Stability-Driven Chunking and Representational Reranking for Dynamic Vector Databases

J. W. Miller
CURV Institute
j.w.miller@curv.institute

January 2026

Abstract

Vector databases are increasingly used in dynamic settings where documents are incrementally ingested, edited, and queried over time. In such regimes, heuristic chunking and similarity-based retrieval can lead to unstable embeddings, excessive re-indexing, and inconsistent retrieval behavior. In this work, we present a staged empirical study of stability-driven chunking and representational reranking for vector databases. We show that chunking based on representational stability produces fewer, more coherent chunks and supports streaming ingestion with bounded deviation from offline segmentation. We report a negative result demonstrating that stability-driven chunking is more sensitive to localized edits under naive, count-based mutation metrics. We then show that this apparent failure is a measurement artifact: when mutation cost is measured in bytes rather than chunk counts, a hybrid chunking policy that combines stability-driven base chunks with localized overlapping micro-chunks achieves the lowest effective update cost. Finally, we demonstrate that representational reranking improves retrieval robustness in the tails, increasing worst-case reformulation stability without affecting mutation cost or determinism. Together, these results disentangle representational coherence, mutation tolerance, and retrieval robustness, and highlight the importance of cost-aligned metrics in evaluating dynamic vector retrieval systems.

1 Introduction

Vector similarity search has become a core component of modern retrieval-augmented generation, agent memory systems, and long-lived knowledge bases. Most vector database systems rely on heuristic chunking strategies—such as fixed-size segments with overlap—combined with approximate nearest-neighbor (ANN) search using cosine or Euclidean similarity. While effective in static benchmarks, these choices implicitly assume stable embeddings and infrequent updates.

In practice, many vector databases operate under continuous ingestion and localized edits. Under such conditions, naive chunking can induce excessive re-embedding, index churn, and retrieval instability. These effects are often invisible to benchmark-style evaluations, which emphasize static accuracy rather than temporal behavior.

This paper investigates chunking and retrieval strategies through the lens of stability. Rather than optimizing for semantic accuracy alone, we ask how chunking and retrieval choices affect structural coherence, mutation tolerance, and robustness over time. We conduct a staged experimental program that isolates these axes and reports both positive and negative results.

2 Experimental Setup

We study a vector database pipeline consisting of deterministic chunking, fixed embedding generation, FAISS-based ANN indexing, and SQLite-backed metadata storage. All experiments are deterministic and reproducible, with per-run manifests recording configuration, versioning, and diagnostics. Unless otherwise noted, cosine similarity is used for ANN candidate generation.

Chunking strategies evaluated include fixed-size overlapping chunking, stability-driven chunking

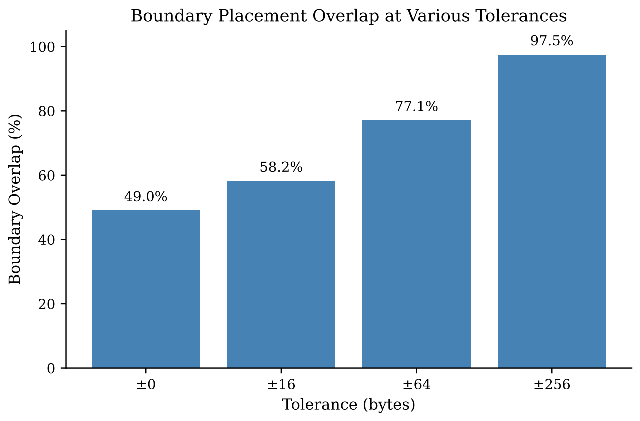

Figure 1: Streaming vs. offline boundary overlap at varying tolerances. Overlap exceeds 97% at ±256 bytes.

based on representational strain signals, and a hybrid policy that combines stability-driven base chunks with localized overlapping micro-chunks in edit regions. Retrieval experiments compare ANN-only retrieval with representational reranking based on stored stability diagnostics.

3 Results

3.1 Stability-Driven Chunking Effects (v1.2.0)

Stability-driven chunking produces substantially fewer chunks with larger mean size compared to fixed-size heuristics (Table 1). Boundaries are placed at points of elevated representational strain rather than arbitrary length thresholds, preserving coherent regions as single units. This reduces index size and ingest-time maintenance overhead, demonstrating that stability signals materially influence segmentation behavior.

3.2 Streaming Parity with Offline Segmentation (v1.3.0)

Under bounded lookahead, streaming execution closely approximates offline stability-driven segmentation. Boundary overlap exceeds 97% within a ±256-byte tolerance, with deviations localized near the commit horizon (Figure 1). Chunk size distributions remain well-behaved, confirming compatibility with live ingestion pipelines.

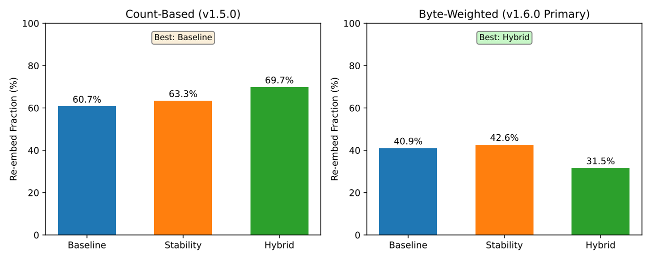

Figure 2: Count-based vs. byte-weighted mutation metrics. Hybrid chunking appears worst under count-based metrics but best under byte-weighted cost.

3.3 Negative Result: Mutation Sensitivity Under Count-Based Metrics (v1.4.0)

When evaluated under localized edits, stability-driven chunking exhibits higher apparent mutation sensitivity than fixed-size overlapping chunking when measured using count-based metrics (Table 2). Baseline chunking preserves more unchanged chunks and higher neighbor overlap over time. This negative result reflects the larger granularity of stability-driven chunks, where any edit invalidates the entire chunk.

3.4 Hybrid Chunking and Localization of Change (v1.5.0)

To address mutation sensitivity, we introduce a hybrid policy that applies stability-driven chunking globally while re-chunking only local edit windows using overlapping micro-chunks. Hybrid chunking localizes 80.7% of re-embeds to edit regions and partially recovers temporal stability relative to stability-only chunking (Table 3). However, under raw chunk-count metrics, hybrid chunking appears worse due to increased micro-chunk counts.

3.5 Metric Correction: Cost-Weighted Mutation Analysis (v1.6.0)

Count-based mutation metrics implicitly treat all chunks as equal cost, mischaracterizing hybrid policies that intentionally vary granularity. When mutation cost is measured in bytes rather than chunk counts, the ranking reverses (Table 4). Hybrid chunking reduces effective mutation cost by 54% relative to the baseline, resolving the apparent contradiction observed under count-based metrics.

Table 1: Chunking Method Comparison (v1.1.0 Baseline vs Stability-Driven)

Metric	**Baseline (Fixed)**	**Stability-Driven**
Chunk Statistics		
Total Chunks	867	567
Mean Chunk Size (bytes)	1,978	2,990
Chunk Reduction	—	−34.6%
Maintenance Cost (vs Baseline)		
Re-embed Fraction	—	65.4%
Index Rebuild Required	—	Yes
Configuration		
Target Chunk Size (bytes)	2,048	2,048
Min/Max Bounds (bytes)	256–4,096	256–4,096
Overlap (bytes)	64	64

Table 2: Temporal Stability Metrics (commit_horizon=1024 bytes)

Metric	**Baseline**	**Stability**
Embedding Drift		
Mean L2	0.0000	0.0000
P90 L2	0.0000	0.0000
ANN Neighbor Churn		
Top-k Overlap	0.531	0.322
Maintenance Cost		
Re-embed Fraction	52.5%	63.4%
Chunks Added	95	71
Chunks Unchanged	86	41
Boundary Effects		
Drift (Near Boundary)	0.0000	0.0000
Drift (Interior)	0.0000	0.0000

Table 3: Hybrid Chunking Comparison (commit_horizon=1024 bytes)

Metric	**Baseline**	**Stability**	**Hybrid**
Mutation Tolerance			
Re-embed Fraction	60.7%	63.3%	69.7%
Chunks Unchanged	72	40	36
Neighbor Churn			
Top-k Overlap	0.428	0.318	0.347
Structural Overhead			
Total Chunks	92	55	67
Mean Chunk Size	1683	2742	711
Micro-chunk Fraction	–	–	52.2%
Localization (Hybrid only)			
Localization Efficiency	–	–	80.7%

Table 4: Cost-Weighted Mutation Metrics (v1.6.0 Re-analysis)

Metric	**Baseline**	**Stability**	**Hybrid**
Count-Based (v1.5.0)			
Re-embed Fraction (%)	60.7	63.3	69.7
Added Chunks	111	69	83
Byte-Weighted (v1.6.0 Primary)			
Re-embed Bytes (KB)	182.0	185.6	82.9
Re-embed Fraction (%)	40.9	42.6	31.5
Localization			
Localization Efficiency	61.3%	94.2%	80.7%
Structural			
Total Chunks	92	55	67
Mean Chunk Size (B)	1683	2742	711
Micro-chunk Fraction	–	–	52.2%
Neighbor Churn			
Top-k Overlap	0.428	0.318	0.347

Byte-weighted re-embed cost reverses ranking: hybrid achieves 54% lower mutation cost than baseline despite higher chunk count.

3.6 Representational Reranking Improves Retrieval Robustness (v2.0.0)

Finally, we evaluate representational reranking as a retrieval-layer modification atop the cost-corrected hybrid baseline. Representational reranking yields modest improvements in mean reformulation overlap (+2.3%) and substantially improves worst-case behavior, increasing P10 overlap by 14.3% (Table 5). A random reranking control collapses performance, confirming that gains arise from representational signals. Importantly, reranking does not affect mutation cost or determinism.

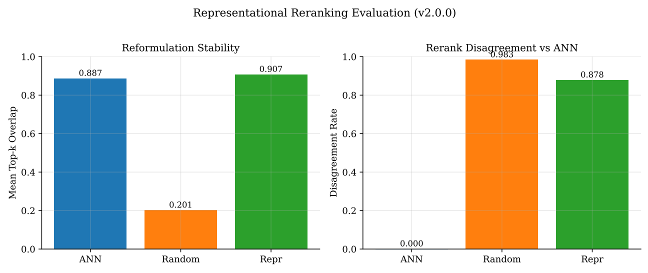

Figure 3: Reformulation overlap comparison across retrieval modes. Representational reranking improves over ANN baseline; random control demonstrates that stability gains are signal-dependent.

4 Discussion

Across all experiments, three competing axes emerge: representational coherence, mutation tolerance, and retrieval robustness. Stability-driven chunking optimizes coherence; fixed-size overlapping chunking optimizes mutation tolerance; representational reranking improves robustness in the tails. No single mechanism optimizes all three simultaneously.

A central finding is that metric choice fundamentally alters system evaluation. Count-based mutation metrics overstate the cost of localized micro-chunking, while byte-weighted metrics align with actual embedding and index maintenance costs. Under cost-aligned metrics, hybrid chunking dominates baseline heuristics.

Hybrid policies do not eliminate trade-offs but localize them, making system behavior predictable and bounded under updates. Representational reranking further improves robustness without attempting to replace similarity-based retrieval. Together, these results suggest that dynamic

Table 5: Representational Reranking Results (v2.0.0)

Metric	**ANN**	**Random**	**Repr**
Retrieval Stability			
Reformulation Overlap	0.887	0.201	0.907
Overlap P10	0.700	0.100	0.800
Boundary Robustness			
Structural Boundary Frac	0.000	0.000	0.000
Rerank Characteristics			
Disagreement Rate	–	0.983	0.878
Mean Rank Shift	–	3.3	2.4
Max Rank Shift	–	9	9

Representational reranking improves reformulation overlap by 2.3% over ANN baseline.

vector databases should decouple segmentation, mutation handling, and retrieval ranking, and should evaluate systems using cost-aligned metrics rather than static benchmarks.

5 Related Work

The Platonic Representation Hypothesis [1] suggests that representations across modalities converge toward shared geometric structure. Our stability diagnostics operationalize a subset of these ideas for chunking and retrieval, though we use proxy signals rather than full representational analysis.

6 Limitations

This study uses proxy stability diagnostics rather than full representational signals, limiting the magnitude of reranking gains. Experiments are conducted at moderate scale and focus on controlled update patterns. Extending these methods with richer diagnostics or learned components is left to future work.

7 Conclusion

We present a comprehensive empirical study of chunking and retrieval strategies for dynamic vector databases. By disentangling structural coherence, mutation tolerance, and retrieval robustness, and by correcting misaligned evaluation metrics, we show that hybrid chunking combined with representational reranking provides a principled and robust design for evolving vector stores. These findings highlight the importance of stability-aware design and cost-aligned evaluation in real-world vector retrieval systems.

Acknowledgments

These diagnostics were developed as part of a broader internal CURV Institute representation-first framework; only the operational components relevant to the experiments are used here.

References

[1] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

A Artifact Appendix

A.1 Repository

All code, data generation scripts, and experiment manifests are available at:

`https://github.com/curv-institute/curv-embedding`

A.2 Requirements

- Python $\geq$ 3.12
- Dependencies declared via PEP 723 inline headers
- Executable via `uv run`

A.3 Reproducing Results

```
git clone https://github.com/curv-institute/curv-embedding
cd curv-embedding
uv run scripts/reproduce.py --run-name <run_name>
```

A.4 Manifest Format

Each run produces:

- `manifest.json`: Configuration snapshot and seeds
- `metrics.jsonl`: Raw metric measurements
- `summary.json`: Aggregated statistics
- `meta.sqlite`: Full chunk and event audit log

A.5 Compute Resources

All experiments were run on a single workstation with an Intel Core i7 processor and 32GB RAM. Total runtime for the full experiment suite is under 10 minutes.