Efficient Multivector Retrieval with Token-Aware Clustering and Hierarchical Indexing

Silvio Martinico*
University of Pisa
Pisa, Italy
ISTI-CNR
Pisa, Italy
silvio.martinico@phd.unipi.it

Franco Maria Nardini
ISTI-CNR
Pisa, Italy
francomaria.nardini@isti.cnr.it

Cosimo Rulli
ISTI-CNR
Pisa, Italy
cosimo.rulli@isti.cnr.it

Rossano Venturini
University of Pisa
Pisa, Italy
rossano.venturini@unipi.it

Abstract

Multivector retrieval models achieve state-of-the-art effectiveness through fine-grained token-level representations, but their deployment incurs substantial computational and memory costs. Current solutions—based on the well-known κ-means clustering algorithm—group similar vectors together to enable both effective compression and efficient retrieval. However, standard κ-means scales poorly with the number of clusters and dataset size, and favours frequent tokens during training while underrepresenting rare, discriminative ones. In this work, we introduce Tachiom, a multivector retrieval system that exploits token-level structure to significantly accelerate both clustering and retrieval. By accounting for tokens' distribution during centroid allocation, Tachiom easily scales to millions of centroids, enabling highly accurate document scoring using only centroids, avoiding expensive token-level computation. Tachiom combines a graph-based index over centroids with an optimized Product Quantization layout for efficient final scoring. Experiments on Ms Marco-v1 and LoTTE show that Tachiom achieves up to 247× faster clustering than κ-means and up to 9.8× retrieval speedup over state-of-the-art systems while maintaining comparable or superior effectiveness.

CCS Concepts

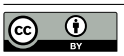
• **Information systems → Top-k retrieval in databases**.

Keywords

Multivector Retrieval, Late Interaction, Clustering, Efficiency.

ACM Reference Format:
Silvio Martinico, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2026. Efficient Multivector Retrieval with Token-Aware Clustering and Hierarchical Indexing. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3805712.3809927

*All authors contributed equally to this research.



This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGIR '26, Melbourne, VIC, Australia
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3809927

1 Introduction

Transformer-based language models [4, 8, 13, 15, 28] have fundamentally reshaped Information Retrieval (IR), shifting the paradigm from lexical-based to representation-based retrieval. Within this landscape, multivector models, such as ColBERT [14, 25], have emerged as a gold standard for effectiveness. By encoding documents as sets of token-level vectors and computing relevance via late interaction, i.e., allowing every query token to interact with every document token, these models capture fine-grained semantic nuances that single-vector dense retrievers often miss. However, the real-world deployment of multivector models remains severely constrained by their computational demands. Storing and searching large sets of token vectors creates massive memory footprints and computational overhead. To mitigate this, state-of-the-art solutions adopt a *gather-and-refine* strategy combined with token quantization [1, 6, 20, 24, 26]. In the gather phase, a lightweight search over token vectors identifies promising candidates; in the refine phase, full MaxSim scores are computed only for the selected candidates. Both phases critically depend on a coarse quantizer—a set of centroid vectors that approximate the token embedding space. During gathering, centroids serve as efficient proxies for locating relevant tokens. For compression, each token vector is decomposed into a centroid assignment and a residual vector, which is then compressed using more precise Vector Quantization (VQ) techniques, like, for instance, Product Quantization [12] (PQ). This two-level scheme substantially improves approximation: centroid assignments distinguish distant vectors at minimal cost, while residual compression captures local nuances.

ColBERTv2 [25] pioneered this centroid-based approach, compressing the residual (elementwise difference between the vector and the centroid) rather than the vector itself. Plaid [24] extended this framework with centroid interaction and inverted-file filtering over clustered token embeddings for efficient retrieval. Subsequent work refined this framework: Emvb [20] introduced bitvector prefiltering to accelerate candidate selection and optimized PQ variants [7, 10] to improve redisuals' compression, achieving significant speedups over Plaid. Warp [26] combined Plaid's centroid

interaction with XTR's lightweight architecture [16]; the authors showed that their method also generalize to COLBERTv2 embeddings. IGP [1] proposed building a graph index on the set of centroids to avoid exhaustive centroids scoring.

Across all these methods, both retrieval quality and compression effectiveness scale directly with the number of centroids: finer-grained centroids enable more precise approximations and more selective gathering. However, standard κ-means clustering—the predominant method for centroids selection—creates a severe bottleneck: it scales poorly with both dataset size and number of clusters, and critically, it ignores the token structure of multivector embeddings. Training even moderately large sets of centroids (e.g., 262K) on multivector collections can require tens of hours on multi-core CPUs or expensive GPU resources, fundamentally limiting achievable granularity. In this paper, we address this bottleneck by exploiting a key structural property of multivector embeddings: *token identity*. Unlike generic vector sets, multivector collections exhibit strong correlations between token type and embedding distribution. Common tokens (e.g., stopwords) dominate κ-means' loss during training, consuming the majority of centroids, while rare domain-specific tokens—often more discriminative for retrieval—receive minimal representation. We introduce TACHIOM, a multivector retrieval system that exploits this structure at both the clustering and retrieval stages. At its core, TACHIOM employs Token-Aware Clustering (TAC), which explicitly allocates centroids based on token frequency and semantic variance. By partitioning the clustering workload across token groups, TAC achieves significantly faster centroids computation over standard κ-means. This efficiency enables scaling to millions of centroids, far beyond prior work. Leveraging these high-resolution centroids, TACHIOM performs gathering using only centroid-level interactions via an HNSW graph index [18], so as to bypass expensive token-level computations entirely. For refinement, a cache-optimized PQ layout tailored for late interaction enables efficient final scoring.

In summary, the novel contributions of this work are:

- We propose TAC, a clustering algorithm that leverages token identity to decompose the global clustering problem into independent per-token subproblems, improving scalability while producing centroids better aligned with retrieval objectives. We further derive a theoretical lower bound on the computational speed-up achieved by TAC over κ-means.
- We introduce TACHIOM, a retrieval architecture that leverages the clustering granularity enabled by TAC to perform centroids-only gathering via graph-based search and efficient full scoring with late interaction-tailored PQ.
- We conduct comprehensive experiments on MS MARCO-V1 and LOTTE, demonstrating that our proposed architectures offer superior efficiency-effectiveness trade-offs. Our results show that TAC achieves up to 247× faster training than standard κ-means, while TACHIOM delivers up to 9.8× faster end-to-end retrieval compared to state-of-the-art systems. To favor reproducibility, we release our implementation of TAC and TACHIOM in Rust on GitHub at `https://github.com/TusKANNy/tachiom`.

2 Methodology

2.1 Token-Aware Clustering

Multivector models such as COLBERT [14] represent each document as a set of contextualized token embeddings. State-of-the-art retrieval methods [1, 7, 20, 24–26] rely on centroids to approximate these large collections of token vectors, typically using κ-means clustering to select representative centroids. However, standard κ-means treats token embeddings as a generic vector set, discarding the structural information provided by token identity. Token-Aware Clustering (TAC) explicitly exploits the token structure of multivector representations. TAC serves two purposes: (i) it significantly accelerates clustering by partitioning the workload into independent per-token subproblems, crucial for the massive vector collections produced by COLBERT-like models; (ii) it balances centroid allocation toward rare tokens, which are underrepresented by standard κ-means despite their importance for retrieval accuracy.

κ-means aims to find a set of clusters $C = \{C_1, \ldots, C_\kappa\}$ that partitions the dataset and minimizes the Within-Cluster Sum of Squares (WCSS) or Inertia, i.e.:

$$\sum_{i=1}^{\kappa} \sum_{\mathbf{t} \in C_i} \|\mathbf{t} - \mathbf{c}_i\|^2 ,$$

where $\mathbf{c}_i$ denotes the centroid of cluster C_i. This objective treats all vectors equally, but multivector collections exhibit severe frequency imbalance. Common tokens (e.g., stopwords) may appear millions of times, while domain-specific terms have way lower frequencies (hundreds or few thousands of occurrences). Analysing two COLBERTv2 multivector collections, we observed that the top 100 most frequent tokens account for over 40% of all vectors, specifically 41% on MS MARCO-V1 and 45% on LOTTE-pooled. Consequently, the WCSS loss is dominated by high-frequency tokens, which consume the majority of centroids despite contributing little to retrieval quality. Meanwhile, rare tokens—which often carry the most discriminative semantic signal [27]—receive minimal representation, degrading approximation quality precisely where it matters most.

Drawing from classic Information Retrieval literature, where *Inverse Document Frequency* (IDF) has long been employed to favor rare, discriminative terms [22, 23], we propose TAC to address this imbalance in the clustering context. TAC decouples centroid allocation from uniform WCSS minimization by explicitly weighting tokens based on their frequency and semantic variance, so as to favor rare tokens. TAC first computes a weight for each token type representing its share of the total centroid budget, then performs independent κ-means clustering for each token type.

TAC distributes the centroid budget through a four-stage pipeline: (1) *Tail Handling* isolates very rare tokens to prevent their complete marginalization; (2) *Damped Scoring* computes an allocation weight for the remaining tokens; (3) *Bounding* enforces strict limits to avoid both under- and over-allocation; and (4) *Budget Reconciliation* adjusts the final assignments to exactly match the global centroid budget.

Phase 1: Tail Handling. We partition token types into three categories based on their frequency n: *Micro tokens* ($n < \mu$) receive 1 centroid each, *Small tokens* ($\mu \leq n < \tau$) receive 2 centroids each, and *Active tokens* ($n \geq \tau$) are allocated centroids dynamically based

on their frequency and semantic variance. This fixed allocation prevents the complete marginalization of extremely rare tokens.

Phase 2: Damped Scoring. For active tokens, we compute a *spread measure* s_j that quantifies semantic variance; we adopt component-wise variance for computational efficiency, i.e.,

$$s_j \coloneqq \frac{1}{n_j} \sum_{i=1}^{n_j} \|\mathbf{t}_{j,i} - \bar{\mathbf{t}}_j\|^2 ,$$

where n_j is the frequency of token j, $\mathbf{t}_{j,i}$ is the i-th occurrence of token j, and $\bar{\mathbf{t}}_j$ is its mean embedding. We then compute a *damped weight*, $\mathrm{w}_j = \sqrt{n_j} \cdot s_j$. The square-root damping applies diminishing returns to highly frequent tokens, preventing them from monopolizing centroids budget; the spread coefficient s_j favors tokens with high semantic variance. The number of centroids κ_j of token j is allocated proportionally to its weight,

$$\kappa_j = \left\lfloor \frac{w_j}{\sum_{i=1}^{N_T} w_i} \cdot B \right\rfloor ,$$

where N_T is the number of different tokens in our collection and B is the remaining centroid budget after tail handling.

Phase 3: Bounding. We enforce a hard floor ε such that $\kappa_j \geq \varepsilon$ for all active tokens to guarantee minimum representation. Additionally, we impose an upper bound to prevent over-allocation: each token must satisfy $\frac{n_j}{\kappa_j} \geq \theta$, ensuring at least θ vectors per centroid on average. This dual bounding prevents both under-representation of rare tokens in low-budget regimes and over-allocation to already well-approximated tokens in high-budget regimes.

Phase 4: Budget Reconciliation. We redistribute surplus or deficit centroids to exactly match the target budget κ. Once centroids are allocated, we perform independent κ_j-means clustering for each token j with its assigned κ_j centroids.

In summary, Tac fundamentally differs from standard κ-means in two key aspects: 1) it prioritizes retrieval quality over pure reconstruction error: by allocating centroids based on token frequency and semantic variance rather than uniform WCSS contribution, Tac ensures that rare, discriminative tokens receive adequate representation in terms of assigned centroids; 2) instead of optimizing a single global objective over all N vectors with κ centroids, Tac solves independent subproblems for each token, clustering the n_j vectors of token j into κ_j centroids. This decomposition substantially reduces the per-iteration computational cost and enables effective parallelization, as each token can be processed independently, whereas standard κ-means requires shared access to the full dataset. Moreover, the damped centroid allocation mitigates the bottleneck caused by highly frequent tokens, further improving both total cost and parallel efficiency.

Theoretical Lower Bound for Speed-up. Standard κ-means requires $O(I \cdot N \cdot \kappa \cdot d)$ operations for I iterations over N vectors with κ centroids in d dimensions. Tac reduces this to $O(I \cdot \sum_{j=1}^{N_T} n_j \cdot \kappa_j \cdot d)$.

We can provide a minimum theoretical speedup of Tac over κ-means. Ignoring floor operations and tail handling (Phase 1) for clarity, we have a total centroids budget κ and the resulting number of centroids assigned to token j becomes $\kappa_j = \frac{w_j}{\sum_{i=1}^{N_T} w_i} \cdot \kappa$. Thus, the speedup achieved by Tac over κ-means is:

$$\frac{\kappa N}{\sum_{j=1}^{N_T} \kappa_j n_j} = \frac{\kappa N}{\sum_{j=1}^{N_T} (w_j / \sum_{i=1}^{N_T} w_i) \kappa n_j} = \frac{\kappa \left(N \sum_{j=1}^{N_T} w_j \right)}{\kappa \left(\sum_{j=1}^{N_T} w_j n_j \right)}$$

$$\geq \frac{N \sum_{j=1}^{N_T} w_j}{\max_{j=1}^{N_T} w_j \sum_{j=1}^{N_T} n_j} = \frac{N \left(\sum_{j=1}^{N_T} w_j \right)}{\left(\max_{j=1}^{N_T} w_j \right) N} = \frac{\sum_{j=1}^{N_T} w_j}{\max_{j=1}^{N_T} w_j} .$$

That is, the speedup is lower-bounded by the ratio of the total weight to the maximum single token weight. Because the cumulative weight across the vocabulary vastly exceeds that of any single token, this formulation guarantees a substantial baseline speedup. Furthermore, this bound reveals a key insight: our damped scoring mechanism not only balances centroid allocation toward rare tokens, but actively suppresses the dominance of the most common ones. By applying square-root scaling to frequencies, Tac directly reduces the maximum single-token weight, strictly improving the speedup guarantee and preventing these dominant tokens from creating a computational bottleneck.

Residuals Compression. After computing the centroids, we compress the residuals, i.e., the difference between each token vector in the dataset and its assigned centroid, using PQ [12]. Because residual magnitudes may vary across tokens due to the non-uniform centroid allocation of Tac, we normalize the residual vectors and store their norms separately. The resulting normalized residuals are more homogeneous and therefore easier to compress effectively.

2.2 Hierarchical Index for Optimized Multivector Retrieval

The speedups achieved by Tac over κ-means allow us to significantly scale up the number of centroids used. This fine-grained clustering significantly improves the approximation provided by centroids alone, thus eliminating the need for full token scoring (centroid + residual) during the gather phase, differently from existing approaches [1, 20, 26]. Furthermore, by decoupling the gather phase from the residual scores calculation, we can specialize the refinement stage with a Product Quantization layout optimized specifically for document-level *MaxSim* evaluation. Building on these observations, we introduce Tachiom (Token-Aware Clustering and Hierarchical Indexing for Optimized Multivector retrieval), a retrieval architecture that combines an Hnsw proximity graph over centroids for efficient gathering with a cache-optimized PQ layout for efficient refinement. Tachiom operates in two main phases: first, it explores centroids to select a set of candidate documents; in the second phase, it uses both centroids and residuals to compute the full *MaxSim* scores of the candidate documents and provide the final ranking.

Index Structure. We build an Hnsw [18] graph over the centroid set; each centroid $\mathbf{c}_i$ maintains an inverted list $\mathcal{L}_i$ containing the document IDs of all tokens assigned to $\mathbf{c}_i$. This design enables document-level scoring during the gather phase without accessing individual token vectors.

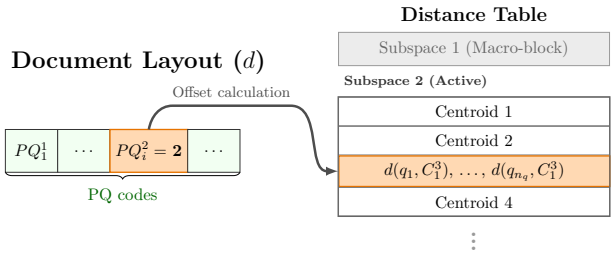


Figure 1: Distance tables layout with 4 centroids per subspace. Evaluating code $j = 2$ for subspace $m = 2$, document token i.

Gather Phase: Centroid-based Document Scoring. For each query token $\mathbf{q}_i$ of a query q:

(1) we traverse the graph to identify the top-κ_c nearest centroids;
(2) for each retrieved centroid $\mathbf{c}_j$, we iterate over its inverted list $\mathcal{L}_j$;
(3) for documents appearing in multiple lists, we retain only the highest centroid similarity: $\tilde{s}_i(d) = \max_{\{j : d \in \mathcal{L}_j\}} \langle \mathbf{q}_i, \mathbf{c}_j \rangle$;
(4) we accumulate partial scores across all query tokens: $\tilde{S}(q, d) = \sum_{i=1}^{n_q} \tilde{s}_i(d)$.

This produces an approximate *MaxSim* score using only centroid-level interactions. Documents appearing in inverted lists for multiple query tokens naturally accumulate higher scores, capturing the multi-token matching signal without decompressing token vectors. Before the next phase, we truncate the candidate set to the top κ_d documents by partial score $\tilde{S}(q, \cdot,)$. We also apply adaptive filtering to further reduce the candidate set, adopting the *Candidates Pruning* (CP) strategy proposed in [19], regulated by a parameter α.

Refine Phase: PQ-Optimized MaxSim Computation. For the pruned candidate set, we compute full *MaxSim* scores using both centroids and PQ-compressed residuals. We adopt specialized data layouts optimized for cache locality and SIMD vectorization.

Document Layout: For each document d, we store all tokens' centroid IDs contiguously, followed by all PQ codes: $[\mathbf{c}_1^d, \ldots, \mathbf{c}_{n_d}^d \mid \mathrm{PQ}_1^1, \ldots, \mathrm{PQ}_{n_d}^M]$, where n_d is the number of tokens in document d and M is the number of PQ subspaces. This enables streaming evaluation: we first accumulate all centroid contributions in one pass, then refine with residuals in a second pass.

Distance Table Layout: PQ requires precomputing distance tables between query tokens and PQ centroids for each PQ subspace. A naive multivector PQ implementation processes each query token independently, resulting in a separate distance table per token. However, because a document token's quantization is fixed, all n_q query tokens must evaluate their distance against the *same* PQ centroid for any given subspace. Consequently, the standard memory layout necessitates n_q scattered memory accesses across n_q disjoint tables, creating a severe memory-bandwidth bottleneck. To eliminate this overhead and explicitly exploit the shared access pattern inherent to the MaxSim operator, we reorganize the precomputed distances into a cache-optimized three-level hierarchy:

- *Macro-blocks*: indexed by PQ subspace (M blocks for M subspaces);
- *Centroid blocks*: indexed by PQ centroid ID (2^b blocks for b-bit codes);
- *Query token micro-blocks*: containing n_q consecutive distances between each query token and the same PQ centroid on that subspace.

When evaluating a document token's PQ code j for subspace m, our layout enables retrieving the distances to all n_q query tokens as a single contiguous micro-block (Figure 1). This avoids the n_q scattered memory lookups of the standard layout, yielding up to a 3.8× speedup in residual distance computation compared to a standard PQ implementation.

3 Experiments

We perform experiments on Ms Marco-v1 [21] (8.8M passages, 598M token vectors, 6,980 dev.small queries) for in-domain evaluation and LoTTE-pooled [25] (2.4M passages, 266M token vectors, 2,931 search/dev queries) for out-of-domain evaluation. We use ColBERTv2 [25] as the encoder with a maximum of 180 tokens per document for LoTTE. We evaluate the retrieval results using the standard metrics for each dataset: MRR@10 for Ms Marco-v1 and Success@5 for LoTTE. Experiments run on an Intel Xeon Silver 4314 CPU @ 2.40GHz with 64 threads. Our implementation is written in Rust `1.92.0-nightly` on top of the kANNolo [3] library. Clustering exploits all 64 threads in parallel, while retrieval experiments are executed sequentially on a single core.

3.1 Clustering Evaluation

We first evaluate Tac's efficiency and effectiveness on Ms Marco-v1 against two κ-means baselines: Faiss [5], which is one of the most widely-adopted κ-means implementations, and FastKMeans-rs, an efficient Rust implementation of the FastKMeans library [2]. Since our implementation uses kANNolo's AVX2-optimized single-vector distance kernels without production-ready external libraries such as Intel MKL [11], we evaluate two Faiss configurations: generic AVX2 (comparable to our setup) and MKL-enabled.

For Tac, we set tail handling thresholds $\mu = 128$ and $\tau = 256$, and bounding parameters $\varepsilon = 4$ and $\theta = 39$ based on preliminary experiments. We compress residuals using PQ with $M = 32$ subspaces and 8-bit codes per subspace. The number of κ-means iterations is set to 10 for both standard κ-means and Tac.

Clustering time measurements include (i) centroids computation and (ii) centroid assignments for all vectors in the dataset. The sample size for κ-means is set to 10M vectors; we assume a 24-hour budget for algorithms' running time. We assess clustering quality by measuring retrieval effectiveness when using the compressed representations to rerank candidates. In order to isolate vector approximation from index approximation, we do not rely on Tachiom for retrieval. Following the methodology in [19], we rerank the top 1,000 documents retrieved by Splade CoCondenser [9] using the RerankIndex with all optimizations disabled—specifically removing early exit and pruning mechanisms—and return the top 10 results for each query. This approach is designed to emulate an exhaustive search while maintaining feasible query times, allowing us to precisely assess our centroids' approximation quality.

The results of this evaluation are reported in Figure 2. As shown in the bottom plot, Tac achieves training times orders of magnitude

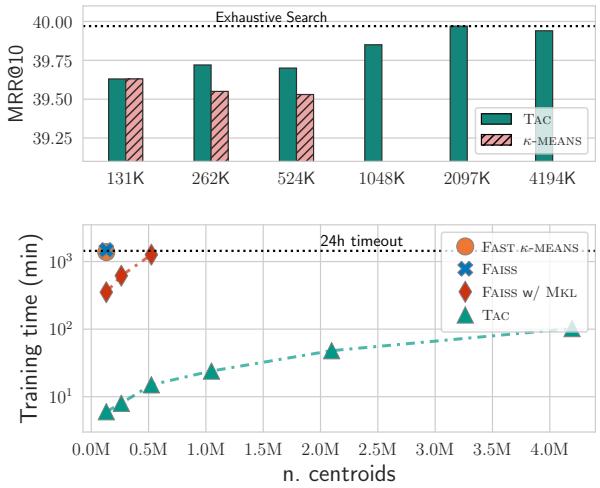


Figure 2: Ms Marco-v1. Top plot: MRR@10 for 1,000 candidates reranking. Bottom plot: clustering time in minutes.

lower than all competing methods, even when they benefit from Intel MKL acceleration. Specifically, Tac yields speedups up to 84× (Faiss with MKL), 247× (Faiss), and 230× (FastKMeans-rs) over the competitors. Our method clusters nearly 600 million 128-dimensional vectors into 262K clusters in just 8 minutes, potentially benefiting prior methods such as Plaid, Emvb, Warp, and Igp, which all rely on this same centroid budget. Furthermore, Tac scales seamlessly to over 4M centroids in only 102 minutes, while the 24-hour timeout prevents the competitors to surpass 131K centroids (Faiss and FastKMeans-rs) and 524K (Faiss with MKL).

The top plot of Figure 2 compares the MRR@10 achieved by Tac and κ-means on Ms Marco-v1 across different centroid budgets (both with normalized residuals for a fair comparison). We report the MRR@10 achieved by exhaustive search over ColBERTv2 as reference[17]. This plot highlights that Tac not only offers substantial speedups but also delivers equal or superior approximation quality at a fixed centroid budget, confirming that our allocation strategy effectively improves centroid assignment and validating the benefits of token-aware design for retrieval effectiveness.

3.2 Retrieval Evaluation

We evaluate the efficiency-effectiveness trade-off offered by Tachiom by comparing it to three state-of-the-art [19] multivector retrieval methods: Emvb [20], Warp [26] and Igp [1]. We report the best (lowest) average query time for Tachiom and all selected competitors at different metrics cut-off points.

For Tachiom, we set the number of centroids to $2^{21} \approx 2$M for LoTTE and $2^{22} \approx 4$M for Ms Marco-v1. The Hnsw graphs over centroids are built with $M = 32$ neighbors and $ef_c = 1500$. At search time, we perform grid search over: number of retrieved centroids per query token $\kappa_c \in \{15, 20, 40, 80, 100, 120\}$, maximum candidate documents $\kappa_d \in \{250, 500, 1000, 2000, 4000\}$, and candidate pruning threshold $\alpha \in \{0.35, 0.4, 0.45, 0.5\}$, with Hnsw search parameter $ef_s = \frac{3}{2}\kappa_c$. For competitors, we follow the grid search procedures from [19]. All the methods use the same residuals size, i.e. 32 bytes per token vector. This is achieved with PQ32 or its variants for Tachiom and Emvb, and with 2 bits per component for the Scalar Quantization employed by Warp and Igp.

Table 1: Average query time (ms). "–" indicates the technique does not reach the effectiveness level.

	Ms Marco-v1 (MRR@10)			**LoTTE** (Success@5)		
	39.0	39.3	39.6	67.5	68.0	68.5
Warp	98 9.8×	98 7.0×	–	49 4.4×	–	–
Igp	72 7.2×	–	–	48 4.3×	–	–
Emvb	55 5.5×	56 4.0×	57 3.8×	54 4.9×	55 3.9×	59 2.5×
Tachiom	**10**	**14**	**15**	**11**	**14**	**23**

Our results are reported in Table 1. Tachiom exhibits superior efficiency across all metric cutoffs, achieving speedups ranging from 2.5× to 9.8× compared to the baselines. Tachiom is up to 5.5× faster than the best available data structure Emvb and matches its peak effectiveness despite employing standard Product Quantization (PQ). Emvb, conversely, relies on computationally expensive PQ variants, namely JMPQ [7]—a supervised method that jointly trains centroids, PQ, and the query encoder—on Ms Marco-v1 and OPQ [10] on LoTTE. These results demonstrate that Tachiom, through its careful centroid selection and ability to scale centroids via Tac, rivals the effectiveness of both JMPQ and OPQ without incurring their additional training overhead.

4 Conclusions and Future Work

In this work, we presented Tac, a clustering algorithm that decomposes the global clustering problem into independent per-token subproblems and allocates centroids to tokens based on their frequency and semantic variance, achieving up to 247× speedup over k-means and improving effectiveness. On top of this, we developed Tachiom, a retrieval architecture that leverages high-granularity centroids to efficiently gather candidates, bypassing expensive token-level interactions. Tachiom achieves a 9.8× speedup over state-of-the-art methods. Together, these contributions unlock the practical utility of multivector retrieval for large-scale datasets. Future work will assess the generalizability of Tac beyond ColBERTv2 and evaluate Tachiom on datasets substantially larger than Ms Marco-v1. Additionally, by leveraging the superior approximation quality of Tac centroids, we intend to investigate higher residual compression ratios.

Acknowledgements

This research was conducted as part of the ENRESFOOD project, funded under the FutureFoodS partnership. The project is co-funded by the Austrian Science Fund (FWF; 10.55776/KIN4661525), The Dutch Ministry of Agriculture, Fisheries, Food Security and Nature (BO-43-219-026), the Italian Ministry of University and Research (MUR), the German Federal Ministry of Research, Technology and Space (031B1717), and the European Union's Horizon Europe research and innovation programme via FutureFoodS, the European Partnership for a Sustainable Future of Food Systems, co-funded under Grant Agreement No. 101136361.

References

[1] Zheng Bian, Man Lung Yiu, and Bo Tang. 2025. IGP: Efficient Multi-Vector Retrieval via Proximity Graph Index. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2524–2533. doi:10.1145/3726302.3730004

[2] Benjamin Clavié and Benjamin Warner. 2025. fastkmeans: Accelerated KMeans Clustering in PyTorch and Triton. https://github.com/AnswerDotAI/fastkmeans/.

[3] Leonardo Delfino, Domenico Erriquez, Silvio Martinico, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2025. kANNolo: Sweet and Smooth Approximate k-Nearest Neighbors Search. In *Proceedings of the 47th European Conference on Information Retrieval*. 400–406.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186. doi:10.18653/v1/N19-1423

[5] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The FAISS library. *arXiv preprint arXiv:2401.08281* (2024).

[6] Joshua Engels, Benjamin Coleman, Vihan Lakshman, and Anshumali Shrivastava. 2023. DESSERT: an efficient algorithm for vector set search with vector set queries. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS)*. Article 2974, 21 pages.

[7] Yan Fang, Jingtao Zhan, Yiqun Liu, Jiaxin Mao, Min Zhang, and Shaoping Ma. 2022. Joint Optimization of Multi-vector Representation with Product Quantization. In *Proceedings of the 11th CCF International Conference Natural Language Processing and Chinese Computing*. 631–642. doi:10.1007/978-3-031-17120-8_49

[8] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[9] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2353–2359. doi:10.1145/3477495.3531857

[10] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized Product Quantization. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 4 (2014), 744–755. doi:10.1109/TPAMI.2013.240

[11] Intel Corporation. 2024. *Intel® oneAPI Math Kernel Library (oneMKL)*. https://software.intel.com/en-us/mkl

[12] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE transactions on pattern analysis and machine intelligence* 33 (2011), 117–28. doi:10.1109/TPAMI.2010.57

[13] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781. doi:10.18653/v1/2020.emnlp-main.550

[14] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 39–48. doi:10.1145/3397271.3401075

[15] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. *arXiv preprint arXiv:2403.06789* (2024).

[16] Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Y. Zhao. 2023. Rethinking the role of token retrieval in multi-vector retrieval. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS)*. Article 677, 22 pages.

[17] Sean MacAvaney and Nicola Tonellotto. 2024. A Reproducibility Study of PLAID. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1411–1419. doi:10.1145/3626772.3657856

[18] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836. doi:10.1109/TPAMI.2018.2889473

[19] Silvio Martinico, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2026. Multivector Reranking in the Era of Strong First-Stage Retrievers. In *Advances in Information Retrieval*, Ricardo Campos, Adam Jatowt, Yanyan Lan, Mohammad Aliannejadi, Christine Bauer, Sean MacAvaney, Avishek Anand, Zhaochun Ren, Suzan Verberne, Nan Bai, and Masoud Mansoury (Eds.). Springer Nature Switzerland, Cham, 49–65.

[20] Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient Multi-vector Dense Retrieval with Bit Vectors. In *Proceedings of the 46th European Conference on Information Retrieval (ECIR)*. 3–17. doi:10.1007/978-3-031-56060-6_1

[21] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches*, Vol. 1773. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[22] Stephen E. Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *J. Documentation* 60 (2004), 503–520. https://api.semanticscholar.org/CorpusID:8864928

[23] G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (Nov. 1975), 613–620. doi:10.1145/361219.361220

[24] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022. PLAID: An Efficient Engine for Late Interaction Retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM)*. 1747–1756. doi:10.1145/3511808.3557325

[25] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3715–3734. doi:10.18653/v1/2022.naacl-main.272

[26] Jan Luca Scheerer, Matei Zaharia, Christopher Potts, Gustavo Alonso, and Omar Khattab. 2025. WARP: An Efficient Engine for Multi-Vector Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2504–2512. doi:10.1145/3726302.3729904

[27] Karen Sparck Jones. 1988. *A statistical interpretation of term specificity and its application in retrieval*. Taylor Graham Publishing, GBR, 132–142.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*. 6000–6010.