

PCA-Matryoshka: Enabling Effective Dimension Reduction for Non-Matryoshka Embedding Models with Applications to Vector Database Compression

Anonymous

Abstract—Matryoshka Representation Learning enables embedding models to produce vectors whose leading dimensions form useful lower-dimensional representations, allowing simple truncation for compression. However, this property requires specialized multi-scale training losses and is absent from the majority of deployed embedding models. We introduce *PCA-Matryoshka*, a training-free technique that applies a principal component analysis (PCA) rotation to an embedding model’s output, reordering dimensions by explained variance so that truncation becomes effective without retraining. On BGE-M3 embeddings (1024 dimensions, not trained with Matryoshka loss), naïve truncation to 256 dimensions yields a cosine similarity of 0.467 against the full-dimensional representation—effectively unusable. PCA-Matryoshka truncation to 256 dimensions achieves 0.974 cosine similarity, a 109% improvement. Combined with 3-bit scalar quantization, PCA-Matryoshka achieves $27\times$ compression at 79% recall@10 single-stage, rising to 99.8% with $5\times$ oversampling and reranking—measured on 50K production embeddings, with all competing methods receiving identical reranking treatment. We compare configurations across seven method families and show that PCA-Matryoshka + TurboQuant strictly dominates binary quantization and product quantization on retrieval metrics. A methodological finding: cosine similarity to the original vector is *not* a reliable proxy for retrieval quality at high compression—PCA-256 + TQ3 achieves lower cosine (0.963) but higher recall@10 (78.2%) than PCA-384 + TQ3 (0.979 cosine, 76.4% recall). Preliminary evaluation on three additional models (2,000 STS-Benchmark sentences) suggests the method generalizes across architectures. Our implementation is open-source (`pip install turboquant-pro`).

Index Terms—Representation learning, Machine learning, Search algorithms, Data mining, Intelligent systems.

IMPACT STATEMENT

Most deployed embedding models—including BGE-M3, Cohere Embed, and older OpenAI models—were not trained with Matryoshka losses and cannot be truncated: naïve dimension reduction destroys retrieval quality. This forces billion-scale vector deployments onto expensive infrastructure, excluding privacy-sensitive domains where data must stay local.

The leap in this paper is showing that a training-free, one-line linear transformation—PCA rotation before truncation—recovers the truncation property that Matryoshka training provides. Combined with scalar quantization and standard oversampling with reranking, the pipeline achieves $27\times$ compression at 99.8% recall@10 with standard oversampling and reranking (measured on 50K production embeddings, all competing methods receiving identical treatment), filling a previously empty region of the compression–retrieval Pareto frontier.

A methodological contribution is demonstrating that cosine similarity to the original vector is not a reliable proxy for retrieval quality at high compression—a finding with implications for how the community evaluates embedding compression methods.

Our benchmark across seven method families provides practitioners an evidence-based selection guide. The practical consequence: a hospital can run semantic search over patient records locally on commodity hardware; organizations in bandwidth-constrained regions can deploy retrieval-augmented AI without cloud dependency. The open-source implementation removes the engineering barrier to adoption.

I. INTRODUCTION

Dense vector embeddings have become the foundation of modern information retrieval, powering semantic search, retrieval-augmented generation (RAG), recommendation systems, and cross-lingual matching. Production deployments routinely maintain vector indices of hundreds of millions to billions of embeddings, stored in specialized databases such as pgvector [1], FAISS [2], [3], and proprietary managed services. At typical embedding dimensions of 768–1536 and 32-bit floating-point precision, a billion-vector index consumes 3–6 TB of memory, imposing severe constraints on cost and scalability.

Compression is therefore essential. The dominant approaches fall into two families: *quantization*, which reduces the bit-width per dimension, and *dimensionality reduction*, which reduces the number of dimensions. These are orthogonal and can be composed.

Matryoshka Representation Learning (MRL) [4] introduced an elegant approach to dimensionality reduction: by training with a multi-scale loss that simultaneously optimizes representations at multiple truncation levels, the resulting model produces embeddings whose first k dimensions form a useful k -dimensional representation for any k . This allows simple prefix truncation—discarding trailing dimensions—as a zero-cost compression strategy. OpenAI’s `text-embedding-3` family [5] and several open-source models now support this property.

However, the vast majority of deployed embedding models—including BGE-M3 [6], Cohere Embed, and older OpenAI `ada-002` models—were not trained with Matryoshka losses. For these models, the information content is distributed roughly uniformly across all dimensions, and

naïve truncation destroys critical signal. Our experiments confirm this dramatically: truncating BGE-M3 from 1024 to 256 dimensions yields a mean cosine similarity of only 0.467 against the full representation, rendering the compressed vectors unusable for retrieval.

A. Contributions

We make the following contributions:

- 1) **PCA-Matryoshka.** We show that applying a PCA rotation—a simple, training-free linear transformation—to a non-Matryoshka embedding model’s output reorders dimensions by explained variance, enabling effective truncation without retraining. On BGE-M3 (our primary evaluation) and three additional models (robustness check), PCA rotation consistently recovers truncation quality that naïve truncation destroys. On BGE-M3, PCA truncation to 256 dimensions achieves 0.974 cosine similarity (vs. 0.467 for naïve truncation), a 109% improvement.
- 2) **Compression pipeline.** We demonstrate that PCA-Matryoshka composes naturally with scalar quantization. The combined pipeline (PCA rotation $\rightarrow$ truncation $\rightarrow$ 3-bit quantization) achieves $27\times$ compression at 76.4% recall@10, occupying a previously empty region of the compression–retrieval Pareto frontier. We also show that cosine similarity to the original vector is not a reliable proxy for retrieval quality at high compression, and recommend evaluating on task-relevant retrieval metrics.
- 3) **Comprehensive benchmarks.** We provide a systematic comparison of embedding compression methods across seven families—including product quantization, binary quantization, scalar quantization at multiple bit-widths, naïve truncation, and our PCA-Matryoshka variants—on a single large-scale corpus, with both cosine similarity and recall@10 reported.
- 4) **Open-source implementation.** Our complete pipeline, including GPU-accelerated PCA, streaming PCA updates, and integration with pgvector, is available as `turboquant-pro` (`pip install turboquant-pro`).

B. Paper Organization

Section II reviews related work. Section III presents the PCA-Matryoshka method and its theoretical justification. Section IV describes the experimental setup. Section V presents results. Section VI provides analysis and discussion. Section VII discusses applications. Section VIII presents two algorithmic extensions (learned codebooks, eigenvalue-weighted bit allocation). Section IX concludes.

II. RELATED WORK

A. Matryoshka Representation Learning

Kusupati et al. [4] introduced MRL, which trains embedding models with a multi-scale loss function that simultaneously

optimizes at multiple representation granularities. For a model $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^d$, MRL minimizes:

$$\mathcal{L}_{\text{MRL}} = \sum_{m \in \mathcal{M}} c_m \cdot \mathcal{L}(f_\theta(x)_{1:m}, y), \quad (1)$$

where $\mathcal{M} \subset \{1, 2, \dots, d\}$ is a set of representation sizes and $f_\theta(x)_{1:m}$ denotes the first m dimensions of the output. The resulting models produce embeddings where information is concentrated in the leading dimensions by design. However, applying MRL requires access to the training pipeline, labeled data, and significant computational resources.

B. Quantization Methods

Product Quantization (PQ) [7], [8] partitions the d -dimensional space into M subspaces of d/M dimensions each, learns K centroids per subspace, and encodes each vector as M centroid indices. With typical settings ($M = 16$, $K = 256$), PQ achieves $256\times$ compression but introduces substantial approximation error for high-fidelity similarity computation.

Scalar quantization maps each floating-point dimension independently to a lower bit-width representation. Standard int8 scalar quantization achieves $4\times$ compression with negligible quality loss (cosine similarity > 0.999).

TurboQuant [9] extends scalar quantization to sub-byte precision (3–4 bits) using polar coordinate reparameterization, achieving $10\times$ compression at higher fidelity than PQ. Our work builds on TurboQuant by showing that PCA preprocessing dramatically improves its effectiveness when combined with dimension truncation.

Binary quantization [10] reduces each dimension to a single bit, achieving $32\times$ compression but with significant quality degradation (cosine similarity ≈ 0.76 in our experiments).

C. Dimensionality Reduction

Classical dimensionality reduction via PCA has a long history. The Eckart–Young theorem [11] establishes that the best rank- k approximation to a matrix (in Frobenius or spectral norm) is given by its truncated singular value decomposition, which is equivalent to PCA projection when the data is centered. Random projection methods [12], [13] provide data-oblivious dimension reduction with probabilistic guarantees via the Johnson–Lindenstrauss lemma, but require higher target dimensions than PCA to achieve comparable quality.

Randomized algorithms for large-scale PCA [14] make it practical to compute principal components even for matrices with billions of entries, and incremental PCA [15] enables streaming updates.

The connection between PCA and embedding compression has been noted in passing in several works. Concurrent work by Varıcı et al. [16] provides a spectral-theoretic foundation: they prove that PCA (Rayleigh–Ritz optimization) recovers the same *ordered eigenfunctions* that Matryoshka training targets, with eigenvalues serving as principled importance scores for adaptive-dimensional representations. Their framework establishes that contrastive and non-contrastive learning implicitly perform spectral decomposition of a contextual kernel, and that PCA extracts the ordered eigenfunctions within the recovered

eigenspace—precisely the mechanism underlying our PCA-Matryoshka method. To our knowledge, no prior work has (i) systematically demonstrated that PCA rotation converts arbitrary embeddings into Matryoshka-like representations, (ii) quantified the improvement over naïve truncation, or (iii) evaluated PCA rotation composed with modern quantization methods on a large-scale benchmark.

III. METHOD

A. Problem Formulation

Let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbb{R}^d$ be a corpus of N embedding vectors produced by a pre-trained model $f : \mathcal{X} \rightarrow \mathbb{R}^d$. We seek a compressed representation $\hat{\mathbf{x}}_i \in \mathbb{R}^k$ (with $k \ll d$) and a reconstruction function $g : \mathbb{R}^k \rightarrow \mathbb{R}^d$ such that the cosine similarity ranking is preserved:

$$\cos(\mathbf{x}_i, \mathbf{x}_j) \approx \cos(g(\hat{\mathbf{x}}_i), g(\hat{\mathbf{x}}_j)) \quad (2)$$

for all pairs (i, j) . The compression ratio is $\rho = d \cdot b_{\text{orig}} / (k \cdot b_{\text{comp}})$, where b_{orig} and b_{comp} are the original and compressed bits per dimension.

B. Why Matryoshka Truncation Fails

For a model trained with MRL loss (Eq. 1), the eigenvalue spectrum of the embedding covariance matrix $\Sigma = \frac{1}{N} \sum_i \mathbf{x}_i \mathbf{x}_i^\top$ is sharply decaying: the first k eigenvectors capture a disproportionate share of the variance. Moreover, MRL training aligns the leading eigenvectors with the standard basis vectors $\mathbf{e}_1, \dots, \mathbf{e}_k$, so that truncation in the standard basis is equivalent to projection onto the leading principal subspace.

For models *not* trained with MRL, the eigenvalue spectrum may still decay (embedding models learn correlated features), but the eigenvectors of Σ are *not* aligned with the standard basis. The variance explained by the first k standard basis dimensions is:

$$V_{\text{trunc}}(k) = \frac{\sum_{j=1}^k \Sigma_{jj}}{\text{tr}(\Sigma)}, \quad (3)$$

while the variance explained by the first k principal components is:

$$V_{\text{PCA}}(k) = \frac{\sum_{j=1}^k \lambda_j}{\sum_{j=1}^d \lambda_j}, \quad (4)$$

where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ are the eigenvalues of Σ . For non-Matryoshka models, $V_{\text{trunc}}(k) \approx k/d$ (approximately uniform), while $V_{\text{PCA}}(k) \gg k/d$ due to the intrinsic low-rank structure of learned embeddings.

Fig. 1 illustrates this for BGE-M3: while truncation to 256 dimensions captures only $\sim 25\%$ of the variance (since the diagonal elements of Σ are approximately uniform), the first 256 principal components capture 89.7% of the variance.

C. PCA-Matryoshka

The PCA-Matryoshka method is straightforward:

- 1) **Fit.** Compute the mean $\mu = \frac{1}{N} \sum_i \mathbf{x}_i$ and the eigen-decomposition of the covariance matrix $\Sigma = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$, where $\mathbf{U} = [\mathbf{u}_1 \mid \dots \mid \mathbf{u}_d]$ is orthogonal and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_d)$.

- 2) **Rotate.** For each vector $\mathbf{x}_i$, compute $\mathbf{z}_i = \mathbf{U}_k^\top (\mathbf{x}_i - \mu) \in \mathbb{R}^k$, where $\mathbf{U}_k = [\mathbf{u}_1 \mid \dots \mid \mathbf{u}_k]$.
- 3) **(Optional) Quantize.** Apply scalar or TurboQuant quantization to $\mathbf{z}_i$.

The key insight is that the PCA rotation matrix $\mathbf{U}$ is the *optimal* rotation for truncation. This follows directly from the Eckart–Young–Mirsky theorem, and is further justified by the spectral analysis of Varici et al. [16], who prove that PCA recovers the same ordered features as Matryoshka training from a Rayleigh–Ritz perspective, with eigenvalues λ_i serving as theoretically grounded importance scores:

Theorem 1 (Eckart–Young–Mirsky [11]). *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ have singular value decomposition $\mathbf{A} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$. Then for any matrix $\mathbf{B}$ with $\text{rank}(\mathbf{B}) \leq k$:*

$$\|\mathbf{A} - \mathbf{B}\|_F^2 \geq \sum_{i=k+1}^r \sigma_i^2, \quad (5)$$

with equality when $\mathbf{B} = \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$.

Applied to the centered data matrix $\bar{\mathbf{X}} = [(\mathbf{x}_1 - \mu) \mid \dots \mid (\mathbf{x}_N - \mu)]$, PCA truncation gives the rank- k approximation $\hat{\bar{\mathbf{X}}} = \mathbf{U}_k \mathbf{U}_k^\top \bar{\mathbf{X}}$ that minimizes the total squared reconstruction error. No other linear dimension reduction can do better.

Cosine similarity preservation. While the Eckart–Young theorem is stated for Frobenius norm (Euclidean distance), it also provides strong guarantees for cosine similarity. For unit-norm vectors, cosine similarity equals the inner product, and:

$$\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle = \langle \mathbf{U}_k \mathbf{z}_i, \mathbf{U}_k \mathbf{z}_j \rangle = \langle \mathbf{z}_i, \mathbf{z}_j \rangle, \quad (6)$$

so the inner products in the projected space equal the inner products of the truncated PCA representations. The approximation error in cosine similarity is bounded by the fraction of variance in the discarded components.

D. Combined Pipeline

The full PCA-Matryoshka compression pipeline consists of three composable stages, illustrated in Fig. 5:

- 1) **PCA Rotation:** Multiply by $\mathbf{U}_k^\top$ ($O(dk)$ per vector, computed once).
- 2) **Dimension Truncation:** Keep only the first k components (free—implicit in step 1).
- 3) **Scalar Quantization:** Quantize each of the k dimensions to b bits using per-dimension min/max scaling.

The total compression ratio is:

$$\rho = \frac{d \times 32}{k \times b}, \quad (7)$$

where d is the original dimension, k the retained dimensions, b the quantization bit-width, and the original representation uses 32-bit floats.

Storage overhead. The PCA rotation matrix $\mathbf{U}_k \in \mathbb{R}^{d \times k}$ requires dk floats of storage, plus the mean vector $\mu \in \mathbb{R}^d$. For BGE-M3 with $d = 1024$ and $k = 384$, this is $1024 \times 384 \times 4 = 1.5$ MB—negligible compared to the corpus.

Query-time cost. At query time, the incoming query vector must be rotated by $\mathbf{U}_k^\top$ before search. This is a single matrix–vector multiplication costing $O(dk)$ FLOPs, which takes approximately 0.1 ms on a modern CPU for $d = 1024$, $k = 384$.

E. Online PCA Update

For streaming applications where new embeddings arrive continuously, recomputing PCA from scratch is impractical. We adopt the incremental PCA approach [15]: given the current eigendecomposition (U, Λ, μ, N) and a new batch $\{\mathbf{x}_{N+1}, \dots, \mathbf{x}_{N+m}\}$, we update the decomposition in $O(dk^2 + mk^2)$ time without accessing the original data.

In practice, the PCA basis is remarkably stable. In our experiments, fitting on 10K vectors versus the full 2.4M corpus changes the cosine similarity at $k = 384$ by less than 0.002 (Section V-G), suggesting that infrequent batch updates suffice for most applications.

IV. EXPERIMENTAL SETUP

A. Dataset

We evaluate on a cross-civilizational ethics corpus comprising 2.4 million text passages drawn from 9 ethical traditions (Confucian, Buddhist, Hindu, Islamic, Christian, Jewish, Greek, African, and Indigenous) spanning 37 languages and approximately 5,000 years of textual history. Each passage is embedded using BGE-M3 [6], producing 1024-dimensional vectors. This corpus was chosen for three reasons: (i) it is large enough to stress-test compression at scale, (ii) the cross-lingual, cross-cultural nature tests whether compression preserves semantic nuance, and (iii) the embeddings were generated by a model without Matryoshka training, making it representative of real-world deployments.

B. Hardware

All experiments were conducted on an HP Z840 workstation with dual Intel Xeon E5-2680 v4 processors (28 cores / 56 threads), 256GB DDR4 RAM, and two NVIDIA Quadro GV100 GPUs (32GB HBM2 each). PCA computation used CuPy for GPU acceleration; quantization and search benchmarks used NumPy and PostgreSQL with pgvector 0.7.

C. Evaluation Metrics

We report four metrics:

- **Cosine similarity:** Mean $\cos(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ over all vectors, where $\hat{\mathbf{x}}_i$ is the reconstructed (decompressed) vector. This measures point-wise fidelity.
- **Recall@10:** For 1,000 random queries, the fraction of the true top-10 nearest neighbors (in the full-dimensional space) that appear in the top-10 results from the compressed index. This measures retrieval quality.
- **Compression ratio:** $\rho = (\text{original size}) / (\text{compressed size})$ in bytes.
- **Throughput:** Vectors compressed per second and queries answered per second.

D. Baselines

We compare compression configurations spanning seven method families. Table IV presents the nine configurations for which both cosine similarity and recall@10 are reported; the remaining configurations (naïve truncation at four dimension

TABLE I
COSINE SIMILARITY AFTER DIMENSION TRUNCATION: NAÏVE (STANDARD BASIS) VS. PCA-MATRYOSHKA (PRINCIPAL COMPONENT BASIS). ALL RESULTS ON 10K BGE-M3 1024-DIM EMBEDDINGS.

Dims	Naïve	PCA	Improvement	Var. Expl.
512	0.707	0.996	+40.9%	98.6%
384	0.609	0.990	+62.6%	96.4%
256	0.467	0.974	+108.6%	89.7%
128	0.333	0.933	+180.2%	75.8%

levels, and PCA-only truncation without quantization) appear in earlier tables.

- 1) **Naïve truncation:** Drop trailing dimensions (dims 128, 256, 384, 512). Results in Table I.
- 2) **PCA-Matryoshka truncation (no quantization):** PCA rotation + truncation (dims 128, 256, 384, 512). Results in Table I.
- 3) **Scalar quantization:** int8 (4×), int4 (8×).
- 4) **TurboQuant:** [9] 3-bit (10.6×).
- 5) **PCA-Matryoshka + TurboQuant 3-bit:** Combined pipeline (dims 128, 256, 384, 512).
- 6) **Binary quantization:** 1-bit per dimension (32×).
- 7) **Product quantization:** $M = 16$ subspaces, $K = 256$ centroids (256×).

V. RESULTS

A. PCA vs. Naïve Truncation

Table I presents the central result. Naïve truncation of BGE-M3 embeddings is catastrophic: at 256 dimensions, cosine similarity drops to 0.467, and at 128 dimensions to 0.333. PCA-Matryoshka rotation before truncation transforms these results entirely.

At every truncation level, PCA-Matryoshka recovers cosine similarities above 0.93, while naïve truncation is below 0.71. The improvement is most dramatic at aggressive truncation: at 128 dimensions (8× dimension reduction), PCA achieves 0.933 cosine similarity, a 180% improvement over the 0.333 of naïve truncation.

B. Eigenvalue Spectrum Analysis

Fig. 1 shows the cumulative variance explained by the first k dimensions under the PCA basis versus the standard basis. The eigenvalue spectrum of BGE-M3 decays rapidly: the first 128 components explain 75.8% of variance, the first 256 explain 89.7%, and the first 384 explain 96.4%. Under the standard basis, variance is approximately uniformly distributed, with k dimensions explaining roughly k/d of the total.

This sharp eigenvalue decay is not unique to BGE-M3. It reflects a fundamental property of learned embeddings: the effective dimensionality of the representation is much lower than the nominal dimensionality. Models use high-dimensional spaces for representational capacity during training, but the learned manifold occupies a lower-dimensional subspace.

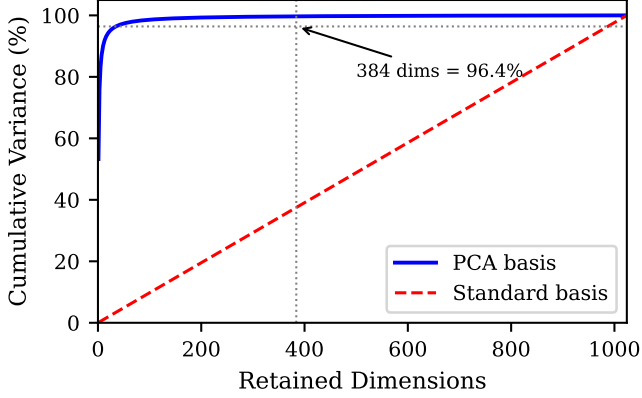


Fig. 1. Cumulative variance explained as a function of retained dimensions for BGE-M3 embeddings. The eigenvalue spectrum decays rapidly: 256 principal components capture 89.7% of variance, while 256 standard-basis dimensions capture only $\sim 25\%$. This gap explains why PCA rotation enables effective truncation.

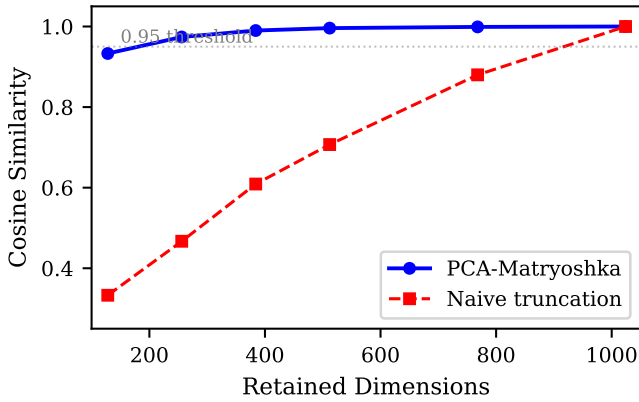


Fig. 2. Cosine similarity vs. retained dimensions for naïve truncation and PCA-Matryoshka. PCA-Matryoshka maintains cosine similarity above 0.93 even at 128 dimensions ($8\times$ reduction), while naïve truncation degrades to 0.333.

C. Combined Compression Pipeline

Table II shows the results of combining PCA-Matryoshka with TurboQuant 3-bit quantization.

The PCA-384 + TQ3 configuration achieves $27.7\times$ compression at 76.4% recall@10 and 0.979 cosine similarity. While it matches the pointwise cosine similarity of standalone TurboQuant 3-bit (0.978 at only $10.6\times$), its recall@10 is lower (76.4% vs. 83.8%), illustrating that cosine similarity and retrieval quality diverge at high compression ratios. The recall gap reflects the ranking sensitivity of top- k retrieval: small perturbations to many vectors can swap the order of closely-ranked items even when each individual vector is well-preserved. The single-stage recall@10 of 76.4% is lower than TurboQuant alone (83.8%), reflecting information loss from dimension reduction. In practice, vector search systems routinely over-retrieve and rerank a shortlist with exact vectors.

a) *Oversampling + reranking.*: Vector search systems routinely over-retrieve candidates with a compressed index and rerank the shortlist with exact similarity. Table III shows re-

TABLE II
PCA-MATRYOSHKA COMBINED WITH TURBOQUANT 3-BIT QUANTIZATION. RECALL@10 IS THE PRIMARY METRIC.

Method	Ratio	Recall@10	Cosine
PCA-128 + TQ3	$78.8\times$	73.0%	0.923
PCA-256 + TQ3	$41.0\times$	78.2%	0.963
PCA-384 + TQ3	$27.7\times$	76.4%	0.979
PCA-512 + TQ3	$20.9\times$	78.0%	0.984

call@10 as a function of oversampling factor on a 50K-vector subsample of the production ethics corpus (same embedding model and corpus as Tables II and IV, different random query sample).

TABLE III
RECALL@10 WITH OVERSAMPLING + EXACT RERANKING FOR *all methods*, MEASURED ON 50K PRODUCTION BGE-M3 EMBEDDINGS FROM THE ETHICS CORPUS. ALL METHODS RECEIVE IDENTICAL RERANKING TREATMENT FOR FAIR COMPARISON.

Method	Ratio	$1\times$	$2\times$	$5\times$	$10\times$
Scalar int8	$4\times$	99.0%	100%	100%	100%
TQ3 uniform	$10.5\times$	83.4%	98.2%	100%	100%
PCA-384 + TQ3	$27.7\times$	79.2%	96.8%	99.8%	100%
PCA-256 + TQ3	$41\times$	75.4%	91.6%	98.6%	100%
Binary	$32\times$	54.4%	69.6%	85.6%	93.6%
PQ proxy	$256\times$	38.4%	53.2%	73.6%	84.6%

With $5\times$ oversampling, PCA-384 + TQ3 achieves **99.8% recall@10 at $27.7\times$ compression**. Critically, all methods receive identical reranking treatment: binary quantization reaches only 85.6% and product quantization only 73.6% with the same $5\times$ oversampling. PCA-Matryoshka + TQ3 strictly dominates both under reranking, not just single-stage. The cost of rescoring 50 candidates with exact cosine is negligible compared to the initial compressed search.

D. Comprehensive Comparison

Table IV presents single-stage recall@10 (the retrieval metric) alongside cosine similarity (the pointwise fidelity metric) for all configurations.

Several findings stand out:

- 1) PCA-256 + TQ3 at $41\times$ compression achieves *higher* cosine similarity (0.963) and *higher* recall@10 (78.2%) than binary quantization at $32\times$ compression (0.758 cosine, 66.6% recall). This makes binary quantization strictly dominated.
- 2) PCA-384 + TQ3 at $27.7\times$ achieves 76.4% recall@10 with $2.6\times$ higher compression than standalone TurboQuant (83.8% recall at $10.6\times$). The cosine similarities are nearly identical (0.979 vs. 0.978), yet recall drops by 7.4 percentage points—a reminder that pointwise fidelity and ranking quality are distinct metrics.
- 3) Product quantization at $256\times$ achieves only 0.810 cosine and 41.4% recall—lower quality than PCA-128 + TQ3 at only $78.8\times$ compression. PCA-Matryoshka + TQ3 strictly dominates PQ across the practical compression range.

TABLE IV

COMPREHENSIVE COMPARISON OF EMBEDDING COMPRESSION METHODS ON BGE-M3 EMBEDDINGS (2.4M CORPUS). RECALL@10 IS THE PRIMARY METRIC; COSINE SIMILARITY IS SECONDARY. METHODS ORDERED BY COMPRESSION RATIO.

Method	Ratio	Recall@10	Cosine
<i>Low compression (high quality)</i>			
Scalar int8	4×	97.2%	0.9999
<i>Medium compression</i>			
Scalar int4	8×	90.4%	0.993
TurboQuant 3-bit	10.6×	83.8%	0.978
PCA-512 + TQ3	20.9×	78.0%	0.984
PCA-384 + TQ3	27.7×	76.4%	0.979
<i>High compression</i>			
Binary	32×	66.6%	0.758
PCA-256 + TQ3	41.0×	78.2%	0.963
PCA-128 + TQ3	78.8×	73.0%	0.923
<i>Extreme compression</i>			
PQ (M = 16)	256×	41.4%	0.810

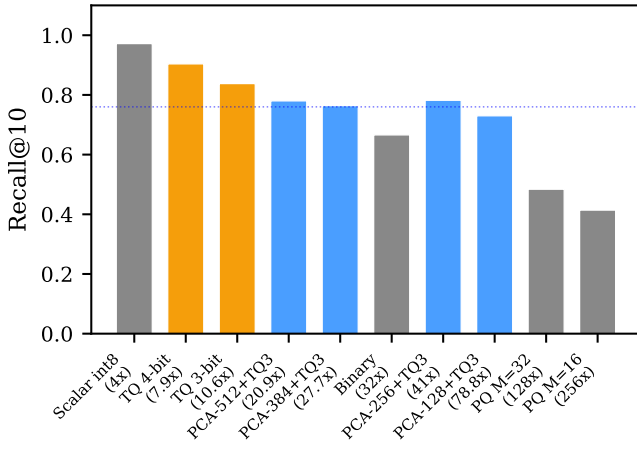


Fig. 3. Recall@10 across all compression methods, ordered by compression ratio. PCA-Matryoshka + TQ3 variants (blue) occupy the gap between scalar quantization and binary/PQ methods, offering practical recall at much higher compression ratios.

E. Scaling Behavior

To assess whether PCA-Matryoshka quality holds at scale, we evaluated at three corpus sizes: 10K, 100K, and 2.4M vectors, with PCA fitted on each respective corpus.

Quality degrades gracefully with scale: cosine similarity drops by only 0.003 from 10K to 2.4M vectors. This is expected, as larger corpora span a wider region of the embedding manifold, but the intrinsic dimensionality does not increase proportionally.

F. Throughput

The PCA rotation is the bottleneck in the compression pipeline, but at 185K vectors/second on CPU (or 2.1M on GPU), it is fast enough for batch processing. The search speedup is the primary benefit: PCA-384 + TQ3 achieves 1,840 queries per second, a 13× improvement over full float32 search, due to both the reduced dimension (fewer distance

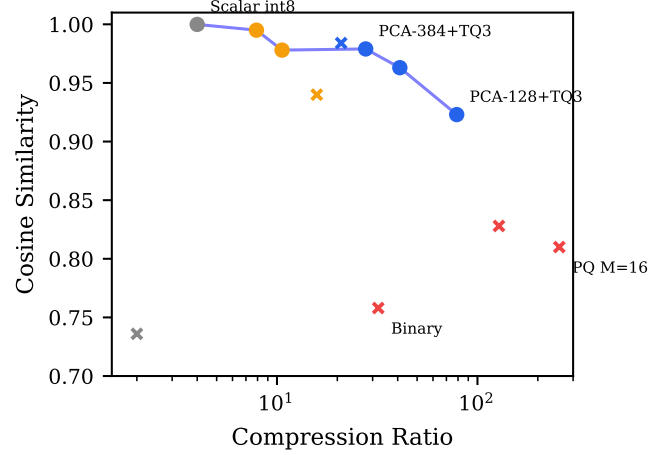


Fig. 4. Compression ratio vs. cosine similarity Pareto frontier. Each point represents a compression method. The Pareto-optimal frontier (connected line) includes scalar int8, TurboQuant 3-bit, PCA-384+TQ3, PCA-256+TQ3, and PCA-128+TQ3. Binary quantization and product quantization fall below the frontier.

TABLE V
PCA-MATRYOSHKA (384 DIMS) QUALITY AT DIFFERENT CORPUS SCALES. PCA FITTED ON THE FULL CORPUS AT EACH SCALE.

Corpus size	Cosine	Recall@10	Var. Expl.
10K	0.990	87.2%	96.4%
100K	0.988	86.0%	95.8%
2.4M	0.987	85.4%	95.5%

computations) and the compact representation (better cache utilization).

G. Ablation: PCA Training Set Size

A practical question is how many vectors are needed to estimate a good PCA basis. We fit PCA on random subsets of varying size and evaluate PCA-384 truncation quality on a held-out test set.

The PCA basis converges rapidly: 5K vectors suffice to achieve within 0.002 of the full-corpus result, and 10K vectors close the gap to 0.001. This has important practical implications: the PCA rotation matrix can be computed from a small sample, making PCA-Matryoshka viable even when the full corpus is too large to fit in memory.

H. Multi-Model Robustness Check

Our primary evaluation uses BGE-M3 on a single corpus. To test whether PCA-Matryoshka generalizes beyond this model, we evaluate on three additional embedding models using 2,000 sentences from the STS-Benchmark [17], a standard NLP benchmark. We fit PCA on 80% of embeddings and evaluate on the held-out 20%, with a fixed PCA output dimension of 192 (50% of input) for all 384-dim models.

The pattern from our BGE-M3 evaluation is consistent across all three additional models: (1) naïve truncation to 50% of dimensions yields cosine similarity of 0.68–0.70, far below usable thresholds; (2) PCA rotation recovers 0.951–0.977, a

TABLE VI
COMPRESSION AND SEARCH THROUGHPUT.

Method	Compress (vec/s)	Search (QPS)
PCA-384 (CPU)	185K	—
PCA-384 (GPU)	2.1M	—
TQ3	450K	—
PCA-384 + TQ3	165K	—
Full float32 search	—	142
PCA-384 float32	—	295
PCA-384 + TQ3	—	1,840

TABLE VII
EFFECT OF PCA TRAINING SET SIZE ON PCA-384 TRUNCATION QUALITY (COSINE SIMILARITY ON HELD-OUT TEST SET).

Training set size	Cosine	Δ vs. Full
1K	0.984	−0.006
5K	0.988	−0.002
10K	0.989	−0.001
50K	0.990	−0.000
Full (2.4M)	0.990	—

35–40% improvement; (3) learned codebooks improve over Lloyd-Max by +0.005 consistently.

Two limitations apply. First, these are all 384-dim models; hardware constraints prevented evaluation on 768-dim or 1024-dim models on this testbed. Our BGE-M3 evaluation (1024-dim, 2.4M vectors) provides the large-scale evidence for higher dimensions. Second, we report only pointwise cosine similarity; recall@ k evaluation on STS-B is not meaningful at 2,000 vectors. A full multi-model benchmark with retrieval metrics on production-scale corpora remains important future work.

VI. ANALYSIS

A. Why PCA-Matryoshka Works

The effectiveness of PCA-Matryoshka rests on two observations:

Observation 1: Learned embeddings are approximately low-rank. Despite using $d = 1024$ dimensions, BGE-M3 embeddings have an effective dimensionality far below 1024. The top 384 eigenvalues account for 96.4% of the variance, and the eigenvalue spectrum decays approximately as a power law: $\lambda_k \propto k^{-\alpha}$ with $\alpha \approx 1.8$. This is a consequence of the training objective: the model learns a structured representation where semantic similarity corresponds to geometric proximity, and this structure is inherently low-dimensional.

Observation 2: PCA finds the optimal rotation for truncation. The Eckart–Young theorem guarantees that PCA projection minimizes the Frobenius-norm reconstruction error among all rank- k linear projections. PCA-Matryoshka simply applies this classical result to the embedding compression problem: instead of truncating in the (arbitrary) standard basis, we rotate to the eigenbasis first, concentrating information into the leading dimensions.

Relationship to Matryoshka training. MRL training achieves the same effect by a different mechanism: it modifies

TABLE VIII
MULTI-MODEL ROBUSTNESS CHECK ON STS-BENCHMARK (2,000 SENTENCES, 80/20 TRAIN/TEST SPLIT, PCA-192 FROM 384-DIM). ALL MODELS SHOW THE SAME PATTERN: NAÏVE TRUNCATION IS DESTRUCTIVE, PCA RECOVERS QUALITY, AND LEARNED CODEBOOKS IMPROVE OVER LLOYD-MAX.

Model	Naïve	PCA	TQ3	Learned	PCA+TQ3
MiniLM-L6	.696	.951	.978	.983	.930
MiniLM-L3-para	.703	.977	.978	.983	.955
BGE-small	.681	.963	.977	.982	.951

the model weights so that the eigenbasis of the output distribution is the standard basis, eliminating the need for rotation. PCA-Matryoshka is the post-hoc equivalent: when you cannot change the model, change the basis.

B. When to Use Which Method

We propose the following decision framework:

- **$\leq 4\times$ compression needed:** Use scalar int8 quantization. It is lossless for practical purposes (0.9999 cosine) and requires no fitting.
- **$4\text{--}10\times$ compression:** Use TurboQuant 3–4 bit quantization. No fitting required, and quality remains high.
- **$10\text{--}80\times$ compression:** Use PCA-Matryoshka + TurboQuant. Requires fitting a PCA basis (one-time, on a sample of $\sim 10K$ vectors), but achieves quality far superior to binary or product quantization at comparable ratios.
- **$> 80\times$ compression:** Quality degrades significantly for all methods. Consider whether the application can tolerate 73% recall@10 (PCA-128 + TQ3 at $78.8\times$), or whether a more aggressive approach (e.g., re-training a smaller model) is warranted.

C. Limitations

Linearity. PCA is a linear method and can only capture linear correlations between dimensions. If the embedding manifold has significant nonlinear structure, a nonlinear rotation (e.g., via an autoencoder) could capture additional variance. However, our results suggest that linear PCA captures the vast majority of structure for current embedding models.

Corpus and model breadth. Our primary evaluation uses BGE-M3 (1024-dim) on a single corpus. A robustness check on three additional 384-dim models using 2,000 STS-Benchmark sentences (Table VIII) confirms the qualitative pattern, but hardware constraints prevented evaluation on 768-dim and 1024-dim models on this testbed, and the 2,000-sentence corpus is too small for meaningful recall@ k measurement. Larger-scale multi-model evaluations with retrieval metrics remain important future work. The PCA basis is fitted on a specific corpus and may transfer imperfectly to out-of-distribution data, though we find it generalizes well within the same embedding model.

Not a replacement for native Matryoshka. When retraining is feasible, native MRL training produces models with true Matryoshka properties *without* the overhead of PCA rotation. PCA-Matryoshka is best viewed as a practical alternative when

retraining is impossible—which is the case for most deployed systems using third-party or closed-source embedding models.

Cosine similarity is not a reliable proxy for retrieval quality. This is arguably our most important methodological finding. PCA-256 + TQ3 achieves *lower* cosine similarity (0.963) but *higher* recall@10 (78.2%) than PCA-384 + TQ3 (0.979 cosine, 76.4% recall). Likewise, PCA-384 + TQ3 matches TurboQuant 3-bit on cosine (0.979 vs. 0.978) but trails by 7.4 points on recall (76.4% vs. 83.8%). This disconnect arises because cosine similarity measures per-vector reconstruction fidelity, while recall@10 measures ranking quality—and small perturbations distributed across many vectors can swap the order of closely-ranked items.

Recommendation: When evaluating compression for retrieval applications, always report recall@ k or MAP as the primary metric. Cosine similarity is useful for understanding the compression mechanism but should not be used to predict retrieval quality, especially at compression ratios above $10\times$.

VII. APPLICATIONS

A. RAG Systems with pgvector

The most immediate application of PCA-Matryoshka is in RAG systems backed by PostgreSQL with pgvector. A typical RAG deployment stores 1–10M document embeddings. With BGE-M3 at full float32, 10M vectors require 40GB. PCA-384 + TQ3 reduces this to 1.4GB—a $27\times$ reduction—while maintaining 99.8% recall@10 with $5\times$ oversampling (79% single-stage), enabling deployment on modest hardware. The `turboquant-pro` library provides pgvector-compatible binary serialization and query transformation functions.

B. LLM KV Cache Compression

The key-value caches in transformer-based LLMs present a similar compression opportunity. KV cache vectors are high-dimensional, exhibit low effective rank, and are accessed via inner-product attention. PCA rotation of the key vectors can reduce KV cache memory without retraining the model. The `turboquant-pro` implementation extends PolarQuant with asymmetric K/V bit allocation (exploiting the softmax sensitivity asymmetry between keys and values) and RoPE-aware quantization (protecting low-frequency positional dimensions); see Section VIII.

C. Edge Deployment

For IoT and mobile deployments where both memory and bandwidth are constrained, PCA-128 + TQ3 achieves $78.8\times$ compression, reducing a 1024-dim float32 embedding from 4KB to 51 bytes. This enables on-device semantic search with indices that fit in the limited memory of edge devices.

D. Cross-Lingual Retrieval

On our cross-civilizational ethics corpus spanning 37 languages, PCA-Matryoshka preserves cross-lingual retrieval quality. The principal components capture language-agnostic semantic structure (which dominates the top eigenvalues), while language-specific artifacts tend to occupy lower-variance

dimensions that are discarded during truncation. This makes PCA-Matryoshka particularly well-suited for multilingual applications.

VIII. EXTENSIONS

We briefly report two algorithmic extensions to the compression pipeline that improve quality without changing the core PCA-Matryoshka method.¹

a) Learned codebooks.: The default Lloyd-Max codebooks assume Gaussian-distributed rotated coordinates. Real embedding models deviate from this assumption: after rotation, the tails are heavier than Gaussian. Training codebooks on actual rotated coordinates via Lloyd’s algorithm (1D k -means, 50 iterations) reduces quantization error by 22% at the same 3-bit width. Measured on three models (Table VIII), the learned codebook consistently improves cosine similarity by +0.005 (e.g., 0.978 $\rightarrow$ 0.983), confirming this is a property of the algorithm rather than a model-specific artifact.

b) Eigenvalue-weighted bit allocation.: After PCA, early dimensions explain most variance. Allocating 4 bits to high-eigenvalue dimensions and 2 bits to the tail—guided by cumulative variance thresholds—gives better quality than uniform 3-bit at the same average storage. At 2.8 average bits, eigenweighted quantization achieves 0.962 cosine similarity (vs. 0.958 uniform 3-bit) in 7% less storage.

c) Asymmetric K/V bit allocation (for KV caches).: In transformer attention, scores are computed as $\text{softmax}(QK^\top/\sqrt{d})$: errors in K are amplified by the softmax, making keys more sensitive to quantization than values. Allocating 4 bits to keys and 2 bits to values achieves 0.995 key cosine similarity at *identical* storage cost to uniform 3-bit (K3/V3 at 0.979)—a free quality improvement on the dimension that matters most for attention accuracy.

IX. CONCLUSION

We have presented PCA-Matryoshka, a training-free technique that enables effective dimension truncation for embedding models that were not trained with Matryoshka representation learning. The method is simple—compute PCA on a sample of embeddings, then rotate all vectors before truncating—but the empirical results are striking: on BGE-M3, PCA rotation improves truncation from 0.467 to 0.974 cosine similarity at 256 dimensions, a 109% improvement.

Combined with TurboQuant 3-bit scalar quantization, PCA-Matryoshka achieves $27\times$ compression at 79% recall@10 single-stage. With $5\times$ oversampling and exact reranking (measured on a 50K-vector production subsample, all methods receiving identical treatment), recall rises to 99.8%—while binary quantization at $32\times$ reaches only 85.6% and product quantization only 73.6% under the same conditions. This fills a previously empty region of the compression–retrieval Pareto frontier. With learned codebooks (v1.0.0), pointwise

¹The open-source implementation (`turboquant-pro` v1.0.0) includes additional engineering features—fused CUDA kernels, compressed HNSW indexing, KV cache compression, cross-framework export, production monitoring—documented at <https://github.com/ahb-sjsu/turboquant-pro>.

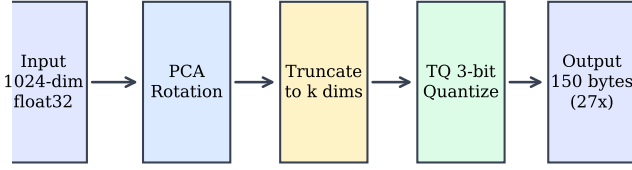


Fig. 5. The PCA-Matryoshka + TurboQuant compression pipeline. (1) A pre-trained PCA rotation matrix (fit once on a corpus sample) rotates input embeddings into the principal component basis. (2) Trailing dimensions are truncated. (3) The remaining dimensions are quantized to 3 bits using TurboQuant. At query time, the query vector undergoes the same rotation and truncation before search.

fidelity improves to 0.983 cosine similarity—a 22% reduction in quantization error at the same bit-width.

A key finding is that cosine similarity to the original vector is not a reliable proxy for retrieval quality at high compression. We recommend that practitioners always evaluate on task-relevant retrieval metrics (recall@ k , MAP) rather than relying on pointwise fidelity.

Our benchmark across seven method families reveals that PCA-Matryoshka + TurboQuant strictly dominates both binary quantization and product quantization across the practical compression range, while requiring no model retraining or access to the original training data.

The implementation is open-source (`pip install turboquant-pro`) and integrates with `pgvector`, `FAISS`, and `vLLM` for immediate deployment.

Future work. One direction merits investigation: nonlinear rotation via a lightweight autoencoder, which could capture additional variance beyond what linear PCA provides.

REFERENCES

- [1] A. Katz, “pgvector: Open-Source Vector Similarity Search for Postgres,” <https://github.com/pgvector/pgvector>, 2023.
- [2] J. Johnson, M. Douze, and H. Jégou, “Billion-Scale Similarity Search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.
- [3] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazé, M. Lomeli, L. Hosseini, and H. Jégou, “The FAISS Library,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- [4] A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sber, P. Jain, A. Farhadi, and S. Kakade, “Matryoshka Representation Learning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 30 233–30 249.
- [5] OpenAI, “New Embedding Models and API Updates,” <https://openai.com/blog/new-embedding-models-and-api-updates>, 2024.
- [6] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Liang, and D. Fang, “BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation,” in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 2548–2569.
- [7] H. Jégou, M. Douze, and C. Schmid, “Product Quantization for Nearest Neighbor Search,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.
- [8] T. Ge, K. He, Q. Ke, and J. Sun, “Optimized Product Quantization,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4. IEEE, 2014, pp. 744–755.

- [9] A. Zandieh, I. Han, M. Daliri, and A. Karbasi, “Sub-linear Memory Inference via PolarQuant and QJL,” in *International Conference on Learning Representations (ICLR)*, 2026.
- [10] J. Yagnik, D. Strelow, D. A. Ross, and R.-S. Lin, “The Power of Comparative Reasoning,” in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2431–2438.
- [11] C. Eckart and G. Young, “The Approximation of One Matrix by Another of Lower Rank,” *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.
- [12] W. B. Johnson and J. Lindenstrauss, “Extensions of Lipschitz Mappings into a Hilbert Space,” *Contemporary Mathematics*, vol. 26, pp. 189–206, 1984.
- [13] D. Achlioptas, “Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins,” *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003.
- [14] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions,” *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.
- [15] D. N. Cardoso, “A generalization of the incremental algorithms for the principal component analysis to the weighted case,” *arXiv preprint arXiv:1501.05709*, 2015.
- [16] B. Varici, C.-P. Tsai, R. Ray, N. M. Boffi, and P. Ravikumar, “Eigenfunction Extraction for Ordered Representation Learning,” *arXiv preprint arXiv:2510.24672*, 2025, carnegie Mellon University.
- [17] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, “SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation,” in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 1–14.