Most MaxSim Winners Flip, Retrieval Survives: Low-Margin Substitution in Sign-Coded Late Interaction*

Anirudh Lakhotia[1], Nischal Helagally Shantharaju[1]

[1]Couchbase, Inc., USA

Abstract

Late-interaction retrieval appears unusually robust to extreme compression. We study a sub-bit sign-coded ColBERT representation that stores each 128-dimensional document token as the 64 signs of a projected vector (0.5 bits per original dimension, 8 B/tok). On a 100k-passage MS MARCO diagnostic slice, sign coding changes the MaxSim winner for roughly 70% of query tokens, yet relevant-document recall remains near the fp32 ceiling and MRR@10 is recoverable by a short fp32 rerank. We find that sign coding substantially disrupts token-level structure while leaving retrieval effectiveness largely intact. We trace this robustness to *low-margin substitution*: although sign coding frequently changes the winning document token, the replacement usually has a similar fp32 score, so the resulting loss in the MaxSim sum remains small. The remaining retrieval loss is therefore concentrated in head ordering among near-tied candidates rather than in losing relevant documents from the candidate set. The mechanism also suggests that projection learning has limited room to help once errors are dominated by low-margin substitutions. Consistent with this prediction, on the full 8.8M-passage MS MARCO corpus a trained projection provides no statistically reliable advantage in MRR@10 or Recall@1000 over a random orthogonal projection at matched storage. These findings suggest that extreme compression for late-interaction retrieval need not faithfully preserve token-level geometry. A simple projection-training-free recipe – a random 64-dimensional projection, 8 B/tok sign-coded document storage, and fp32 reranking of the top 100 candidates – recovers fp32 MRR@10. The resulting representation is a compact storage format that is also cheap to score, complementary to pruning-based retrieval engines. Code and artifacts are available at https://github.com/anirudhlakhotia/subbit-multivector-retrieval.

Keywords

ColBERT, late interaction, MaxSim, sign quantisation, low-margin substitution, ColBERT compression, reproducibility

1. Introduction

Late-interaction retrievers such as ColBERT [1, 2] keep one embedding per document token and score a query–document pair with MaxSim, $s(Q, D) = \sum_i \max_j q_i \cdot d_j$: each query token is compared against every document token, only the largest similarity is kept, and these per query token maxima are summed into the final score. Because each document token carries a 128-dimensional fp32 embedding, the index is large: on the full 8.8M MS MARCO passage corpus it runs to several hundred gigabytes. This storage cost has motivated increasingly aggressive per-token compression, and several production multi-vector systems already offer sign-coded document tokens [3, 4, 5], keeping only the sign of each coordinate and discarding magnitude. We study a variant that goes further, first projecting each document token to r=64 dimensions and keeping only the signs of the projection: 64 sign bits for the 128 original dimensions, or 0.5 bits per dimension (hence sub-bit), 8 bytes per token. This perturbs token-level similarities heavily, yet retrieval effectiveness stays close to the fp32 reference.

On a 100k-passage MS MARCO slice that we use for token-level diagnostics (end-to-end results are confirmed at the full 8.8M scale), this r=64 code changes the document token that achieves the maximum similarity for roughly 70% of query tokens. We measure these winner changes within each query's top-100 fp32 candidate passages, the part of the ranking where retrieval metrics are determined. Most winners change, yet retrieval survives.

ReNeuIR 2026: 5th Workshop on Reaching Efficiency in Neural Information Retrieval, co-located with ACM SIGIR 2026, July 24, 2026, Melbourne, Australia

*Accepted at ReNeuIR 2026, co-located with ACM SIGIR 2026.

anirudh.lakhotia@couchbase.com (A. Lakhotia); nischal.hs@couchbase.com (N. H. Shantharaju)

Many compression methods are designed to preserve or estimate similarity: ColBERTv2 stores centroid ids with quantised residuals [2], and PLAID serves that representation with centroid interaction and pruning [6]; EMVB combines bit-vector filtering with product quantisation [7]; ITQ learns an orthogonal rotation before thresholding [8]; and RaBitQ uses a random rotation with correction terms to estimate distances or inner products with error bounds [9]. The sign code studied here is different: it discards magnitude and uses no reconstruction or correction term during candidate scoring, so individual token similarities can drift substantially from their fp32 values. That retrieval survives anyway suggests that robustness does not require faithful per-token similarities, pointing instead to a property of the MaxSim aggregation itself.

To understand the discrepancy, we examine the document tokens that replace the original MaxSim winners. They are rarely arbitrary: the new winner is usually an alternative whose fp32 score was already close to the original's, so the swap changes that query token's contribution only slightly, and even widespread winner changes move the summed score little. Winners with a clear lead, where a swap would genuinely cost, are rare. We call this *low-margin substitution*; the $r{=}64$ sub-bit regime is the most stringent test of this mechanism that we consider, since it discards both magnitude and half the dimensions.

If low-margin substitution explains why retrieval effectiveness survives, then what appears sufficient is not exact fp32 geometry but enough of the dense high-scoring local neighbourhood that substitutions remain small in fp32 score. Under this view, learning has limited room to help once random sign coding already keeps substitutions within that low-margin score plateau. Consistent with this prediction, at full 8.8M scale a trained projection and a random orthogonal one are statistically indistinguishable in MRR@10, the mean reciprocal rank of the first relevant passage in the top ten (§5).

The practical consequence is a training-free (for the projection) recipe. A random orthogonal projection to $r{=}64$ generates candidates, and a full-precision rerank of the top 100 recovers fp32 MRR@10 at both corpus scales, with no centroid table (§9). On the 100k diagnostic corpus the sign-coded payload reaches fp32-level MRR@10, after the rerank, from a per-token index 2.25–$8.25\times$ smaller than ColBERTv2's residual codec and cheaper to score under a matched exhaustive harness; it is complementary to pruning-based engines like PLAID, not a replacement for them (§9.1).

Contributions. We study sign-coded ColBERT in a sub-bit regime ($r{=}64$, 0.5 bits per original dimension), where each document token is represented only by the signs of a low-rank projection, and we ask what survives, why, and whether the projection must be learned.

- **The phenomenon.** On MS MARCO, sign coding changes the MaxSim winner for a majority of query tokens, yet relevant-document recall stays near the fp32 reference and MRR@10 is recovered by reranking; across nine out-of-domain (for the encoder) BEIR collections, sign coding retains most of fp32's NDCG@10. Most winners change, yet retrieval survives (§6).
- **The mechanism.** Most winner changes are low-margin substitutions between near-tied document tokens, so each query token's contribution barely moves. We characterise how these substitutions add up: query scores stay stable and relevant documents stay in the candidate set despite widespread winner churn (§7).
- **Learned and random projections converge at scale.** At full 8.8M MS MARCO scale a learned low-rank projection is indistinguishable from a random orthogonal one in MRR@10 and nearly identical in Recall@1000 (§5).
- **The recipe.** A training-free (for the projection) sub-bit pipeline using a random orthogonal projection with a full-precision rerank of the top candidates reaches fp32-level MRR@10 (§9) and, on the 100k diagnostic corpus, fp32-level MRR@10 from a 2.25–$8.25\times$ smaller per-token index than ColBERTv2's residual codec, complementary to pruning-based engines (PLAID), not a replacement for them (§9.1).

2. Related work

Work on late-interaction retrieval cuts cost in three ways: storing fewer vectors, compressing each vector, or serving compressed representations more efficiently. Each asks whether retrieval quality survives compression. Less direct work asks which token-level MaxSim properties must be preserved under aggressive representation compression, especially when compression changes the winning document token.

Efficient late-interaction retrieval. Several systems compress and serve the centroid index. ColBERTv2 compresses token embeddings into centroid ids with quantised residuals [2]; PLAID accelerates search over that representation with centroid interaction and pruning [6]. EMVB adds bit-vector prefiltering and uses product quantisation in place of PLAID's residual compressor [7]. WARP serves XTR-trained retrievers through implicit decompression that avoids vector reconstruction [10]. TACHIOM scales token-aware centroid allocation to millions of centroids and scores documents from those centroids directly [11]. This line improves how token representations are indexed and searched. It validates end-to-end quality at each approximation stage, but does not isolate which token-level interactions a ranking depends on.

Vector-count reduction. A separate line of work reduces the number of document vectors. Some methods prune tokens: a study of token-dropping strategies showed that ColBERT passage indexes can be pruned by up to 30% at small cost [12], Zong and Piwowarski regularise for score preservation while keeping about 30% of document tokens [13], and static low-IDF token removal [14] and a Voronoi-cell influence criterion [15] thin the index. Others produce fewer vectors by construction: ColBERTer learns whole-word representations and drops vectors that do not contribute to scoring [16], ConstBERT fixes a constant number of vectors per document [17], query-agnostic budgeting [18] and CRISP's clustering [19] cut it further, token pooling merges similar document vectors [20], and MUVERA collapses the multi-vector set into a fixed-dimensional encoding with random space partitions for candidate generation, reranking candidates with exact MaxSim [21]. Where these methods change how many vectors are scored, we keep the token structure intact and compress each token's representation.

Token compression and learned projections. Two families compress the individual vector. Learned codes minimise reconstruction or ranking error: trained codebooks in PQ, learned rotations in ITQ and OPQ, and supervised codes in BPR and JPQ [22, 8, 23, 24, 25]. A data-oblivious family instead draws random projections independent of the data. SimHash [26] and the broader random-projection and hashing literature [27, 28, 29] need no training data, and RaBitQ [9] pairs a random orthogonal rotation with binary data codes and stored per-vector correction terms, yielding an unbiased distance/inner-product estimator with high-probability per-pair error bounds rather than a rank-order guarantee. CoRECT shows that in single-vector retrieval such non-learned compression scales to 100M passages with no statistically significant quality loss [30]; whether this carries to token-level MaxSim is open. Closer to our setting, Clavié et al. train ColBERT variants with alternative projection blocks and evaluate them under PLAID indexing with 4-bit residuals, reporting NDCG@10 gains on TREC-DL19 and three of four out-of-domain BEIR subsets [31], comparing variants within a learned-projection family. We instead study sign codes derived from low-rank projections, including fully random ones. Our goal is not to improve similarity preservation but to understand which aspects of MaxSim stay necessary once individual similarities are heavily perturbed. To our knowledge, no prior work directly compares trained and random projections for sign-coded late-interaction retrieval across both the sub-bit low-dimensional setting ($r{=}64$) and the full-dimensional one-bit setting ($r{=}128$). ITQ performed the analogous comparison for single-vector binary codes [8]; we perform it for late interaction at corpus scale (§5).

Reranking and compressed serving. One-bit token representations for late interaction are already well established in both research and production: ColBERTv2's b=1 mode [2] and Bi-ColBERT [32] on the research side, and deployed systems on the production side. Vespa and Elasticsearch sign-code per-token ColBERT and ColPali document vectors to 1 bit per dimension, 32× smaller than fp32, scoring MaxSim in Hamming space or asymmetrically against full-precision queries [3, 33, 4]. Qdrant reports binary quantisation for ColPali document vectors as a direct compressed scoring path, with faster search than scalar quantisation at similar accuracy [5].

These systems mostly use binary or one-bit document-token representations; we study a more aggressive regime that first projects each document token to a lower-dimensional space and then sign-codes it. Rescoring is itself a standard two-stage pattern: BPR retrieves candidates by Hamming distance over binary passage codes and rescores them with the continuous query embedding [22], and cascade rankers rerank a cheap first-stage candidate set with a stronger model [34, 35]. The closest comparison point is SPLATE, which matches PLAID quality on ColBERTv2 by rescoring the top 50 candidates from a SPLADE-style sparse adapter [36]. Our pipeline follows the same shape, using low-rank sign-coded MaxSim for candidate generation and fp32 MaxSim for reranking. These systems show that aggressive compression can preserve a useful candidate set when followed by a higher-precision reranker. Why that candidate set survives is the question we take up.

Understanding MaxSim. MaxSim decomposes into one argmax per query token, and prior analyses have studied how those matches are formed and which tokens carry the score. XTR, the retriever WARP serves, retrains the encoder so that token retrieval surfaces the document tokens that matter for scoring, letting documents be ranked from retrieved tokens alone [37], and a matching analysis traces the score to co-occurring, high-importance token pairs [38]. Weighted Chamfer scoring adds IDF-based or few-shot-trained token weights to the sum over query tokens [39], and an empirical analysis characterises MaxSim's length bias and the negligible signal beyond the top-1 document token [40]. Prior analyses use this decomposition to ask which tokens matter and how matches are formed. We use it for a different question: why compressed MaxSim rankings can remain effective even when many per-token argmaxes change. We show that, in our compressed setting, this robustness is largely explained by low-margin substitution.

3. Method

3.1. Representation

Let $d \in \mathbb{R}^{128}$ be a ColBERTv2 document token. The sign-coded encoding is

$$\mathrm{enc}_D(d) \;=\; \mathrm{sign}(R\,d) \;\in\; \{-1,+1\}^r, \tag{1}$$

where $R \in \mathbb{R}^{r\times 128}$ is a fixed projection; the resulting sign code uses $r/8$ bytes per token. We compare three choices of R at the same byte budget per token: *identity truncation* ($R = [I_r \mid 0]$), *random orthogonal* (a fixed row-orthonormal matrix, $RR^\top = I_r$ with $r < 128$, drawn once at a given seed), and *trained* (learned using the objective in §3.3). The three variants share this byte budget and differ only in R (each optionally carrying the same query-side scale head, §3.3, which adds no per-token storage). Most of the analysis follows r=64.

3.2. Retrieval scoring

Documents are stored as sign codes, but queries stay in fp32 (asymmetric scoring): $\mathrm{enc}_Q(q) = R\,q \in \mathbb{R}^r$ is a float query projection dotted against the binary document codes. The score is the standard MaxSim, which matches each query token to its single best document token and sums those matches,

$$s(Q,D) \;=\; \sum_{i=1}^{m} \max_{j=1,\dots,n} \mathrm{enc}_Q(q_i)\cdot \mathrm{enc}_D(d_j), \tag{2}$$

computed on the stored sign codes. Scoring optionally includes a small learned query-side scale head: a scalar gain $S(q) \in [1, 1.5]$ computed from the 128-dim query token (129 parameters; §3.3) that multiplies the projected query token. It is query-side, so it adds no per-token storage and can accompany any choice of R; the headline comparison (Table 1) gives all three the same scale-head architecture and training opportunity, isolating R. We use this asymmetric scoring throughout.

The projection R is shared across all tokens and stored once for the entire index.[1]

We take r=64 (0.5 bits/dim, 8 B/tok) as the primary operating point and also report r=128 (1 bit/dim, 16 B/tok).

At evaluation the ColBERTv2 encoder is frozen and R is held fixed (at the seed, the identity, or the trained checkpoint). Scoring is therefore ordinary MaxSim: the sign code changes only how document tokens are represented, not how their scores are aggregated.

3.3. Training objective

The trained variant learns a projection R that approximates selected fp32 MaxSim scores of the original model under sign coding, training on (query, positive, negative) triples with the ColBERTv2 encoder frozen. The score a query assigns a document is determined by the few document tokens that compete to be each query token's best match. The objective therefore has three components: a top-K anchor that preserves those competitors, a guard that discourages sign coding from introducing spurious new winners, and an orthogonality regulariser that keeps the projection well-conditioned.

Formally, for a query token q_i and document token d_j, let $s^T_{ij} = q_i \cdot d_j$ denote the teacher (fp32) score and $s^S_{ij} = r^{-1/2}\,\mathrm{enc}_Q(q_i) \cdot \mathrm{enc}_D(d_j)$ the sign-coded student score (the $r^{-1/2}$ factor removes the growth of the sign-vector norm with r and gives the squared-error terms below a comparable scale; as a global constant it leaves ranking unchanged, so the scorer in Eq. 2 omits it). Let $\mathcal{T}^K_i = \mathrm{argtop}^K_j\, s^T_{ij}$ be the teacher's K highest-scoring document tokens for q_i.

The *top-K anchor* matches the student's scores to the teacher's on $\mathcal{T}^K_i$, the document tokens most likely to win each query token's MaxSim. Because MaxSim depends only on the best-matching document token, these competitive tokens are a natural place to focus the distillation:

$$\mathcal{L}_{\text{topk}} \;=\; \frac{1}{m}\sum_{i=1}^{m}\sum_{j\in\mathcal{T}^K_i}\left(s^S_{ij} - s^T_{ij}\right)^2; \tag{3}$$

we use K=5.

The *guard* discourages a previously uncompetitive token from becoming the student's winner solely because sign coding perturbs the scores: it anchors the strongest such outsider back to its teacher score,

$$\mathcal{L}_{\text{guard}} \;=\; \frac{1}{m}\sum_{i=1}^{m}\left(s^S_{ij^*_i} - s^T_{ij^*_i}\right)^2, \quad j^*_i = \arg\max_{j\notin\mathcal{T}^K_i} s^S_{ij} \text{ (stop-grad)}. \tag{4}$$

The *orthogonality regulariser* discourages R from collapsing several coded dimensions onto the same direction: $\mathcal{L}_{\text{ortho}} = \|RR^\top - I_r\|^2_F$.

Each term is averaged over the positive and negative document of a training triple, written $\mathcal{L}^{\pm}$. The composite objective is

$$\mathcal{L} \;=\; \lambda_{\text{topk}}\,\mathcal{L}^{\pm}_{\text{topk}} + \lambda_{\text{guard}}\,\mathcal{L}^{\pm}_{\text{guard}} + \lambda_{\text{ortho}}\,\mathcal{L}_{\text{ortho}}, \tag{5}$$

with $\lambda_{\text{topk}} = \lambda_{\text{guard}} = 1$ and $\lambda_{\text{ortho}} = 0.001$. The scale head produces a single multiplier in $[1, 1.5]$, a function of the 128-dim query token, that rescales the projected query token (129 parameters). This trained R and its scale head are used for the rows labelled trained "+ scale"; the mechanism analyses (§6) and the rerank recipe (§9) use R without the scale head, and no-scale and frozen-R controls are labelled separately.

[1] R costs $r \times 128 \times 4$ bytes once (32 KiB at r=64), under 5×10^{-3} B/tok beyond ∼100k passages.

4. Experimental setup

On MS MARCO we compare trained, random, and identity projections at matched storage budgets. On BEIR we use identity vs. random as a non-trained rotation-invariance control, and fp32 to compare retention.

Query convention. We use standard ColBERTv2 augmented queries throughout: the 100k diagnostic slice, the 8.8M corpus, and BEIR all follow this convention.

Scale	Use
50k triples	training budget
100k passages	diagnostic analysis
8.8M passages	full-corpus evaluation

Training. R and the 129-parameter scale head are trained with AdamW (learning rate 10^{-3}, weight decay 10^{-4}), batch size 32, gradient clipping at 1.0, and a cosine schedule with 250 warmup steps over 25k steps (best checkpoint near step 15k). R is trained on 50,000 (query, positive, negative) triples – 10,000 query–positive pairs from the MS MARCO passage-train qrels restricted to the 100k-passage training subset, each paired with five non-qrel negatives drawn uniformly at random from the same subset (seed 42), with no mined hard negatives. We train at seeds 42, 43, and 44 on the 100k diagnostic slice and report diagnostic results as the mean across them; the full 8.8M evaluation uses seed 42. In the 8.8M comparison the random baseline uses a separately trained scale head with R frozen to a random orthonormal matrix. Training and the 100k diagnostic evaluation run on an Apple M4 Pro (48 GB); the full 8.8M retrieval evaluation runs on a single NVIDIA A100-80GB.

Evaluation. We evaluate on MS MARCO Passage dev queries (6,980) against the full 8.8M corpus and the 100k diagnostic slice, and on nine BEIR domains [41]: nfcorpus, scifact, arguana, scidocs, fiqa, trec-covid, webis-touche2020, quora,[2] and nq. The four larger Wikipedia-derived corpora (hotpotqa, dbpedia-entity, fever, climate-fever) were excluded because they exceeded our encoding budget. The encoder is the public `colbert-ir/colbertv2.0` checkpoint. All first-stage retrieval is exact. Asymmetric MaxSim between float queries and sign-coded documents is computed exhaustively over all 8,841,823 passages, with no ANN index or candidate pruning. The PLAID rows in Table 1 isolate PLAID's centroid+b-bit residual *codec* (C=32,768): we apply it to the shared frozen embeddings and score it with the same exhaustive MaxSim harness as every other row, dropping PLAID's candidate generation and centroid pruning so that the compression scheme is the only factor varying across rows. Run as a full engine over the same passages (independently re-encoded), official PLAID agrees with our codec within 0.004 MRR@10 (0.8593 codec vs. 0.8631 engine, Table 8). The ITQ, PQ, OPQ, and RaBitQ rows are likewise our own implementations on this dev slice. Two-stage rows (§9) rerank the exact top-K candidates ($K \in \{100, 1000\}$) with full-dimensional fp32 MaxSim. We report r=64 (0.5 bits/dim) as the primary operating point and r=128 (1 bit/dim) alongside it. We report MRR@10 and Recall@$\{100, 1000\}$ on MS MARCO and NDCG@10 on BEIR; statistical uncertainty on MS MARCO is a per-query paired bootstrap over the 6,980 dev queries (10,000 resamples). Reported storage counts include only the per-token payload and exclude amortised projection matrices and centroid codebooks.

5. Does learning the projection matter?

The central question left open by Section 3 is whether learning the projection improves retrieval quality. Table 1 shows that on the full 8.8M MS MARCO corpus it does not: a trained projection and a random

[2] Quora is evaluated on its dev split (5,000 queries), following the standard BEIR convention inherited from prior ColBERT-family work. The other eight corpora use their test splits.

Table 1
Storage–quality across compression methods on MS MARCO dev. **Top:** 100k diagnostic slice. **Bottom:** full 8.8M corpus, asymmetric scoring. The matched-byte comparison is trained R vs. random orthogonal R at r=64, 8 B/tok, where the two are statistically indistinguishable. PLAID, ITQ, PQ, OPQ, and RaBitQ are our own implementations (§4); storage is per-token ([†]RaBitQ 24=16+4+4 B; [‡]fp32 8.8M is a reference from a different harness). At matched bytes, learning the projection yields no measurable advantage at corpus scale.

Method	B/tok	MRR@10	Rec@100	Rec@1000
100k MS MARCO dev				
fp32 ColBERTv2	512	0.8641	0.994	0.9989
PLAID b=4 [6]	66	0.8627	0.994	0.9989
PLAID b=2	34	0.8593	0.993	0.9988
PLAID b=1	18	0.8498	0.993	0.9987
ITQ r=64, asym [8]	8	0.8319	0.9871	0.9958
PQ 8×8 [23], ADC	8	0.8330	0.9867	0.9954
OPQ 8×8 [24]	8	0.8048	0.9802	0.9947
RaBitQ [9], asym	24[†]	0.8582	0.9925	0.9982
Trained R, r=64 asym	8	0.8496	0.991	0.9972
Random orthogonal R, r=64 asym	**8**	**0.8393**	0.990	0.9963
Reference (different evaluation harness)				
fp32 ColBERTv2[‡] [2]	512	0.397	—	—
Full 8.8M MS MARCO dev, asymmetric scoring				
RaBitQ	24[†]	0.390	0.909	0.982
ITQ r=64	8	0.369	0.889	0.973
headline: trained vs. random vs. identity, r=64 + scale (8 B/tok)				
Trained R, r=64 + scale	**8**	**0.3723**	0.900	0.975
Random orthogonal R, r=64 + scale	**8**	**0.3724**	0.892	0.973
Identity R, r=64 + scale	**8**	**0.3735**	0.894	0.975
control: r=128 + scale (16 B/tok), only R changes				
Trained R, r=128 + scale	16	0.3889	0.912	0.982
Random orthogonal R, r=128 + scale	16	0.3916	0.911	0.983
Identity R, r=128 + scale	16	0.3921	0.910	0.983

orthogonal projection perform nearly identically at matched storage. We then examine a smaller diagnostic slice and a cross-domain BEIR panel to understand the scope of this result.

Two comparisons recur, and they are distinct. We report trained-vs-random on the 100k diagnostic slice and at full 8.8M scale; BEIR uses identity/sign vs. random as a separate non-trained rotation-invariance control.

Full 8.8M corpus. At r=64, training the projection provides no statistically reliable advantage over a random orthogonal one. The two nearly tie on MRR@10 and Recall@1000 (Δ MRR@10 $= -0.0001$; Table 1). Training slightly raises Recall@100 ($+0.0087$): more relevant documents enter the top-100 candidates, but this does not translate into an MRR@10 gain. This 8.8M comparison uses a single training seed (42; the 100k diagnostics average seeds 42–44), so we read it as the absence of a reliable advantage at this budget rather than as proof of exact equality.

100k diagnostic slice. The diagnostic slice shows a different picture. Here the trained projection leads the random orthogonal baseline by about one MRR@10 point (0.8496 vs. 0.8393; Figure 1), with the sign code scored directly, before any fp32 rerank (§9).[3] The advantage is real in this smaller setting (per-query paired bootstrap $+0.0100$, 95% CI $[+0.0055, +0.0144]$; §8), which makes its absence at full scale a phenomenon to explain rather than a training failure.

[3]The query-side scale head is not load-bearing: removing it changes trained R by $+0.0003$ MRR@10 (within noise) and nothing for random R; the plain sign code suffices.

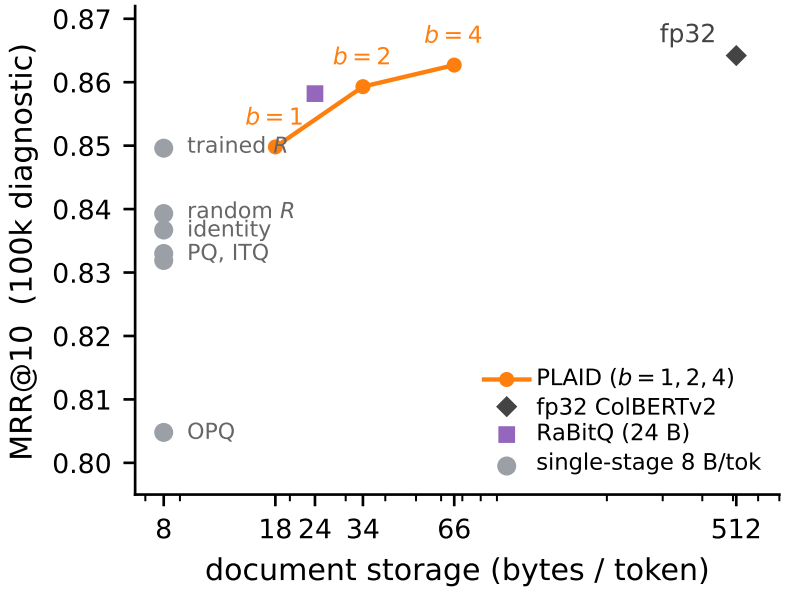

Figure 1: Single-stage storage–quality on the 100k diagnostic slice (MRR@10 vs. document bytes per token, log scale; values in Table 1). At 8 B/tok the sign codes (trained R, random R, identity) and the matched-byte single-vector baselines (PQ, ITQ, OPQ) trail the larger PLAID residual codes, RaBitQ, and fp32 ColBERTv2 in quality, at substantially lower storage. Trained and random R are nearly indistinguishable.

Table 2
Per-corpus NDCG@10 at r=64, 8 B/tok on nine BEIR domains: fp32 ColBERTv2, raw sign(d), and a random orthogonal R. Δ = sign(d)− random (positive favours sign(d)); ret. = sign(d)/fp32. The sign(d) vs. random comparison is the rotation-invariance control of two non-trained projections, not the trained vs. random comparison of Table 1. Sign coding retains a mean 93.8% of fp32 NDCG@10 (random rotation 93.3%), and the choice of rotation has little effect; retention above 100% (webis-touche2020) reflects small-n variance. NQ's fp32 baseline is omitted for compute ("−"); per-query NDCG@10 arrays are published for paired-bootstrap CIs.

Corpus	n_q	fp32	sign	rand	Δ	ret.
scifact	300	0.6464	0.6308	0.5988	**+0.032**	97.6%
trec-covid	50	0.8007	0.7841	0.7693	+0.015	97.9%
fiqa	648	0.3473	0.2999	0.3145	−0.015	86.4%
webis-touche2020	49	0.2697	0.2770	0.2659	+0.011	102.7%
quora	5000	0.8525	0.8239	0.8278	−0.004	96.6%
scidocs	1000	0.1581	0.1403	0.1420	−0.002	88.7%
nq	3452	−	0.5327	0.5312	+0.002	−
nfcorpus	323	0.3301	0.3125	0.3140	−0.001	94.7%
arguana	1406	0.3336	0.2871	0.2880	−0.001	86.0%

Other settings. The same insensitivity holds beyond r=64 and beyond MS MARCO. At r=128 the full-corpus point estimates likewise show no trained edge: trained R is −0.0027 MRR@10 relative to random orthogonal R (Table 1). The paired bootstrap available for the separate identity-versus-random rotation control also includes zero (Δ MRR@10 = +0.0033, 95% CI [−0.0015, +0.0082]). Across nine BEIR domains the differences between sign(d) and a random orthogonal projection are small and inconsistent (median $|\Delta$ NDCG@10$|$ = 0.004; Table 2). SciFact is the only corpus with a statistically reliable difference (+0.032, Holm-adjusted p = 0.005). Relative to the fp32 ColBERTv2 baseline, sign coding is not free but is cheap and, again, projection-insensitive: across the eight domains with an fp32 run it retains a mean 93.8% of NDCG@10, and the random rotation retains 93.3%, so the choice of projection barely moves a roughly 6% quantisation cost.

Together, the full-corpus MS MARCO result and the BEIR panel show that once document tokens are

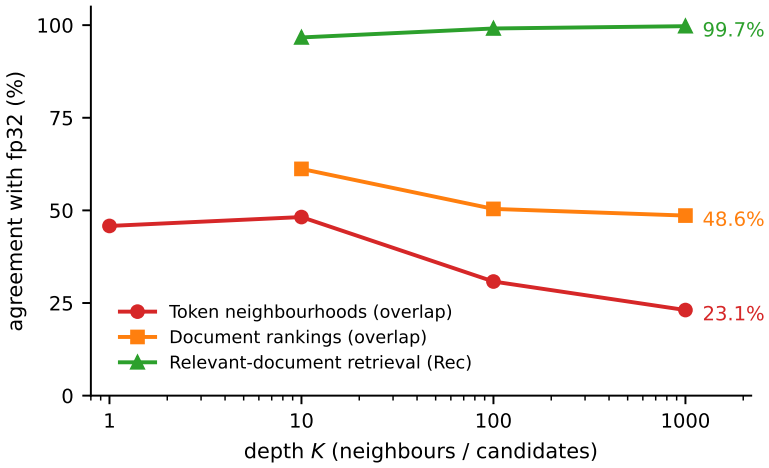

Figure 2: Agreement with fp32 at three levels (r=64, 100k diagnostic slice): token-neighbourhood overlap (random orthogonal R), document-ranking overlap (random orthogonal R), and relevant-document recall (trained R). Degradation decreases up the levels: token geometry is most affected, document ranking less so, and recall remains near the ceiling.

reduced to sign codes, retrieval quality is largely insensitive to the choice of projection. Yet this code changes the document token that wins the MaxSim for most query tokens. MaxSim winners change, while retrieval metrics remain comparatively stable. We turn next to what sign coding changes in the representation, and what it leaves intact.

6. What sign coding preserves and what it does not

Section 5 showed that retrieval quality survives sign coding even though the MaxSim-winning document tokens change. The natural account is that sign coding largely preserves the underlying representation, so retrieval proceeds on nearly the same geometry as fp32. We test that account by measuring preservation at three levels: token geometry, measured as nearest-neighbour identities among document tokens; document rankings, measured as the MaxSim-induced ordering of documents for a query; and retrieval quality, measured as whether the relevant document is retrieved (Figure 2). These levels are preserved very unevenly: token geometry is heavily disrupted, document rankings substantially changed, and only retrieval quality is nearly intact. We report random-orthogonal results at the geometry and ranking levels because these measurements are largely insensitive to the projection (§5, Table 3). All three levels are measured on the 100k diagnostic slice.

Token geometry. Start at the bottom, with the geometry retrieval is built from. A sub-bit sign code can keep the broad directional structure of the embedding space while losing the fine ordering among the many tokens that sit close together; if retrieval quality depends on preserving local token structure, substantial damage at this level should be reflected higher up. We sample 1000 anchor doc-tokens from the 100k diagnostic corpus and compute each anchor's top-k neighbours under fp32 and under an r=64 random-orthogonal sign code, over a pool of 200k doc tokens. The overlap is only about half at small k (48.2% at k=10) and falls to 23.1% at k=1000 (Figure 2). That is far above the 0.5% random baseline, so the code is not noise – yet more than half of fp32's nearest-neighbour identities are already gone at k=10. Token geometry is substantially disrupted.

Document rankings. A document's ranking is not a property of any single token neighbourhood but of the aggregate MaxSim score over all its tokens, which we therefore measure separately. The disruption is milder here but still large: only roughly half of fp32's top-1000 documents survive sign coding (random orthogonal R), against under a quarter of its token neighbours (Figure 2), with moderate

Table 3
Agreement between sign-coded and fp32 document rankings per query, on the 100k diagnostic MS MARCO dev slice, r=64 asymmetric scoring. Pearson is computed over the per-pair query–document scores; Spearman, Kendall, and overlap@K compare the rankings the two scorers induce (fp32 ColBERTv2 is 1.000 by construction). Scores remain strongly correlated while the rankings agree only moderately: sign coding preserves the approximate order, not the exact one.

Projection	Pearson	ρ	τ	ovlp@10	ovlp@100	ovlp@1000
fp32 ColBERTv2	1.000	1.000	1.000	1.000	1.000	1.000
Trained R, asym	0.788	0.740	0.550	0.657	0.567	0.558
PCA r=64	0.752	0.719	0.529	0.619	0.531	0.534
Random orth. r=64	0.692	0.645	0.465	0.612	0.504	0.486
Identity r=64	0.670	0.635	0.456	0.609	0.498	0.480

rank correlations (Spearman ρ 0.63–0.74). Table 3 reports the breakdown across projections: top-K overlap runs 0.61–0.66 at K=10, drops to 0.48–0.57 at K=100, and changes little thereafter. Document rankings are substantially altered; the representation is not faithfully preserved at this level either.

Retrieval quality. Even a changed ranking need not change retrieval quality. Document-ranking agreement is only a surrogate: what the candidate set must preserve is not fp32's order but the relevant document's presence, and we measure that directly (head order, which MRR@10 depends on, is restored later by reranking, §9). At r=64 asymmetric, trained R reaches Rec@1000 = 0.9972 against fp32's 0.9989 (Table 5) – the relevant document is nearly always retrieved, even though only 56% of fp32's exact top-1000 recurs.

What the ranking damage changes is the *rank* of the relevant document, not its presence. Among the worst 5% of queries by MRR@10 drop (349 of 6,980, mean shift −0.527), 322 (92.3%) still hold the relevant document inside the top-10 at a worse rank; only 27 lose it from the top-10 entirely.

These measurements rule out a simple preservation account. Sign coding substantially alters local token geometry and changes many document rankings, yet relevant-document retrieval remains almost unchanged. The explanation therefore cannot be that sign coding faithfully preserves the representation. We next examine the retrieval mechanism itself (§7).

7. Why retrieval survives

§6 established the puzzle: sign coding substantially disrupts token geometry and alters many document rankings, yet retrieval quality remains nearly intact. We therefore examine how the MaxSim winners change under sign coding and what score loss those changes incur. All measurements in this section use the 100k diagnostic slice unless marked otherwise.

Most winners change. MaxSim assigns each query token a winner – the document token achieving the maximum inner product – and the per-query score (Eq. 2) is the sum of those winning inner products. Sign coding substitutes a different representative for each document token, so winners can change; this happens most readily for q-tokens whose fp32 winner holds only a slim lead over the runner-up. We call this fp32 lead the *margin* and mark a q-token as low-margin when it falls below 0.2. On the 6,980 dev queries against the 100k diagnostic corpus, low-margin q-tokens flip their argmax at a 71.30% rate; only 28.70% survive. The high-margin tail (margin $\geq$ 0.2) is only 0.50% of all q-tokens (Figure 3). With most winners changing, whether retrieval survives comes down to how much score each replacement gives up.

Most replacements are near-equal in score. The clue is in what "low-margin" means for the document side. When a query token's fp32 winner holds only a small lead, many document tokens sit

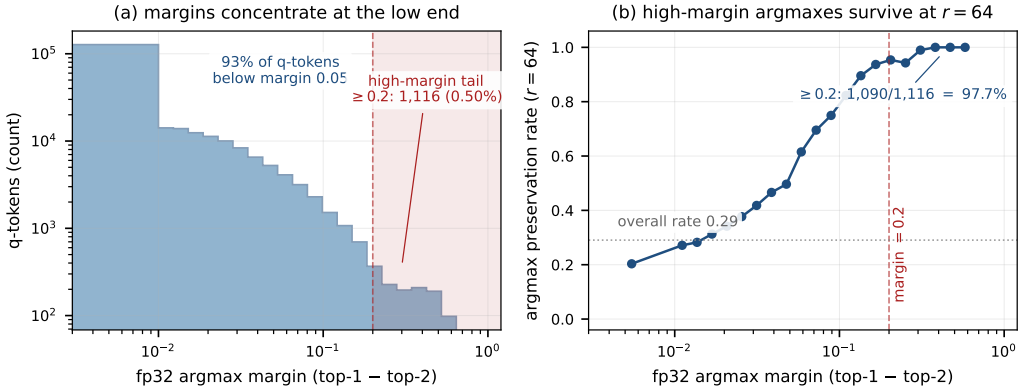

Figure 3: Argmax margin and its preservation under sign coding (r=64, 100k diagnostic slice, all 6,980 MS MARCO dev queries). **Left:** per-query-token margins concentrate at the low end, and the high-margin tail (≥ 0.2) is a small fraction of q-tokens. **Right:** high-margin winners almost always retain their fp32 argmax under sign coding; flips occur among the low-margin near-ties, where the score cost is small.

Table 4
Rank of the substitute winner under fp32 scoring when a flipped low-margin q-token's sign-coded argmax differs from its fp32 argmax (158,452 flipped tokens, r=64, 100k diagnostic slice). The substitute is close to the original in fp32 *score* but widely dispersed in fp32 *rank*: rarely the runner-up, often beyond rank 10. The low-margin head is a dense plateau of near-equal tokens, not a short list of close runners-up.

Substitute fp32 rank	Share of flipped tokens
2	18.9%
3–5	25.4%
6–10	17.0%
Beyond 10	38.7%
Median rank	7

close behind it in fp32 score. The relevant question is therefore not where the substitute lands in rank, but how far it sits from the original winner in score. We measure that cost directly.

Table 4 shows that substitute winners are not concentrated among the top few fp32 runners-up. Among the 158,452 flipped low-margin q-tokens, only 18.9% of substitute winners ranked second under fp32; 38.7% ranked beyond tenth, and the median rank of the substitute is 7. The dense plateau of near-equal-score tokens produces a distribution of substitute ranks that is spread across the head, not clustered near the top. Ordinal rank is therefore a misleading lens: the substitute is not near the winner in rank, but it is near the winner in fp32 score, and it is that score difference – not the rank gap – that determines the substitution cost.

The score loss is tiny. The aggregate residual is the central measurement. For each flipped low-margin q-token, δ_i is the fp32 score of the sign-coded winner minus the fp32 score of the fp32 winner – negative by construction, and near zero when the substitution costs almost nothing in fp32 units. Among the 158,452 flipped low-margin q-tokens the median δ_i is -0.0353, and 62.0% lie within 0.05 fp32-score units of zero (Figure 4, left). Summed per query, the median aggregate loss is 4.2% of the fp32 MaxSim sum, and 98.6% of queries lose at most 10% (Figure 4, right): most low-margin argmaxes flip, yet the median query loses only 4.2% of its MaxSim score.

The relevant quantity is therefore not argmax preservation but the size of the score residual: most winners change, yet the score they contribute changes very little. We propose this small residual as the link between the token-level disruption of §6 and the near-intact relevant-document recall measured there (0.9972 vs. fp32 0.9989); we report the association, not a controlled causal test.

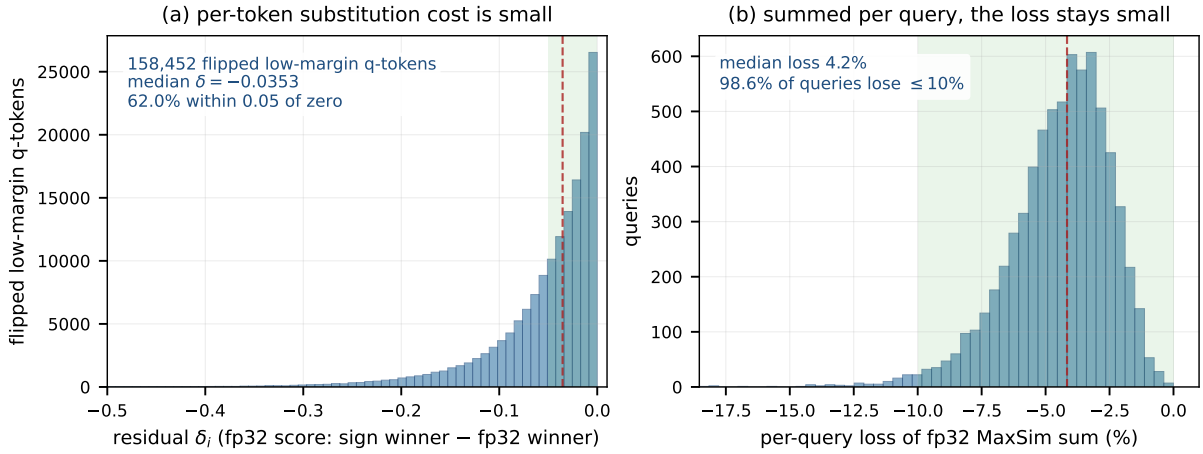

Figure 4: Direct score residual of flipped low-margin q-tokens (r=64, 100k diagnostic slice). **Left:** for each q-token, δ_i is the fp32 score forfeited by selecting the sign-coded winner rather than the fp32 winner ($\delta_i \leq 0$ by construction), almost always a small loss. **Right:** aggregated per query, the loss is a small fraction of the MaxSim total, with nearly all queries within a few percent. The substitution cost is small per token and remains small after aggregation.

7.1. Where the MaxSim sum lives

The mechanism rests on the low-margin bulk carrying the score, so the decisive check is where the per-query MaxSim sum lives: on that bulk, or on the small high-margin tail whose argmaxes do survive sign coding.

On the 6,980 dev queries against the 100k diagnostic slice at r=64, high-margin q-tokens are 0.50% of the pooled population (median query carries zero) and contribute 0.51% of the pooled per-query MaxSim sum (median 0.0%). The sum sits on the low-margin bulk.

The high-margin tail is also where preservation is strongest: those q-tokens keep 97.67% of their fp32 argmaxes and 98.05% of their sum contribution under sign coding. But that stability acts on only a 0.51% slice. The low-margin bulk that carries the rest flips at 71.30%, and its per-token residuals stay near zero (62.0% within 0.05 fp32-score units; Figure 4). The picture is therefore a local one: retrieval survives not because argmaxes are preserved, but because the cost of replacing them is usually small.

Why projection choice barely matters. Because the mechanism turns on the fp32 margin distribution – a property of the embeddings, not of any projection – projection choice has little to act on: the high-margin tail is rare under identity, random, and trained R alike. We measure the low-margin residuals only for the random projection on the 100k slice. The near-ties across all three projections at 8.8M (§5) are consistent with the same low-margin regime holding for the other projections and at full scale, though we do not re-measure the decomposition there.

Second-encoder check. A second encoder provides a limited direction check on the mechanism. ConstBERT$_{32}$ [17], a ColBERT-style multi-vector model with a fixed C=32 document vectors, has a somewhat thinner high-margin tail than ColBERTv2 in our 100k diagnostic probe: its fraction of per-q-token argmaxes at margin ≥ 0.2 is 0.35%, compared with ColBERTv2's 0.50%. Applying the same random orthogonal R at r=64, ConstBERT retains 93.2% of its fp32 MRR@10 (0.858 → 0.800). At lower r the reported gap between the two encoders widens: ColBERTv2 retains 0.707 at r=32 and 0.388 at r=16 (84.9% and 46.6% of its fp32 reference), while ConstBERT falls to 0.628 (73.2%) and 0.222 (25.9%). This is consistent with the margin-distribution account, but with only two encoders it is a sanity check rather than a predictive law. The 8.8M experiment was not re-run on ConstBERT.

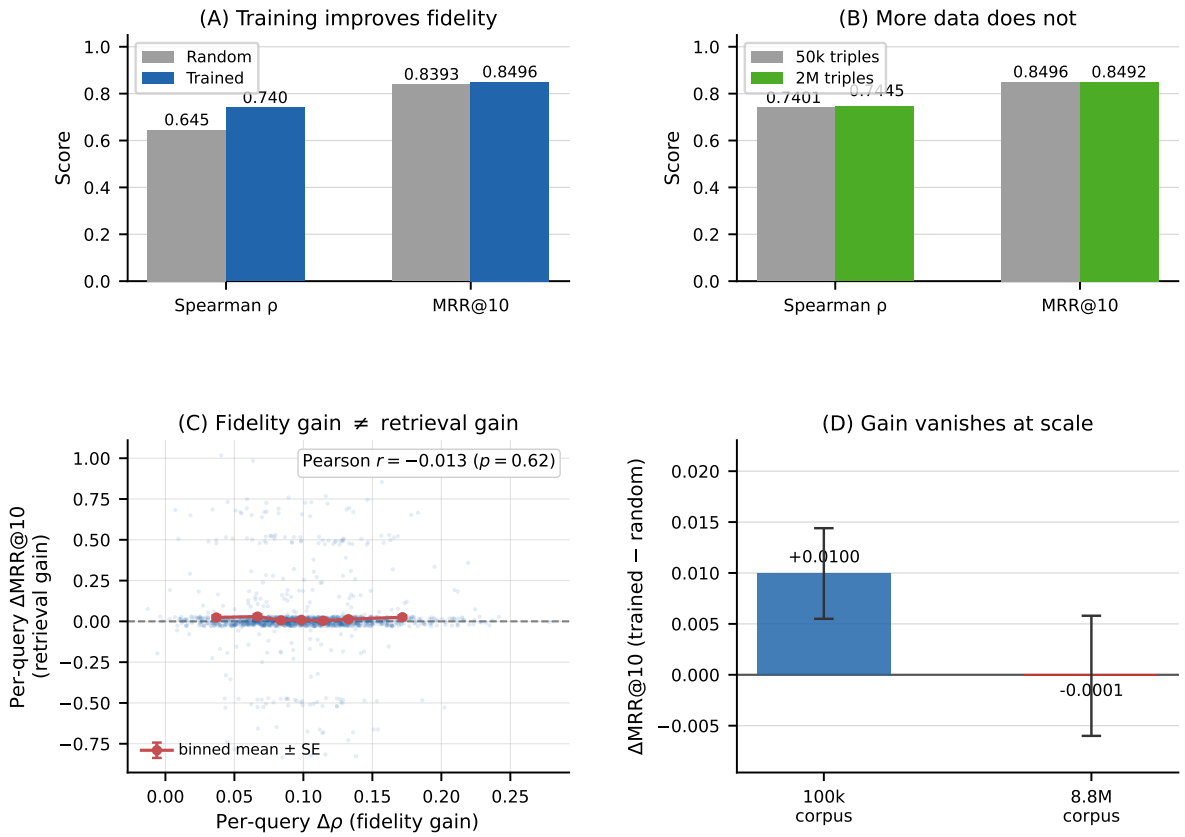

Figure 5: What training changes, and what it does not (r=64, 100k diagnostic slice). **(A)** A trained R agrees with fp32 rankings more closely than a random orthogonal R (Spearman ρ and MRR@10). **(B)** The rank-order fidelity does not grow with $40\times$ more training data (50k vs. 2M triples). **(C)** Across a 1,500-query sample the per-query fidelity gain $\Delta\rho$ is uncorrelated with the per-query retrieval gain ΔMRR@10 (Pearson $r = -0.013$, $p = 0.62$; points jittered vertically, line is the binned mean $\pm$ SE). **(D)** The trained-vs-random advantage is $+0.0100$ on the 100k slice but -0.0001 at the full 8.8M corpus (95% CIs). Training reshapes the scorer, yet the reshaping has little predictive value for retrieval and vanishes at scale.

8. What training actually buys

If low-margin substitution governs retrieval and the projection barely moves it (§5), what does training change? On the 100k slice trained R does lead the random orthogonal baseline – MRR@10 0.8496 vs. 0.8393, $+0.0103$ aggregate; per-query paired bootstrap $+0.0100$ (95% CI $[+0.0055, +0.0144]$, $p < 0.001$, n=6,980). The gain is real, and this section asks what produces it (Figure 5).

Training improves rank-order fidelity. What it measurably improves is the agreement between the sign-coded and fp32 document rankings – the same Spearman, Kendall, and overlap@K as §6 (Table 3). Relative to a random orthogonal R, trained R raises Spearman ρ 0.64 $\rightarrow$ 0.74, Kendall τ 0.47 $\rightarrow$ 0.55, and overlap@10 0.61 $\rightarrow$ 0.66. Training learns a real, distinct direction: trained R sits $\approx 21^\circ$ from the PCA basis, so it is neither noisy PCA nor a failed optimisation.

The improvement does not grow with more data. A checkpoint trained on 2M triples – $40\times$ the data – reaches essentially the same fidelity (ρ 0.7445 vs. 0.7401) and the same 100k MRR@10 (0.8492 vs. 0.8496): more data moves neither.

Fidelity gains do not predict which queries improve. Per query, the gain in rank-order fidelity is essentially unassociated with the change in MRR@10 (Pearson $r = -0.013$, $p = 0.62$): queries

Table 5
Two-stage retrieval on the 100k diagnostic MS MARCO dev slice (r=64). Stage 1 is sign-coded asymmetric MaxSim retrieving the top-1000 at 8 B/tok; stage 2 rescores the top-K at fp32. Reranking 100 candidates recovers fp32-level MRR@10. Rec@1000 here is the two-stage ceiling set by the stage-1 top-1000, not full-corpus fp32 retrieval (Table 1).

Pipeline	MRR@10	Rec@100	Rec@1000
fp32 MaxSim only	0.8641	0.9940	0.9989
Sign-coded, single stage (r=64)	0.8496	0.9910	0.9972
Sign-coded → fp32 rerank K=100	**0.8642**	**0.9910**	**0.9972**

where trained R gains MRR@10 and queries where it loses show the same mean fidelity gain ($+0.100$ vs. $+0.096$). The substitution-level quantities of §7 agree – per-query changes in MaxSim mass lost, argmax-flip rate, and substitute rank are all uncorrelated with the MRR@10 change ($|r| < 0.04$). Nor does training make relevant documents' codes more confident: relevant and non-relevant doc tokens sit the same mean distance from the trained hyperplanes (0.08017 vs. 0.08018). The direction training finds is real, but its per-query fidelity gain predicts neither which queries improve nor by how much – a small, broadly distributed effect rather than one concentrated in identifiable queries.

The trained-vs-random advantage is small and does not survive scale. The $+0.0100$ on the 100k slice is -0.0001 at full 8.8M (95% CI $[-0.0060, +0.0058]$). Training consistently improves agreement between sign-coded and fp32 rankings, but at full scale that improved agreement no longer yields a measurable MRR@10 gain.

This disconnect is not specific to training: among the matched-byte baselines (Table 1), OPQ minimises reconstruction error yet scores below plain PQ (0.8048 vs. 0.8330), another case where faithful reconstruction does not track retrieval quality. The pattern is consistent with training improving ordering within an already-retrievable candidate set: once retrieval is reduced to preserving that candidate set, little remains for training to improve.

9. A projection-training-free retrieval pipeline

The mechanism points to the deployment. Sign coding keeps the relevant document in the candidate set and damages only the head ordering inside it (§6–§7), so a small fp32 rerank of the head repairs what the sign code lost without disturbing the candidate set. The sign code becomes a tiny candidate index, the rerank restores the order, and projection training drops out: an 8 B/tok sign-coded stage 1 followed by an fp32 top-100 rerank recovers fp32-level MRR@10.

On the 100k diagnostic slice, sign-coded MaxSim at r=64 (8 B/tok; throughout this section "sign-coded" denotes this r=64 code) retrieves a top-K at low byte budget; fp32 MaxSim then rescores those K at full precision. Stage 2 pays K fp32 inner products per query, cheap when $K \ll N$.

The rerank closes the gap. Stage 1 alone reaches MRR@10 $= 0.8496$, 0.0145 below fp32's 0.8641. An fp32 rerank of the sign-coded top-100 closes the gap to within 0.0001 (0.8642 vs. 0.8641 – a tie within run-to-run noise). Rec@1000 is 0.9972 after stage 1 and unchanged after rerank at K=100; Rec@100 also remains at 0.9910 because positions 11–100 stay in sign-coded order at this cutoff. Sign-coded Rec@100 (0.9910) and Rec@1000 (0.9972) fall slightly short of the full fp32 values (0.9940 and 0.9989), so the candidate set is not identical to fp32 retrieval. The stage-1 projection does not matter to the MRR@10 result: a random orthogonal R reaches MRR@10 $= 0.8641$ after the same K=100 rerank, against 0.8642 for trained R (the table uses trained R for stage 1; the parity at corpus scale is established with a paired bootstrap below). The rerank, not the training, closes the gap.

Table 6
Stage-2 fp32 rerank latency over the top-K candidates (100k diagnostic slice; 32-token augmented queries, Apple M4 Pro, 48 GB RAM, macOS 26.0.1, PyTorch 2.8.0, 8-thread CPU; median of per-query medians over 5 interleaved rounds, IQR in parentheses; same harness as Table 8). Latency scales with K, not corpus size; at K=100 it is two orders of magnitude below an exhaustive fp32 scan (Table 8).

Configuration	median ms (IQR)
Stage-2 fp32 rerank, $K = 100$	1.1 (0.25)
Stage-2 fp32 rerank, $K = 256$	2.1 (0.29)
Stage-2 fp32 rerank, $K = 1024$	8.5 (0.25)

It holds at full scale. Single-stage MRR@10 is 0.3723 for trained R and 0.3724 for random orthogonal R at 8 B/tok. An fp32 rerank of the top-100 sign-coded candidates lifts random R to 0.4002 and trained R to 0.3997, both reaching the fp32 ColBERTv2 level reported by [2] (≈0.397) from the 8 B/tok index. Trained and random R are indistinguishable after rerank: paired bootstrap Δ MRR@10 $= -0.0005$ (95% CI $[-0.0012, +0.0002]$, n=6,980) at K=100, and $\Delta = +0.0000$ (95% CI $[-0.0005, +0.0006]$) at K=1000 where both reach 0.4001; both intervals include zero. For this trained checkpoint, projection training adds no statistically reliable MRR@10 gain beyond reranking, at corpus scale as at 100k.

Where the residual misses concentrate. Stage 2 orders the same candidates at full precision, recovering the head that low-margin flips had shuffled. The 169 queries (of 6,980) that still miss after K=100 are markedly more near-tied at the document level: their median fp32 top-1 vs. top-2 MaxSim margin is 0.56, compared with 3.68 for resolved queries, roughly 6.6× smaller. These are the query-level analogues of the low-margin q-tokens of §7.1 – the fp32 rerank has limited room to separate them because the fp32 encoder itself is uncertain there, so the gap that survives rerank is consistent with fp32's own uncertainty rather than a sign-coding failure. The recipe is therefore K=100: a 100-document fp32 rerank closes the MRR@10 gap to within 0.0001 of fp32 at a cost that scales with K rather than corpus size N (Table 6).

Storage. The 8 B/tok is the stage-1 candidate index, scored for every query against every document token and so RAM-resident – the tier sign coding shrinks, and the same tier PLAID holds at 34 B/tok (b=2). The fp32 vectors stage 2 needs (≈ 512 B/tok) are read only for the K≈100 reranked candidates per query, so they can sit on a cold SSD tier or be re-encoded on demand. Total doc-side storage is therefore not reduced: the saving is in the always-resident candidate index, the binarise-then-rescore tiering that BPR and binary-quantised vector databases already use [22, 5].

Stage-2 rerank latency. The rerank scores only the top K candidates, so its cost scales with K, not the corpus – the source of its cheapness. Under the same interleaved float harness as the full scan (Table 6; 32-token augmented queries, median of 5 round medians), the stage-2 fp32 rerank costs 1.1 ms at K=100, 2.1 ms at K=256, and 8.5 ms at K=1024; at K=100 that is roughly 130× below the 144 ms exhaustive fp32 scan (Table 8). Candidate generation is the separate stage-1 cost, so this is the rerank-stage figure alone.

9.1. Comparison with PLAID

If a trained projection has no edge over a random one and an fp32 rerank closes the head ordering gap, the practical question is whether the payload has a useful storage–quality point. Sign coding is a per-token storage format, not a retrieval engine: it sets how cheaply each document token is stored and scored, not which candidates are selected – candidate pruning (PLAID's centroid pruning [6]) is an orthogonal front-end we do not replace. We isolate this compression layer by scoring all codecs exhaustively over the 100k diagnostic slice. Table 7 places the sign-coded payload against our reimplementation of ColBERTv2's

Table 7
Payload storage–quality vs. PLAID at 100k MS MARCO dev (ColBERTv2 encoder). The endorsed recipe (bold) is a random orthogonal R at 8 B/tok with an fp32 top-100 rerank: it reaches fp32-parity MRR@10 from a $4.25\times$ smaller candidate index than PLAID b=2, with no projection training. Single-stage, random R sits below PLAID b=2 and trained R just above PLAID b=1, but the rerank closes the gap, so the trained row is reported only for reference. (The recipe's random R carries no scale head, unlike the trained-vs-random comparison at 8.8M.) PLAID b=1 is ColBERTv2's 1-bit residual-quantisation mode [2], the established sub-byte multi-vector compressor; the identity-sign row is the no-rotation, raw sign-coding baseline used by deployed multi-vector databases. The PLAID $b \in \{1, 2, 4\}$ rows are our codec reimplementation (centroid+b-bit residual, C=32,768) on the shared frozen embeddings, exhaustively scored – the codec, not the engine (Table 1); the Pareto framing follows [42] but the numbers are not transcribed from it. Centroid codebooks are amortised across the corpus and excluded from per-token bytes; the amortisation is non-negligible at 100k (PLAID b=2 all-in ≈ 36.5 B/tok; see the storage accounting paragraph).

Method	B/tok	MRR@10	NDCG@10
fp32 ColBERTv2	512	0.8641	0.8889
PLAID b=4	66	0.8627	0.8875
PLAID b=2	34	0.8593	0.8841
PLAID b=1	18	0.8498	0.8756
Random orthogonal R, r=64	8	0.8393	0.8670
+ fp32 rerank (K=100)	8	**0.8641**	–
Trained R, r=64 (this work, no scale)	8	0.8500	0.8755
Identity sign, r=64	8	0.8367	0.8644

b-bit residual codes at the same byte budget, on a storage–quality Pareto in the efficiency–effectiveness spirit of MacAvaney and Tonellotto [42].

The recipe needs no training, and on the storage–quality axis it reaches a point the residual codes do not. A random orthogonal R scores MRR@10 $= 0.8393$ single-stage, below PLAID b=2; the fp32 rerank of its top-100 lifts it to fp32 parity, 0.8641 (§9), from an 8 B/tok index – $4.25\times$ smaller than PLAID b=2's 34 B/tok and with no centroid table.[4] A comparable rerank may also lift the residual codec (we do not evaluate that variant), so the saving is not in quality but in the candidate-index storage that reaches it. Trained R reaches 0.8500 single-stage, but the rerank closes the difference (0.8642 vs. 0.8641 at K=100): the training that prior work assumes necessary adds nothing beyond what the rerank delivers.

Storage accounting assumptions. The $4.25\times$ win is a doc-side, candidate-index figure; the per-token bytes exclude PLAID's centroid codebook, which is amortised across the corpus and so shrinks with corpus size.[5] As §9 notes, this is the always-resident candidate index; the stage-2 fp32 store is a separate cold tier.

Retrieval latency. Scoring a residual codec first reconstructs an approximate fp32 token – a centroid plus a dequantised residual – and then scores it at the full 128 dimensions; the sign code is scored directly as a 64-dimensional vector, half the width and with no reconstruction. Under exhaustive scoring that makes it $1.21\times$ cheaper than fp32 ColBERTv2 (119.5 vs. 144.2 ms/query; Table 8). On quality the recipe, the official PLAID engine (0.8631), and uncompressed fp32 (0.8641) are within 0.0011 MRR@10 of one another. The engine itself runs far below any exhaustive scan (8.0 ms median, about $15\times$ faster than the exhaustive sign-coded scan) because it prunes to at most 4,096 candidates first – a step orthogonal to the payload that could be combined with the sign code, though we do not evaluate that integration. Storing document tokens as sign bits and scoring them against full-precision

[4]The sign code's own amortised R matrix at r=64 ($64 \times 128 \times 4 = 32$ KiB) is below 5×10^{-3} B/tok at 100k and negligible.
[5]The codebook is C=32,768 centroids $\times$ 128 dims $\times$ 4 bytes $=$ 16 MiB. At the 100k diagnostic ($\approx$ 6.73M tokens) it adds ≈ 2.5 B/tok, raising PLAID b=2 from 34 to ≈ 36.5 B/tok and the ratio to $\approx 4.6\times$; at 8.8M ($\approx$ 597M tokens) it amortises to ≈ 0.03 B/tok and the all-in ratio approaches the bare $4.25\times$.

Table 8
Per-query retrieval latency on the 100k diagnostic MS MARCO dev slice (32-token augmented queries on an Apple M4 Pro, 48 GB RAM, macOS 26.0.1, PyTorch 2.8.0, 8-thread, batch 1, CPU; median of per-round per-query medians over 5 interleaved rounds, IQR in parentheses; query encoding excluded). The two exhaustive rows isolate the distance kernel (64-dimensional sign code vs. 128-dimensional fp32); residual-codec rows are omitted because, scored exhaustively, each reconstructs an fp32 token and so costs the same as fp32. [§]The official PLAID engine (colbert-ai 0.2.22, b=2, same machine) prunes to at most 4,096 candidates, so it does different work from the exhaustive rows; on the same passages its quality is within 0.0011 MRR@10 of fp32.

Configuration	B/tok	median ms (IQR)
Sign-coded r=64, exhaustive	8	**119.5** (0.5)
fp32 ColBERTv2, exhaustive	512	144.2 (0.3)
PLAID engine, official (b=2, pruned)[§]	≈ 34	8.0 (0.4)

queries is already a deployed pattern: Vespa and Elasticsearch both offer binarised document-side token vectors, whose sign bits preserve angular similarity [26], scored with asymmetric or Hamming late-interaction [3, 33, 4]. Dropping the 8 B/tok payload into such a pruned engine, and keeping the smaller index, is the open step.

The mechanism turns into a recipe: an 8 B/tok sign-coded candidate index plus a 100-document fp32 rerank recovers fp32-level MRR@10, with no projection training and an optional frontend pruning engine.

10. Discussion

Where encoder sensitivity would come from. The per-query MaxSim sum sits on the low-margin bulk, not the high-margin tail (§7.1). An encoder's tolerance to small r should therefore track how much of its score sits at very low margin, where flips are frequent and grow costlier as the code coarsens; the high-margin fraction is only an indirect summary of this. Across our two encoders that summary moves in the expected direction: ConstBERT$_{32}$, with a thinner high-margin tail (0.35% vs. ColBERTv2's 0.50% of q-tokens at margin ≥ 0.2), degrades faster as r shrinks (§7). But the statistic separates the two encoders by little, so we read it as consistent with the account rather than a predictor.

What the decomposition suggests for stage-1 compressors. The decomposition also suggests a different design question. Methods that bet on a dominant per-token signal – top-K argmax pruning, LSH bucketing, ColBERTer's reduction towards one dimension per token [16] – tend to privilege the high-margin tail. In our decomposition, that tail largely survives sign coding but carries only 0.51% of the per-query sum (§7.1); the low-margin bulk carries the other $\approx 99.5\%$. This suggests that, for this encoder and code, a stage-1 compressor may not need to preserve every fp32 token winner. It may be enough to keep substitutions to a small local change in score across that bulk, a weaker target than per-token reconstruction fidelity. Designing compressors that target this bulk directly is a possible direction.

What the null does and does not say. The observed convergence of trained and random projections is specific to the sub-bit sign regime studied here. On the 100k slice a trained projection does win (§8), but the per-query rank-order fidelity it buys is uncorrelated with the change in MRR@10 (Pearson $r = -0.013$), and the advantage is not visible at 8.8M. We do not read this as evidence that trained projections are useless in general: under higher-bit residual codes, which preserve more of the fp32 score, learned projection blocks still report gains [31]. The claim is narrower. In our r=64 sign-code setting, the rank-order fidelity training sharpens had little predictive value for retrieval. This resembles other negative results in retrieval, such as the finding that scaling pointwise cross-encoder reranking does not monotonically improve effectiveness [43].

Scope and limitations. The mechanism comes from one encoder family (ColBERTv2 on MS MARCO), with ConstBERT$_{32}$ as a single cross-check at 100k scale but not at the full 8.8M. The corpus-scale null rests on one training seed (42; the 100k diagnostics average seeds 42–44), trained on random rather than mined hard negatives, so the learned checkpoint is a budgeted one rather than an exhaustively tuned. The margin decomposition suggests why these choices may not be the binding constraint – the high-margin tail carries only $\approx 0.5\%$ of the per-query sum in our diagnostic slice (§7.1) – but we have not tested larger training budgets, alternative negative-mining schemes, or other encoder families. Whether the account generalises beyond the ColBERT family remains open. The BEIR panel (Table 2) is a rotation-invariance control, not the trained-vs-random null, and covers nine of thirteen corpora; the four largest (HotpotQA, DBPedia-Entity, FEVER, Climate-FEVER) were excluded for compute, and SciFact's $+0.032$ outlier is the only corpus with a statistically reliable difference – the other eight fall within $|\Delta\,\text{NDCG@10}| \leq 0.015$.

Every latency figure is single-CPU, batch-1, and exhaustive (§9, §9.1); none extends to GPU, distributed, or pruned serving. In our local run, the official PLAID engine with pruning is much faster than any exhaustive scan. Sign coding is a per-token storage format and distance kernel, complementary to such engines rather than a replacement, and folding its payload behind candidate pruning at full scale remains future work. We hold *asymmetric* scoring (fp32 queries, sign-coded documents) fixed throughout; the fully binary popcount variant is a separate speed–quality point, costing 0.0132 MRR@10 single-stage ($0.8496 \rightarrow 0.8364$) at r=64. The 8.8M fp32 ColBERTv2 reference (0.397) is the level reported by [2], not an in-harness measurement.

11. Conclusion

Sign coding substantially disrupts token geometry but not retrieval. We attribute this to low-margin substitution: most MaxSim winners change, but each replacement is a near-tied document token, so the per-query sum barely moves. This unsettles a common assumption. Many prior compressors are built to track the fp32 reference per token; at sub-bit sign coding the aggregate MaxSim sum survives even when token neighbourhoods, argmaxes, and document rankings are substantially altered. In this sign-code regime, per-token fidelity is not the right target; aggregate-sum stability appears to be.

Three lines of evidence agree that the choice of projection – including whether it is learned – has no reliable full-scale MRR@10 role in this regime. At 8.8M, trained and random projections are statistically indistinguishable (Δ MRR@10 $= -0.0001$, 95% CI $[-0.0060, +0.0058]$). At r=128, identity sign coding and a random rotation tie (0.3921 vs. 0.3916). Across nine BEIR domains a plain sign code and a random rotation differ by a median $|\Delta\,\text{NDCG@10}| = 0.004$. Training improves fp32 rank-order fidelity (§8), but that improvement does not transfer to retrieval effectiveness at scale.

The practical result is a training-free (for the projection) recipe: a random orthogonal R at 8 B/tok generates candidates, and an fp32 rerank of the top 100 recovers fp32-level MRR@10 at both scales, with a doc-side payload $4.25\times$ smaller than PLAID's 2-bit residual codec. The recipe is complementary to pruning-based engines such as PLAID; combining the smaller payload with a pruned candidate step is the open engineering task.

Two questions remain. Multi-bit residual coding may restore a role for training: more bits lower the argmax-flip rate, leaving less low-margin substitution for the aggregator to absorb and more head ordering for training to improve. And encoder coverage is thin: the margin-distribution account rests on two encoder families, and whether it generalises beyond them is open, and can be tested on further late-interaction encoders.

In the one-bit and half-bit sign-code regimes we study, projection choice is not where full-scale MRR@10 is decided.

Declaration on Generative AI

During the preparation of this work, the author(s) used Claude for rephrasing, grammar and spelling check. Additionally, Claude was used for co-writing the code for experiments. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] O. Khattab, M. Zaharia, ColBERT: Efficient and effective passage search via contextualized late interaction over BERT, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2020, pp. 39–48. doi:10.1145/3397271.3401075.

[2] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, M. Zaharia, ColBERTv2: Effective and efficient retrieval via lightweight late interaction, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), ACL, 2022, pp. 3715–3734. doi:10.18653/v1/2022.naacl-main.272.

[3] J. K. Bergum, Announcing the Vespa ColBERT embedder, Vespa Blog, 2024. URL: https://blog.vespa.ai/announcing-colbert-embedder-in-vespa/.

[4] P. Straßer, B. Trent, Scaling late interaction models in Elasticsearch – part 2, Elastic Search Labs Blog, 2025. URL: https://www.elastic.co/search-labs/blog/late-interaction-model-colpali-scale.

[5] S. Aquino, Advanced retrieval with ColPali & Qdrant vector database, Qdrant Blog, 2024. URL: https://qdrant.tech/blog/qdrant-colpali/.

[6] K. Santhanam, O. Khattab, C. Potts, M. Zaharia, PLAID: An efficient engine for late interaction retrieval, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM), ACM, 2022, pp. 1747–1756. doi:10.1145/3511808.3557325.

[7] F. M. Nardini, C. Rulli, R. Venturini, Efficient multi-vector dense retrieval with bit vectors, in: Advances in Information Retrieval: 46th European Conference on Information Retrieval (ECIR), Springer, 2024, pp. 3–17. doi:10.1007/978-3-031-56060-6_1. arXiv:2404.02805.

[8] Y. Gong, S. Lazebnik, Iterative quantization: A procrustean approach to learning binary codes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2011, pp. 817–824. doi:10.1109/CVPR.2011.5995432.

[9] J. Gao, C. Long, RaBitQ: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search, Proceedings of the ACM on Management of Data (SIGMOD) 2 (2024) 1–27. doi:10.1145/3654970.

[10] J. L. Scheerer, M. Zaharia, C. Potts, G. Alonso, O. Khattab, WARP: An efficient engine for multi-vector retrieval, in: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2025, pp. 2504–2512. doi:10.1145/3726302.3729904. arXiv:2501.17788.

[11] S. Martinico, F. M. Nardini, C. Rulli, R. Venturini, Efficient multivector retrieval with token-aware clustering and hierarchical indexing, 2026. arXiv:2604.28142.

[12] C. Lassance, M. Maachou, J. Park, S. Clinchant, A study on token pruning for ColBERT, arXiv preprint arXiv:2112.06540 (2021). arXiv:2112.06540.

[13] Y. Zong, B. Piwowarski, Towards lossless token pruning in late-interaction retrieval models, in: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2025, pp. 2407–2417. doi:10.1145/3726302.3730100. arXiv:2504.12778.

[14] A. Acquavia, N. Tonellotto, C. Macdonald, Static pruning for multi-representation dense retrieval, in: Proceedings of the ACM Symposium on Document Engineering (DocEng), 2023, pp. 1–10. doi:10.1145/3573128.3604896.

[15] Y. Kankanampati, Y. Zong, N. Tomeh, B. Piwowarski, J. Le Roux, A Voronoi cell formulation for principled token pruning in late-interaction retrieval models, in: Proceedings of the 49th

International ACM SIGIR Conference on Research and Development in Information Retrieval, 2026. doi:10.1145/3805712.3809726.

[16] S. Hofstätter, O. Khattab, S. Althammer, M. Sertkan, A. Hanbury, Introducing neural bag of whole-words with ColBERTer: Contextualized late interactions using enhanced reduction, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM), 2022, pp. 737–747. doi:10.1145/3511808.3557367. arXiv:2203.13088.

[17] S. MacAvaney, A. Mallia, N. Tonellotto, Efficient constant-space multi-vector retrieval, in: Proceedings of the 47th European Conference on Information Retrieval (ECIR), 2025, pp. 237–245. doi:10.1007/978-3-031-88714-7_22. arXiv:2504.01818.

[18] H. Qin, A. Martin, R. Jha, C. Zuo, R. Kriz, B. Van Durme, Multi-vector index compression in any modality, 2026. arXiv:2602.21202.

[19] J. Veneroso, R. Jayaram, J. Rao, G. Hernández Ábrego, M. Hadian, D. Cer, CRISP: Clustering multi-vector representations for denoising and pruning, arXiv preprint arXiv:2505.11471 (2025). arXiv:2505.11471.

[20] B. Clavié, A. Chaffin, G. Adams, Reducing the footprint of multi-vector retrieval with minimal performance impact via token pooling, arXiv preprint arXiv:2409.14683 (2024). arXiv:2409.14683.

[21] L. Dhulipala, M. Hadian, R. Jayaram, J. Lee, V. Mirrokni, MUVERA: Multi-vector retrieval via fixed dimensional encodings, in: Advances in Neural Information Processing Systems (NeurIPS), 2024, pp. 101042–101073. doi:10.52202/079017-3204. arXiv:2405.19504.

[22] I. Yamada, A. Asai, H. Hajishirzi, Efficient passage retrieval with hashing for open-domain question answering, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), Association for Computational Linguistics, 2021, pp. 979–986. doi:10.18653/v1/2021.acl-short.123.

[23] H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2011) 117–128. doi:10.1109/TPAMI.2010.57.

[24] T. Ge, K. He, Q. Ke, J. Sun, Optimized product quantization for approximate nearest neighbor search, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2946–2953. doi:10.1109/CVPR.2013.379.

[25] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, S. Ma, Jointly optimizing query encoder and product quantization to improve retrieval performance, in: Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM), 2021, pp. 2487–2496. doi:10.1145/3459637.3482358. arXiv:2108.00644.

[26] M. S. Charikar, Similarity estimation techniques from rounding algorithms, in: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing (STOC), 2002, pp. 380–388. doi:10.1145/509907.509965.

[27] W. B. Johnson, J. Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, Contemporary Mathematics 26 (1984) 189–206. doi:10.1090/conm/026/737400.

[28] D. Achlioptas, Database-friendly random projections: Johnson–Lindenstrauss with binary coins, Journal of Computer and System Sciences 66 (2003) 671–687. doi:10.1016/S0022-0000(03)00025-4.

[29] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Communications of the ACM 51 (2008) 117–122. doi:10.1145/1327452.1327494.

[30] L. Caspari, M. Dinzinger, K. G. Dastidar, C. Fellicious, J. Mitrović, M. Granitzer, CoRECT: A framework for evaluating embedding compression techniques at scale, arXiv preprint arXiv:2510.19340 (2025). arXiv:2510.19340.

[31] B. Clavié, S. Lee, R. Takehi, A. Shakir, M. P. Kato, Simple projection variants improve ColBERT performance, arXiv preprint arXiv:2510.12327 (2025). arXiv:2510.12327.

[32] Y. Chen, Y. Zhang, H. Guo, R. Tang, I. King, An effective post-training embedding binarization approach for fast online top-K passage matching, in: Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International

Joint Conference on Natural Language Processing (Volume 2: Short Papers), Association for Computational Linguistics, 2022, pp. 102–108. doi:10.18653/v1/2022.aacl-short.14.

[33] J. K. Bergum, Scaling ColPali to billions of PDFs with Vespa, Vespa Blog, 2024. URL: https://blog.vespa.ai/scaling-colpali-to-billions/.

[34] R. Nogueira, K. Cho, Passage re-ranking with BERT, arXiv preprint arXiv:1901.04085 (2019). arXiv:1901.04085.

[35] R. Nogueira, Z. Jiang, R. Pradeep, J. Lin, Document ranking with a pretrained sequence-to-sequence model, in: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 708–718. doi:10.18653/v1/2020.findings-emnlp.63. arXiv:2003.06713.

[36] T. Formal, S. Clinchant, H. Déjean, C. Lassance, SPLATE: Sparse late interaction retrieval, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 2635–2640. doi:10.1145/3626772.3657968. arXiv:2404.13950.

[37] J. Lee, Z. Dai, S. M. K. Duddu, T. Lei, I. Naim, M.-W. Chang, V. Zhao, Rethinking the role of token retrieval in multi-vector retrieval, in: Advances in Neural Information Processing Systems (NeurIPS), 2023, pp. 15384–15405. doi:10.52202/075280-0677. arXiv:2304.01982.

[38] Q. Liu, G. Guo, J. Mao, Z. Dou, J.-R. Wen, H. Jiang, X. Zhang, Z. Cao, An analysis on matching mechanisms and token pruning for late-interaction models, ACM Transactions on Information Systems 42 (2024) 1–28. doi:10.1145/3639818. arXiv:2403.13291.

[39] S. Archish, A. Garg, K. Shiragur, N. Kayal, Incorporating token importance in multi-vector retrieval, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2026, pp. 32860–32866. doi:10.1609/aaai.v40i39.40566. arXiv:2511.16106.

[40] A. Edy, M. Conti, Q. Macé, Working notes on late interaction dynamics: Analyzing targeted behaviors of late interaction models, arXiv preprint arXiv:2603.26259 (2026). arXiv:2603.26259.

[41] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. Gurevych, BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models, in: Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2021. arXiv:2104.08663.

[42] S. MacAvaney, N. Tonellotto, A reproducibility study of PLAID, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2024, pp. 1411–1419. doi:10.1145/3626772.3657856. arXiv:2404.14989.

[43] M. Jacob, E. Lindgren, M. Zaharia, M. Carbin, O. Khattab, A. Drozdov, Drowning in documents: Consequences of scaling reranker inference, in: Proceedings of the ReNeuIR Workshop at SIGIR 2025, 2025. arXiv:2411.11767.