects the property, with more bits being assigned to the leading dimensions. Specifically, at low bit budgets, we observe that aggressively assigning more bits to the leading dimensions leads to noticeable gains over uniform allocation for the same quantization techniques. We evaluate Scalar and Product Quantization (hereafter referred to as SQ and PQ respectively) using a greedy bit allocation scheme with fixed buckets that group consecutive dimensions. Under this scheme, the algorithm is granted a bit budget in fixed increments, and assigns the additional bits to the bucket which gives the greatest improvement. Under this scheme, we see that there is up to an 8% relative improvement over uniform for PQ and over 18% for SQ when measuring 100-recall@100. The gains are asymmetric and both model and dataset dependent; however, even if margins are thin, there are consistent gains. An important feature of this approach is the improvement in the sub-1 bit regime. Here, uniform bit allocation is demonstrably suboptimal for vectors with the MRL property (refer to Figure 1). Unsurprisingly, our biggest wins are consistently in this regime.

This work adopts an exploratory, data-driven approach to this problem. However, the greedy allocation scheme as described here is inefficient and especially slow on large datasets. An interesting future direction would be to design a fast algorithm to determine optimal allocation, and explore a closed form expression for bit allocation under the Matryoshka property.

2 RELATED WORK

Vector quantization finds its roots in Shannon's work in compression and rate distortion functions [13, 17, 32]. The simplest of these is Scalar Quantization (SQ) [1, 9, 40], which involves discretizing the range of values in a representation uniformly to fewer bits of precision. In the context of embeddings, Product Quantization (PQ) [20, 25] has found ubiquity in reducing the sizes of indices for nearest neighbor search. PQ is used very commonly in industry and is integrated into several nearest-neighbor libraries as a default. While PQ uses k-means on a group of dimensions (commonly referred to as "subvectors"/"chunks"), other techniques inspired by it have also tried out other clustering variants. Recent work has also adapted PQ for online settings by removing the need for pre-processing and look up tables, and modern SIMD-based implementations are incredibly fast [3, 4]. However, typically, these techniques do not provide any guarantees on the quality of the quantized vector in terms of error bounds. More recently, RaBitQ [12] provided a quantization scheme that produces a bit string and provides an unbiased distance estimator with provable, asymptotically optimal error bounds. Moreover, RaBitQ and its variants show superior performance over PQ-style quantizers.

Matryoshka Representation Learning (MRL) was first proposed by Kusupati et al. [21], to allow a single representation to capture information at multiple granularities. Specifically, vectors exhibiting this property allow users to drastically reduce the footprint and computational overhead of using the full vector, with minimal preprocessing (such as truncation). This allows for the same embedding to be used in multiple downstream tasks with varying computational capacities with virtually no additional cost. In the years since its introduction, multiple large embedding models have defaulted to producing vectors that exhibit the Matryoshka property.

There has been recent work on dynamic quantization schemes [5, 35] that adaptively adjust precision at query time under latency constraints. However, they are embedding-agnostic and impose significant computational overhead. Our approach, instead, leverages the inherent structural properties of the embeddings. The adaptivity in our scenario occurs offline during index construction. We use the inherent variance decay in MRL embeddings to perform offline bit allocation. While our work does not focus on an efficient implementation, we still maintain query time performance of static, uniform-width codebooks. There is also a growing body of work on quantization aware training [19, 41]; these methods simulate low-precision arithmetic during the training process with great success. However, they incur significant training overhead. Our approach, in contrast, is post-training quantization, which is more commonly used in practice for its ease in deployment and lower cost.

3 A GREEDY BIT ALLOCATION SCHEME

The core idea behind quantization is that lower precision in data is often sufficient to maintain the global structure of the data. This indicates that for most practical purposes, one can reduce the range of values required to represent the data. Therefore, unlike the traditional 32-bit float data type that is typically used by default, one can reduce the bit complexity and use data types with a smaller bit footprint.

3.1 Matryoshka embeddings

Our work focuses on embeddings generated by *Matryoshka Representation Learning* (MRL) training, a line of investigation initiated by Kusupati et al. [21]. Suppose we have a labeled dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $x_i \in \mathcal{X}$ and $y_i \in [L]$. MRL optimizes a multi-class softmax cross-entropy loss $\mathcal{L} : \mathbb{R}^L \times [L] \to \mathbb{R}_+$ across a set of nested dimensions $m \in \mathcal{M}$.

Each nested dimension m utilizes a separate linear classifier parameterized by $\mathbf{W}^{(m)} \in \mathbb{R}^{L \times m}$. The overall objective minimizes the weighted empirical risk over the feature extractor parameters θ_F and classifiers:

$$\min_{\{\mathbf{W}^{(m)}\}_{m\in\mathcal{M}},\, \theta_F} \frac{1}{N} \sum_{i\in[N]} \sum_{m\in\mathcal{M}} c_m \cdot \mathcal{L}\left(\mathbf{W}^{(m)} \cdot F(x_i;\theta_F)_{1:m}\ ;\ y_i\right). \quad (1)$$

Here $F(x_i;\theta_F)_{1:m}$ denotes the truncation of the extracted feature to its first m coordinates. $c_m \geq 0$ denotes the relative importance scale of dimension m (set to $c_m = 1, \forall m \in \mathcal{M}$ by default).

While the MRL property is inherent to the loss function, a strict mathematical or geometric definition for the resulting embeddings remains elusive. Through empirical investigation, however, we identify an exploitable geometric proxy: the decay in variance across dimensions as the dimension index increases. By construction, the

nested objective concentrates information in the initial coordinates, resulting in higher variance in the early dimensions and lower residual variance in the later dimensions.

This hierarchy of variances manifests in embeddings from models trained with MRL; Figure 2 illustrates this decay in recent models from both Cohere and OpenAI. This variance decay provides an intuitive explanation for why Matryoshka embeddings succeed in practice. When all dimensions of an embedding exhibit similar levels of variance, they carry equal importance. However, concentrating variance in the initial dimensions mimics the leading digits of a numerical value, creating a standalone, coarser (but still informative) representation without the trailing dimensions.

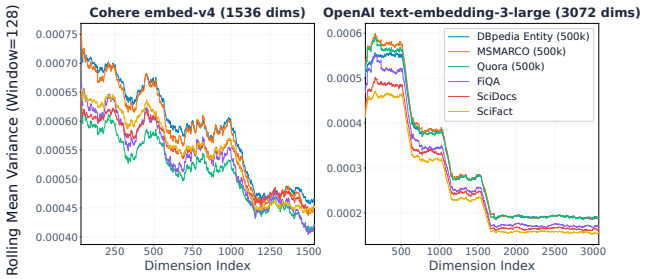

Figure 2: Observed variance differences by dimension in embeddings produced by two different embedding models: OpenAI's `text-embedding-3-large` and Cohere's `embed-v4`. We plot the rolling window average variance per dimension across different datasets and observe very prominent iso-variance "levels" across the dimensions. The rolling window is 128 dimensions in length.

3.2 Greedy Allocation Framework

To systematically evaluate the paper's hypothesis, we propose a variable-bit allocation scheme that allows room for exploiting geometric structure in the embeddings. The framework partitions consecutive dimensions into contiguous buckets and assigns varying bit budgets to each, naturally leading to a structure-aware, variable-bit quantization scheme.

We partition the total embedding dimensionality D into K contiguous buckets $B_1, B_2, \ldots, B_K$. The size of bucket B_k is represented as $d_k = |B_k|$, such that $\sum_{k=1}^{K} d_k = D$. Instead of starting the search from zero bits—which would trivially yield zero recall—the search initializes with a uniform byte allocation vector $\vec{b}^{(0)} = (b_{\text{init}}, \ldots, b_{\text{init}})$ across all buckets.

We perform a greedy local search to find an optimized bit allocation vector $\vec{b} = (b_1, \ldots, b_K)$, where $b_k \geq 0$ represents the budget (in bytes) allocated to bucket B_k. At each iteration, the algorithm evaluates the marginal utility of allocating more memory to a specific region of the embedding. We consider candidate configurations by temporarily increasing the bit budget of each bucket in turn by a fixed step-size increment δ (in bytes). We then evaluate the search recall $R(\vec{b}, \mathcal{V})$ on a validation query set $\mathcal{V}$ using exact distance computation over the quantized dataset. The candidate bucket that yields the maximum empirical recall "wins" the iteration and permanently receives the δ bits. The search proceeds iteratively until the predefined global target memory budget is achieved. This procedure is detailed in Algorithm 1. Note that our algorithm does not explicitly focus on MRL embeddings. However, our results show that this scheme heavily concentrates bits in the leading dimensions, thereby recovering the latent MRL structure that we know these embeddings possess.

Algorithm 1 Greedy Bit Allocation

Input: Initial uniform allocation $\vec{b}^{(0)} = (b_{\text{init}}, \ldots, b_{\text{init}})$, step-size increment δ, validation set $\mathcal{V}$
Output: Optimized allocation vector $\vec{b}$

$\vec{b} \leftarrow \vec{b}^{(0)}$
repeat
  $k^* \leftarrow \arg\max_{k \in \{1,\ldots,K\}} R(\vec{b} + \delta\vec{e}_k, \mathcal{V})$
  $\vec{b} \leftarrow \vec{b} + \delta\vec{e}_{k^*}$
until target budget is achieved
return $\vec{b}$

Note that the algorithm's runtime is linearly proportional to K times the number of increments needed to reach the target budget.

3.2.1 Greedy Product Quantization. In standard Product Quantization (PQ) [20], a high-dimensional vector space is decomposed into lower-dimensional, mutually orthogonal subspaces. These subspaces are quantized independently using separate k-means codebooks, allowing a massive number of effective centroids to be represented compactly. In our greedy variable-bit paradigm, we apply PQ locally within each bucket B_k, strictly constrained by its dynamically assigned byte budget.

Specifically, when a bucket B_k of size d_k receives a budget of b_k bytes, we must partition its d_k dimensions into exactly b_k contiguous subvectors. By utilizing codebooks with $C = 256$ centers, the integer index of the nearest centroid for each subvector fits perfectly into exactly $\log_2(256) = 8$ bits (1 byte) of storage. Thus, the total byte budget precisely dictates the number of subvectors a bucket is split into.

Because d_k is rarely perfectly divisible by b_k during a dynamic allocation, we handle dimension remainders by front-loading them. The remainder dimensions $r = d_k \bmod b_k$ are distributed evenly by adding exactly one extra dimension to the first r subvectors. For example, if a bucket of 10 dimensions is allocated a 3-byte budget, it is partitioned into three subvectors of lengths 4, 3, and 3. This dynamic subvector partitioning and quantization procedure is outlined in Algorithm 2[1].

[1]In every iteration of the greedy search, the algorithm evaluates K candidate allocations. For PQ, evaluating a candidate requires training new k-means codebooks for the upgraded bucket. To keep offline index construction tractable, the trained codebooks $C_k(b)$ are cached for each bucket k and byte-budget b, ensuring the clustering algorithm is executed only once per configuration.

Algorithm 2 PQ Dimension Partitioning

Input: Sub-vector $x^{(k)} \in \mathbb{R}^{d_k}$ of bucket B_k, budget b_k (in bytes), bucket training data $\mathcal{X}_{\text{train}}^{(k)}$
Output: Quantized code vector $\vec{q}^{(k)} \in \{0, \ldots, 255\}^{b_k}$

$s \leftarrow \lfloor d_k / b_k \rfloor$, $r \leftarrow d_k \bmod b_k$
Partition $x^{(k)}$ and $\mathcal{X}_{\text{train}}^{(k)}$ into b_k contiguous subvectors, where the first r subvectors have dimension $s+1$ and the remaining have dimension s.
for each subvector index $j = 1$ **to** b_k **do**
    Train codebook $C_{k,j}$ with 256 centers using k-means on the j-th partition of $\mathcal{X}_{\text{train}}^{(k)}$
    Quantize $x_j^{(k)}$ to closest centroid: $\vec{q}^{(k)}[j] \leftarrow \arg\min_{c \in C_{k,j}} \|x_j^{(k)} - c\|_2$
end for
return $\vec{q}^{(k)}$

3.2.2 Greedy Scalar Quantization. Scalar Quantization (SQ) [9] compresses embeddings by independently discretizing the continuous coordinate values of each dimension into a finite set of representation levels. In our variable-bit paradigm, we represent each dimension using a code with a specific bit-width w_i selected from $\{0, 2, 4, 8\}$[2].

A budget of b_k bytes allocates $N_{\text{bits}} = 8b_k$ total bits to the bucket of size d_k. While standard SQ operates comfortably at higher, uniform bit-widths, modern scale often pushes systems into the extreme sub-1 bit or sub-2 bit regime. In these constrained environments, it is impossible to allocate even the minimum 2-bit width to every dimension. The algorithm must selectively drop (assign 0 bits to) a fraction of the coordinates.

To maintain structural integrity without introducing complex indexing metadata, we distribute these sparse bits as evenly as possible across the d_k dimensions using a uniform stride allocation. We begin by establishing a baseline bit-width $W_{\text{base}} \in \{0, 2, 4\}$ across all dimensions in the bucket. To allocate any remaining bits, we select u dimensions to upgrade to the next higher level $W_{\text{next}} \in \{2, 4, 8\}$ by stepping through the bucket with a uniform stride $\Delta = \lfloor d_k / u \rfloor$. For instance, a 192-dimension bucket with a bit budget $N_{\text{bits}} = 192$ has a baseline of 0 bits and receives 96 2-bit upgrades – we assign 2 bits to every second dimension ($\Delta = 2$) and discard the other dimensions.

Once the bit-widths $\{w_i\}$ are assigned, each dimension is quantized using standard SQ at its respective bit-width. This deterministic allocation and scaling procedure is detailed in Algorithm 3.

[2]A 1-bit quantization splits the distribution directly at its mode, artificially pushing high-density central values to the extremes. To preserve the unimodal nature of standard embedding coordinates, 1-bit widths are explicitly avoided. A bit-width of 0 corresponds to completely discarding, or pruning, the dimension.

Algorithm 3 SQ Bit-Width Allocation

Input: Sub-vector $x^{(k)} \in \mathbb{R}^{d_k}$ of bucket B_k, budget b_k (in bytes)
Output: Quantized code vector $\vec{q}^{(k)}$

Define allowed bit-widths: $S \leftarrow \{0, 2, 4, 8\}$
$N_{\text{bits}} \leftarrow 8 \cdot b_k$
$W_{\text{base}} \leftarrow \max\{w \in S \mid w \cdot d_k \le N_{\text{bits}}\}$
if $W_{\text{base}} = 8$ **then**
    $W_{\text{next}} \leftarrow 8$
    $u \leftarrow 0$
else
    $W_{\text{next}} \leftarrow \min\{w \in S \mid w > W_{\text{base}}\}$
    $u \leftarrow (N_{\text{bits}} - d_k \cdot W_{\text{base}}) / (W_{\text{next}} - W_{\text{base}})$
end if
$w_i \leftarrow W_{\text{base}}$ for $i = 1, \ldots, d_k$
if $u > 0$ **then**
    $\Delta \leftarrow d_k / u$
    Upgrade dimensions along the stride: $w_{\text{round}(j \cdot \Delta)+1} \leftarrow W_{\text{next}}$ for $j = 0, \ldots, u-1$
end if
for each dimension $i = 1$ **to** d_k **do**
    $\vec{q}^{(k)}[i] \leftarrow \text{ScalarQuantize}(x_i^{(k)}, w_i)$
end for
return $\vec{q}^{(k)}$

4 EXPERIMENTS

System. All experiments were run on a Linux x86_64 machine with 16 hardware threads on an Intel Xeon Platinum 8272CL CPU at 2.60 GHz and 32 GiB of main memory, with no GPU. We built DiskANN to use its quantization utilities, using the repository's standard CMake/ C++17 toolchain and the documented Linux dependencies for OpenMP, Boost, and Intel MKL. We do not use DiskANN's graph indexing. We use brute-force L2 distance computation to compute exact Nearest Neighbors at all instances. The code is branched from Microsoft's DiskANN repository in C++ and a link to our implementation is provided in the list of artifacts.

Datasets. We evaluate our results on MS Marco [27], DBpedia-Entity [15], FiQA [23], SciDocs [6], SciFact [39] and Quora; some of the standard datasets for evaluating quantization techniques (MTEB [10, 26]). We draw these datasets from the BEIR benchmark [36] to ensure standardization and reproducibility. For the larger datasets (MS Marco, DBpedia, Quora), the base corpus is truncated to the first 500,000 entries. The smaller datasets are used in their entirety. The evaluation queries are taken from the standard evaluation split, as detailed in Table 1.

Quora represents a symmetric duplicate question detection task (where queries and corpus entries are short questions of similar length), complementing the asymmetric retrieval characteristics of MS Marco and DBpedia-Entity. Furthermore, we also include results on some smaller datasets: FiQA, SciDocs, and SciFact. Structurally, they offer a mix of retrieval paradigms: FiQA and SciFact feature asymmetric tasks mapping short questions or claims to longer text paragraphs, whereas SciDocs evaluates document-to-document retrieval where both queries and corpus entries are of similar length.

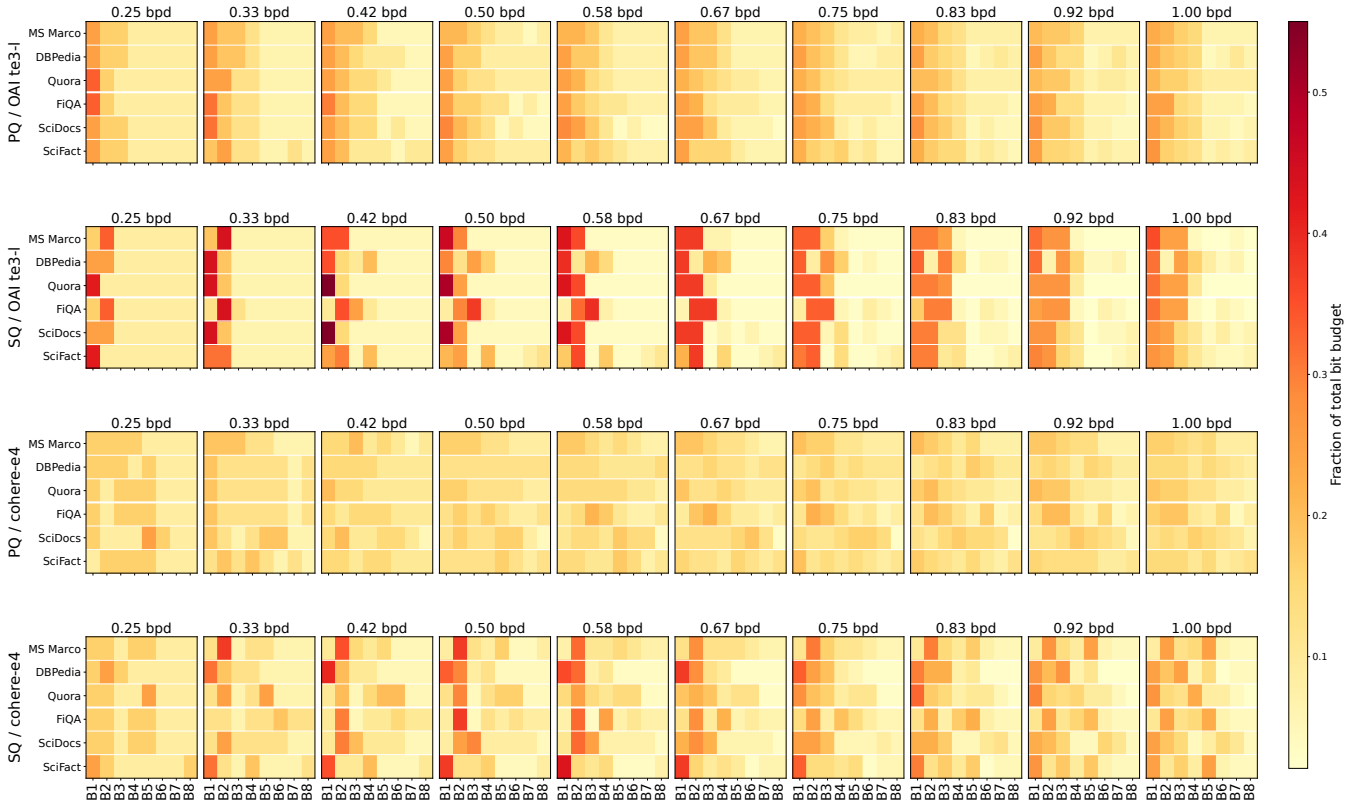

Figure 3: Optimized bit allocation across buckets at different bit budgets, noted by bits per dimension (bpd). Observe that optimized allocation concentrates bits towards the initial buckets, in line with what we expect due to the MRL property.

Dataset	Base Vectors	Queries	Split
MS Marco	500,000	6,980	Dev
DBpedia-Entity	500,000	400	Test
Quora	500,000	10,000	Test
FiQA	57,638	648	Test
SciDocs	25,657	1,000	Test
SciFact	5,183	300	Test

Table 1: Datasets.

Embedding models. We use one of OpenAI's most capable commercial embedding models, `text-embedding-3-large` [29], and Cohere's latest embedding model, `embed-v4` [7]. The full dimensionality of `text-embedding-3-large` is 3072, and that of `embed-v4` is 1536. We generate full-dimension embeddings from both models.

Metric. We compute the average 100-recall@100 over the query set. Crucially, the recall is measured via brute-force exact L2 (Euclidean) distance search over the dequantized (inflated) database vectors. This isolates the quantization error itself, ensuring the metric is independent of any error introduced by approximate indexing schemes.

Setup. The baseline uses uniform allocation of bits across all dimensions. To facilitate a direct comparison between embeddings of different dimensionalities, we present our results in terms of Bits per Dimension (bpd). We evaluate budgets starting from ≈ 0.17 bits per dimension (1/6 bpd) up to 1 bit per dimension, with a step size of 1/12 bits per dimension. For greedy allocation, we split the full dimensions into 8 equal contiguous buckets. Each bucket has 384 dimensions for `text-embedding-3-large` (3072/8 = 384) and 192 dimensions for `embed-v4` (1536/8 = 192).

Hyperparameters. The greedy search starts with an initial uniform allocation of 64 bytes (8 bytes for each of the 8 buckets) for embeddings from `text-embedding-3-large` and of 32 bytes (4 bytes for each of the 8 buckets) for embeddings from `embed-v4`. At each iteration, we consider candidate configurations by increasing the bit budget of each bucket in turn by a step-size increment of $\delta = 8$ bytes and $\delta = 4$ bytes respectively for `text-embedding-3-large` and `embed-v4`. The sweep runs for 40 iterations, at which point it reaches the target total budget of 1 bit per dimension, which is 384 bytes and 192 bytes for `text-embedding-3-large` and `embed-v4` respectively. During the training phase (e.g. for computing product quantization codebooks or scalar quantization scaling factors), we random-sample exactly 10% of the base corpus vectors.

Observations. Figure 1 shows the gains over uniform achieved by our greedy allocation scheme. Moreover, the winning bit allocations

assign more bits to the initial dimensions, as noted in Figure 3. The asymmetry of the optimal bit allocations discovered by our greedy algorithm varies across settings; overall, it is more pronounced in scalar quantizers, and this coincides with the instances of highest relative gains. On the other hand, the PQ case exhibits smoother decay in bit allocation from the leading to the trailing dimensions. For completeness, we also compare with straightforward truncation to the leading dimensions according to the bit budgets (see Table 2). While Figure 1 focuses on the improvement over the uniform baseline, it also plots the actual recall achieved in both the uniform and variable cases. Note the diminishing returns at higher bit budgets, which correspond to the already high baseline recall. We observe that the gains from a variable bit allocation scheme are typically higher for the OpenAI embeddings; this is in line with our observation in Figure 2, where we notice that the variance levels are more distinct in case of the embeddings generated by `text-embedding-3-large`.

5 LIMITATIONS AND FUTURE WORK

We present this work as a preliminary, exploratory exposition on variable allocation schemes for Matryoshka embeddings. Our results tell us that it is *possible* to improve over the uniform bit allocation process using a variable scheme. However, several important questions remain, we list some concrete directions below:

(1) *A concrete mathematical description of MRL embeddings:* The variance observations in Figure 2 show that the variance is a good proxy for the MRL property. However, it remains to show whether a concrete mathematical or geometric description of the embeddings can be derived from the loss function. This would also help formalize a quantization scheme that explicitly exploits the MRL property.
(2) *An efficient allocation algorithm:* The current greedy approach is extremely expensive as it iterates over a lot of candidate allocations. Moreover, this is not a practical approach to finding the best scheme. Moreover, our experiments are limited to fixed, discrete buckets; it is possible that a better quantization exists by taking finer buckets; in our greedy approach, this would blow up the complexity of finding the optimized scheme even more. A practical quantizer requires a more efficient implementation, potentially leveraging a deterministic heuristic, or a lightweight randomized/iterative process.
(3) *Systems challenges:* Dynamic bit allocation introduces significant systems challenges. Storing large volumes of data using custom-sized data types requires adapting existing architectures. This sacrifices the memory access efficiency provided by uniform data types, posing a particular challenge for SIMD-friendly architectures. Especially with constraints like latency, this poses a challenging problem.
(4) *Ablation studies:* A thorough, ablation study of the setup with different numbers of buckets, further metrics (such as recall at different thresholds), and further hyperparameter tuning across further models is required for a better understanding of this landscape.

Cohere embed-v4

Dataset	Bits per Dimension					
	0.17 (32 B) (8d)	0.33 (64 B) (16d)	0.50 (96 B) (24d)	0.67 (128 B) (32d)	0.83 (160 B) (40d)	1.00 (192 B) (48d)
MS Marco	0.24	1.04	2.30	3.87	5.92	8.64
DBpedia	0.20	0.68	1.54	2.38	3.76	5.47
Quora	1.95	7.13	12.59	17.46	22.62	27.62
FiQA	1.19	2.98	5.52	7.72	9.70	12.61
SciFact	5.73	8.22	10.66	12.66	15.36	17.79
SciDocs	2.08	4.15	6.75	8.76	10.61	12.93

OpenAI text-embedding-3-large

Dataset	Bits per Dimension					
	0.17 (64 B) (16d)	0.33 (128 B) (32d)	0.50 (192 B) (48d)	0.67 (256 B) (64d)	0.83 (320 B) (80d)	1.00 (384 B) (96d)
MS Marco	1.80	6.65	13.53	19.98	25.97	31.38
DBpedia	1.21	4.60	9.63	14.86	20.02	24.73
Quora	7.18	19.32	31.13	39.85	46.05	50.97
FiQA	5.67	13.59	21.34	27.93	34.02	38.96
SciFact	10.68	17.93	24.52	29.58	34.02	38.19
SciDocs	5.84	12.71	20.77	27.41	32.96	37.63

Table 2: Recall@100 for simple dimension truncation (no quantization) across different compression rates (bits per dimension). The absolute memory limits (in bytes) and their corresponding 32-bit float truncation dimensions are shown in parentheses.

ACKNOWLEDGMENTS

K. S. Sreeramji was supported by the the Walmart Centre for Tech Excellence at IISc.

REFERENCES

[1] Cecilia Aguerrebere, Ishwar Singh Bhati, Mark Hildebrand, Mariano Tepper, and Theodore Willke. 2023. Similarity Search in the Blink of an Eye with Compressed Indices. *Proc. VLDB Endow.* 16, 11 (July 2023), 3433–3446. https://doi.org/10.14778/3611479.3611537

[2] Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. 2026. jina-embeddings-v5-text: Task-Targeted Embedding Distillation. arXiv:2602.15547 [cs.CL] https://arxiv.org/abs/2602.15547

[3] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2017. Accelerated Nearest Neighbor Search with Quick ADC. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (Bucharest, Romania) *(ICMR '17)*. Association for Computing Machinery, New York, NY, USA, 159–166. https://doi.org/10.1145/3078971.3078992

[4] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. *Proc. VLDB Endow.* 9 (2015), 288–299. https://api.semanticscholar.org/CorpusID:5966664

[5] Mingkai Chen, Cheng Liu, Shengwen Liang, Lei Zhang, Xiaowei Li, and Huawei Li. 2026. ANNS-AMP: Accelerating Approximate Nearest Neighbor Search via Adaptive Mixed-Precision Computing. arXiv:2606.07156 [cs.PF] https://arxiv.org/abs/2606.07156

[6] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *ACL*.

[7] Cohere. 2025. Announcing Embed Multimodal v4. https://docs.cohere.com/changelog/embed-multimodal-v4 Accessed: 2026-06-11.

[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on*

Recommender Systems (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

[9] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2026. The Faiss Library. *IEEE Transactions on Big Data* 12, 2 (2026), 346–361. https://doi.org/10.1109/TBDATA.2025.3618474

[10] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. 2025. MMTEB: Massive Multilingual Text Embedding Benchmark. *arXiv preprint arXiv:2502.13595* (2025). https://doi.org/10.48550/arXiv.2502.13595

[11] Jianyang Gao and Cheng Long. 2023. High-Dimensional Approximate Nearest Neighbor Search: with Reliable and Efficient Distance Comparison Operations. *Proc. ACM Manag. Data* 1, 2, Article 137 (June 2023), 27 pages. https://doi.org/10.1145/3589282

[12] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 3, Article 167 (May 2024), 27 pages. https://doi.org/10.1145/3654970

[13] Allen Gersho and Robert M. Gray. 1992. *Vector Quantization and Signal Compression*. The Springer International Series in Engineering and Computer Science, Vol. 159. Springer.

[14] 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*. https://arxiv.org/abs/1908.10396

[15] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity V2: A Test Collection for Entity Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, 1265–1268. https://doi.org/10.1145/3077136.3080751

[16] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based Retrieval in Facebook Search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2553–2561. https://doi.org/10.1145/3394486.3403305

[17] J. Y. Huang and Peter M. Schultheiss. 1963. Block Quantization of Correlated Gaussian Random Variables. *IEEE Transactions on Communications* 11 (1963), 289–296.

[18] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (Dallas, Texas, USA) *(STOC '98)*. Association for Computing Machinery, New York, NY, USA, 604–613. https://doi.org/10.1145/276698.276876

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2704–2713. https://doi.org/10.1109/CVPR.2018.00286

[20] Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[21] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2022. Matryoshka Representation Learning. In *Advances in Neural Information Processing Systems*. 30233–30249.

[22] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. 2007. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition* 40, 1 (2007), 262–282. https://doi.org/10.1016/j.patcog.2006.04.045

[23] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1941–1942. https://doi.org/10.1145/3184558.3192301

[24] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[25] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. 2018. A Survey of Product Quantization. *ITE Transactions on Media Technology and Applications* 6, 1 (2018), 2–10.

[26] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. MTEB: Massive Text Embedding Benchmark. *arXiv preprint arXiv:2210.07316* (2022). https://doi.org/10.48550/ARXIV.2210.07316

[27] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016 (CEUR Workshop Proceedings)*, Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne (Eds.). CEUR-WS.org. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[28] Nomic AI. 2024. nomic-embed-text-v1.5 Model Repository. https://huggingface.co/nomic-ai/nomic-embed-text-v1.5 Accessed: 2026-06-11.

[29] OpenAI. 2024. New embedding models and API updates. https://openai.com/blog/new-embedding-models-and-api-updates Accessed: 2026-06-11.

[30] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2009. Nearest neighbors in high-dimensional data: the emergence and influence of hubs. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 865–872. https://doi.org/10.1145/1553374.1553485

[31] Madhuri Shanbhogue, Zhe Li, Shanfeng Zhang, Gustavo Hernández Ábrego, Shih-Cheng Huang, Aashi Jain, Daniel Salz, Sonam Goenka, Chaitra Hegde, Ji Ma, Feiyang Chen, Jiaxing Wu, Tanmaya Dabral, Babak Samari, Kevin Poulet, Daniel Cer, Kaifeng Chen, Paul Suganathan, Hui Hui, Jovan Andonov, Philippe Schlattner, Jay Han, Iftekhar Naim, Wing Lowe, Vladimir Pchelin, Albert Yang, Yi-Ting Chen, Zhongli Ding, Grace Zhang, Georg Heigold, Yichang Chen, Antoine Reveillon, Brendan Mccloskey, Wenlei Zhou, Dahun Kim, Rui Meng, Emma Wang, Jack Zheng, Halley Fede, Zhen Yang, Keegan Mosley, Brian Potetz, Sahil Dua, Henrique Schechter Vera, Shen Gao, Hesen Zhang, Andreas Hess, Hengxuan Ying, Alberto Montes, Karan Gill, Min Choi, Sebastian Russo, Anja Hauth, Jinhyuk Lee, Michael Boratko, Megan Barnes, Vikram Rao, Claudiu Musat, Cyril Allauzen, Ehsan Variani, Shankar Kumar, Tom Bagby, Junyi Jiao, Yang Gu, Tengxin Li, Ayush Agrawal, Roberto Santana, Dev Nath, Stephen Karukas, Shuoxuan Han, Lucia Loher, Alice Twu, Nidhi Vyas, Siddharth Bhai, Frank Palma Gomez, Wangyuan Zhang, Chaoren Liu, Jizheng Yang, Steve Qiu, Shijie Zhang, Sujay Kulkarni, Sascha Rothe, Sean Nakamoto, Raphael Hoffmann, Zach Gleicher, Yunhsuan Sung, Qin Yin, Tom Duerig, and Mojtaba Seyedhosseini. 2026. Gemini Embedding 2: A Native Multimodal Embedding Model from Gemini. arXiv:2605.27295 [cs.CV] https://arxiv.org/abs/2605.27295

[32] Claude E. Shannon. 1959. *Coding Theorems for a Discrete Source With a Fidelity Criterion*. Vol. 7. Institute of Radio Engineers, International Convention Record, 325–350. https://doi.org/10.1109/9780470544242.ch21

[33] Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, Gopal Srinivasa, Suhas Jayaram Subramanya, Andrija Antonijevic, Dax Pryce, David Kaczynski, Shane Williams, Siddarth Gollapudi, Varun Sivashankar, Neel Karia, Aditi Singh, Shikhar Jaiswal, Neelam Mahapatro, Philip Adams, Bryan Tower, and Yash Patel. 2023. DiskANN: Graph-structured Indices for Scalable, Fast, Fresh and Filtered Approximate Nearest Neighbor Search. https://github.com/Microsoft/DiskANN

[34] Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. 2023. SOAR: Improved Indexing for Approximate Nearest Neighbor Search. In *Neural Information Processing Systems*. https://arxiv.org/abs/2404.00774

[35] Ganap Ashit Tewary, Nrusinga Charan Gantayat, and Jeff Zhang. 2026. AQR-HNSW: Accelerating Approximate Nearest Neighbor Search via Density-aware Quantization and Multi-stage Re-ranking. arXiv:2602.21600 [cs.IR] https://arxiv.org/abs/2602.21600

[36] 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)*. https://openreview.net/forum?id=wCu6T5xFjeJ

[37] Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, Weiyi Wang, Zhe Li, Gus

Martins, Jinhyuk Lee, Mark Sherwood, Juyeong Ji, Renjie Wu, Jingxiao Zheng, Jyotinder Singh, Abheesht Sharma, Divyashree Sreepathihalli, Aashi Jain, Adham Elarabawy, AJ Co, Andreas Doumanoglou, Babak Samari, Ben Hora, Brian Potetz, Dahun Kim, Enrique Alfonseca, Fedor Moiseev, Feng Han, Frank Palma Gomez, Gustavo Hernández Ábrego, Hesen Zhang, Hui Hui, Jay Han, Karan Gill, Ke Chen, Koert Chen, Madhuri Shanbhogue, Michael Boratko, Paul Suganthan, Sai Meher Karthik Duddu, Sandeep Mariserla, Setareh Ariafar, Shanfeng Zhang, Shijie Zhang, Simon Baumgartner, Sonam Goenka, Steve Qiu, Tanmaya Dabral, Trevor Walker, Vikram Rao, Waleed Khawaja, Wenlei Zhou, Xiaoqi Ren, Ye Xia, Yichang Chen, Yi-Ting Chen, Zhe Dong, Zhongli Ding, Francesco Visin, Gaël Liu, Jiageng Zhang, Kathleen Kenealy, Michelle Casbon, Ravin Kumar, Thomas Mesnard, Zach Gleicher, Cormac Brick, Olivier Lacombe, Adam Roberts, Qin Yin, Yunhsuan Sung, Raphael Hoffmann, Tris Warkentin, Armand Joulin, Tom Duerig, and Mojtaba Seyedhosseini. 2025. EmbeddingGemma: Powerful and Lightweight Text Representations. arXiv:2509.20354 [cs.CL] https://arxiv.org/abs/2509.20354

[38] Voyage AI. 2026. Voyage 4. https://blog.voyageai.com/2026/01/15/voyage-4/ Accessed: 2026-06-11.

[39] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *EMNLP*.

[40] Roger Weber, Hans-Jörg Schek, and Stephen Blott. 1998. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 194–205.

[41] Jiaqi Zhang. 2025. Survey of Quantization-Aware Training (QAT) Applications in Deep Learning Quantization. *Proceedings of the 2025 International Symposium on Artificial Intelligence and Computational Social Sciences* (2025). https://api.semanticscholar.org/CorpusID:284019251

[42] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. arXiv:2506.05176 [cs.CL] https://arxiv.org/abs/2506.05176