Octree-Indexed Sentence Embeddings for LLM Memory Retrieval

Rob Rohan* Claude Sonnet 4.6
Anthropic

May 2026 *(draft)*

Abstract

This study investigates spatial indexing of sentence embeddings for in-process LLM memory retrieval. We project 384-dimensional embeddings from all-MiniLM-L6-v2 to lower dimensions using PCA, UMAP, and t-SNE, then index the projected coordinates in a spatial tree. Queries navigate the tree to retrieve a candidate set, which is re-ranked by cosine similarity on the full vectors.

On a 25-sentence pilot corpus, PCA 16D achieves 100% recall@5 at ∼35 μs transform latency. On 120k AG News headlines, PCA routing fails entirely (under 1% recall) because the top principal components capture distributional structure rather than semantic similarity. UMAP achieves 32.8% recall@5 at 24 ms per query, which is 21 times slower than brute-force cosine at this corpus size (1.15 ms, 100% recall). The primary finding is that for corpora below approximately one million entries, brute-force cosine search is likely faster and more accurate than any indexed pipeline evaluated here. The practical value of the 3D octree is as a visualisation tool: the index structure renders directly as an interactive scatter plot of the memory store's semantic content.

Keywords: sentence embeddings, approximate nearest neighbour search, octree, KD-tree, PCA, UMAP, t-SNE, LLM memory

1 Introduction

LLM harnesses that maintain context across sessions need a mechanism to store and retrieve past observations. The standard approach uses an external vector database such as Chroma, Pinecone, or Weaviate. These systems add a separate process or network connection, which increases complexity for a self-contained application.

This study explores a simpler alternative: project sentence embeddings to a low-dimensional space and index them in a spatial tree. The full-dimensional vectors are stored as payloads on the tree leaves. At query time, the tree provides a candidate set, which is re-ranked by cosine similarity on the full vectors.

A key requirement for an LLM memory store is that new entries can be added at any time without rebuilding the index. An octree supports this directly: inserting a new point requires only a single traversal from root to leaf. Methods such as HNSW and IVF do not support incremental insertion cleanly. HNSW requires connecting the new node into the proximity graph, and IVF may require re-clustering if the new point falls far from existing centroids. For a memory store that updates continuously as the LLM operates, this property matters.

A 3D spatial tree can also be rendered directly as a scatter plot of the memory store. A practitioner can inspect which entries the model considers semantically similar and trace why a particular query retrieved the results it did. This property is not available with HNSW or inverted file index systems, where the index structure is not directly inspectable.

*Correspondence: rob@therohans.com

This research addresses the following questions:

- RQ1: Does PCA-projected spatial indexing achieve competitive recall for sentence embedding retrieval at realistic corpus sizes?
- RQ2: Do nonlinear projections (UMAP, t-SNE) improve routing quality over linear PCA?
- RQ3: At what corpus size does brute-force cosine search become slower than a spatial index?

The answer to RQ1 is no at 120k sentences. The answer to RQ2 is yes, UMAP routes more accurately than PCA, but not accurately enough to justify its cost at this scale. RQ3 remains unanswered; we were unable to identify a crossover point within the scope of this study.

2 Background

2.1 Sentence Embeddings

Dense sentence representations from transformer models are the standard input to semantic search systems. Reimers and Gurevych [10] introduced Sentence-BERT, which produces fixed-length embeddings by mean-pooling token representations from a siamese BERT network [4] fine-tuned on natural language inference pairs. The all-MiniLM-L6-v2 model [12] used in this study is a knowledge-distilled sentence-transformer fine-tune of MiniLM, producing 384-dimensional embeddings at lower inference cost than the full BERT model.

2.2 Approximate Nearest Neighbour Search

ANN search over dense vectors is a well-studied problem. The main approaches are:

Graph-based (HNSW). Hierarchical Navigable Small World graphs [7] navigate a multi-layer proximity graph greedily at query time. HNSW achieves strong recall/latency results on standard benchmarks [1]. The index structure is not directly visualisable.

Inverted File Index (IVF). IVF methods [5] partition the embedding space into Voronoi cells via k-means clustering. A query is routed to the nearest centroid and candidates are re-ranked within that cell. This is structurally closest to our approach. The difference is that IVF uses learned centroids and our method uses fixed axis-aligned spatial splits.

Random Projection Trees (ANNOY). Bernhardsson [3] build a forest of binary trees by splitting with random hyperplanes. The structure is similar to ours without an explicit dimensionality reduction step.

Spatial Trees. KD-trees [2] partition d-dimensional space using axis-aligned splits. Octrees are the 3D case, splitting space into eight equal octants at each level. Both degrade toward brute force above roughly $d \approx 20$ due to the curse of dimensionality, which motivates the projection step.

2.3 Dimensionality Reduction

PCA finds the linear projection that maximises variance. The transform is a matrix multiply, making it fast to apply to new points at query time. UMAP [8] preserves local neighbourhood structure and tends to produce cleaner cluster separation than PCA. Transforming a new point requires an optimisation step rather than a matrix multiply. Section 4 quantifies the latency

difference. t-SNE [11] also preserves local structure and supports out-of-sample transform via the `openTSNE` implementation [9].

2.4 Dense Retrieval

Dense Passage Retrieval [6] showed that bi-encoder retrieval outperforms BM25 for open-domain question answering when trained on sufficient data. Our setting differs: we use a general-purpose sentence encoder and our corpus is a personal memory store (hundreds to thousands of entries) rather than a large document collection.

3 Method

3.1 Pipeline

The retrieval pipeline has two stages.

At index time, each memory m_i is embedded to a 384-dimensional vector $\mathbf{v}_i$. The vector is projected to d dimensions: $\mathbf{p}_i = \phi(\mathbf{v}_i)$, where ϕ is PCA, UMAP, or t-SNE. The projected coordinate $\mathbf{p}_i$ is inserted into a spatial tree with $(\mathbf{v}_i, \text{text}_i)$ stored as the leaf payload.

At query time, the query is embedded to $\mathbf{v}_q$, projected to $\mathbf{p}_q$, and the tree is navigated to retrieve a candidate set $\mathcal{C}$. Candidates are re-ranked by cosine similarity on the full vectors and the top k are returned.

3.2 Spatial Trees

3D Octree. The root node spans the bounding cube of all projected points. Each internal node splits into eight equal octants. Leaf nodes store up to `MAX_POINTS` entries. At query time, the tree navigates to the leaf containing the query point. If the leaf holds fewer than k entries, the search walks up to the parent and collects all points in the parent's subtree, repeating until at least k candidates are found. Candidates are then re-ranked by cosine similarity on the full vectors.

To prevent infinite subdivision when many points project to nearly identical 3D coordinates, we cap the tree depth at 20 levels.

KD-Tree. For $d > 3$ we use a standard KD-tree (scikit-learn `KDTree`). The tree returns $\ell \cdot k$ candidates ($\ell = 3$ in all experiments), which are cosine re-ranked. The KD-tree does not use a walk-up; the expansion factor ℓ serves the same purpose by retrieving a larger candidate set upfront.

Hyperparameter coupling. UMAP and t-SNE with `n_neighbors` $= n$ arrange each point so its n nearest semantic neighbours are spatially close in the projected space. Setting `MAX_POINTS` $= n$ ensures a single leaf holds one complete preserved neighbourhood. If `MAX_POINTS` $< n$, the tree subdivides before the neighbourhood is intact. We use `n_neighbors` $= 15$ and `MAX_POINTS` $= 15$ throughout the scale experiments.

3.3 Embedding Model

We use `sentence-transformers/all-MiniLM-L6-v2` [12], producing 384-dimensional sentence embeddings. The retrieval method is not specific to this model.

Algorithm 1 Two-stage memory retrieval (octree variant)

Require: Query text q, projection $\phi : \mathbb{R}^{384} \to \mathbb{R}^d$, octree T, target k

$\mathbf{v}_q \leftarrow \textsc{Embed}(q)$

$\mathbf{p}_q \leftarrow \phi(\mathbf{v}_q)$

$L \leftarrow \textsc{Navigate}(T, \mathbf{p}_q)$ ▷ walk root to leaf

$\mathcal{C} \leftarrow \textsc{Expand}(L, k)$ ▷ walk up until $|\mathcal{C}| \geq k$

$\mathcal{R} \leftarrow \textsc{CosineRank}(\mathcal{C}, \mathbf{v}_q)$ **return** $\mathcal{R}[1{:}k]$

4 Experiments

4.1 Pilot Study

4.1.1 Setup

We constructed a 25-sentence corpus spanning five topics (machine learning, memory/recall, databases, systems programming, and nature), five sentences per topic. Ground-truth nearest neighbours were computed by brute-force cosine similarity in the full 384-dimensional space. Three projections were evaluated: PCA 3D with an octree index, UMAP 3D (`n_neighbors`=10, `min_dist`=0.1, cosine metric), and PCA 16D with a KD-tree.

4.1.2 Results

Table 1 shows recall@5 and transform latency.

Table 1: Pilot results on a 25-sentence corpus.

Method	Variance explained	Recall@5	Transform latency
PCA 3D	26.9%	90.4%	∼35 μs
UMAP 3D	—	94.4%	∼939 ms
PCA 16D	83.9%	100.0%	∼35 μs

PCA 16D achieves perfect recall at the same latency as PCA 3D. UMAP 3D recall is marginally higher than PCA 3D, but its transform latency of 939 ms is impractical for a live query pipeline. These results on the pilot corpus suggested PCA 16D was a viable approach. The scale experiments did not support that conclusion.

4.2 Scale Experiments: AG News 120k

4.2.1 Setup

We embedded all 120,000 titles from the AG News training split [13] using all-MiniLM-L6-v2. Embeddings were pre-computed offline in batches of 256 to avoid in-notebook memory exhaustion. The corpus covers four topic classes: World, Sports, Business, and Sci-Tech. Held-out queries are 50 titles sampled at random from the 7,600-title test split.

Ground truth is brute-force cosine similarity on full 384-dimensional vectors. Recall@5 is the fraction of each query's 5 true nearest neighbours returned after cosine re-ranking. We evaluated six pipelines: brute-force cosine, PCA 3D with octree, PCA 8D with KD-tree, PCA 16D with KD-tree, UMAP 3D with octree, and t-SNE 3D with octree. For t-SNE we used `openTSNE` [9] with the Barnes-Hut gradient method, which is required for 3D embeddings.

4.2.2 PCA Variance Collapse

Table 2 shows PCA 3D variance explained as corpus size grows.

Table 2: PCA 3D variance explained by corpus size (AG News titles).

Corpus size	Variance explained
25	26.9%
200	12.0%
2,000	9.3%
120,000	8.9%

The collapse is expected. With a small, topically homogeneous corpus, the top principal components capture genuine semantic structure. With a large, diverse corpus, they are dominated by vocabulary frequency and other distributional properties that do not separate sentences by meaning. At 8.9% variance explained, the 3D projection is not a useful routing key.

4.2.3 Results

Table 3 shows recall@5 and mean query latency.

Table 3: Recall@5 and mean query latency, 120k AG News titles, 50 held-out queries.

Pipeline	Recall@5	Mean latency
Brute-force cosine	100.0%	1.15 ms
PCA 3D + octree	0.8%	0.10 ms
PCA 8D + KD-tree	0.8%	0.14 ms
PCA 16D + KD-tree	0.0%	0.15 ms
UMAP 3D + octree	32.8%	23.98 ms
t-SNE 3D + octree	19.2%	128.54 ms

Brute-force cosine is both faster and more accurate than every indexed pipeline. UMAP 3D achieves the highest recall among the indexed methods at 32.8%, but at 24 ms per query it is 21 times slower than brute-force. t-SNE has lower recall than UMAP at a higher latency and provides no practical advantage over UMAP.

PCA fails at every dimensionality tested. This is consistent with the variance collapse in Table 2: points that are semantically similar do not project to the same spatial region, so the routing step cannot retrieve them.

The corpus size at which an indexed pipeline becomes faster than brute-force was not determined. Brute-force cosine over 120k embeddings costs 1.15 ms. An indexed pipeline would need both lower latency and useful recall to be a practical replacement.

5 Discussion

5.1 Relationship to IVF

IVF and this approach share the same coarse-to-fine structure. The difference is the partitioning function. IVF uses k-means centroids that adapt to the embedding distribution. The octree uses fixed axis-aligned splits, which support incremental insertion without re-clustering. Fixed splits produce unbalanced cells when embeddings cluster unevenly, which is common with sentence embeddings.

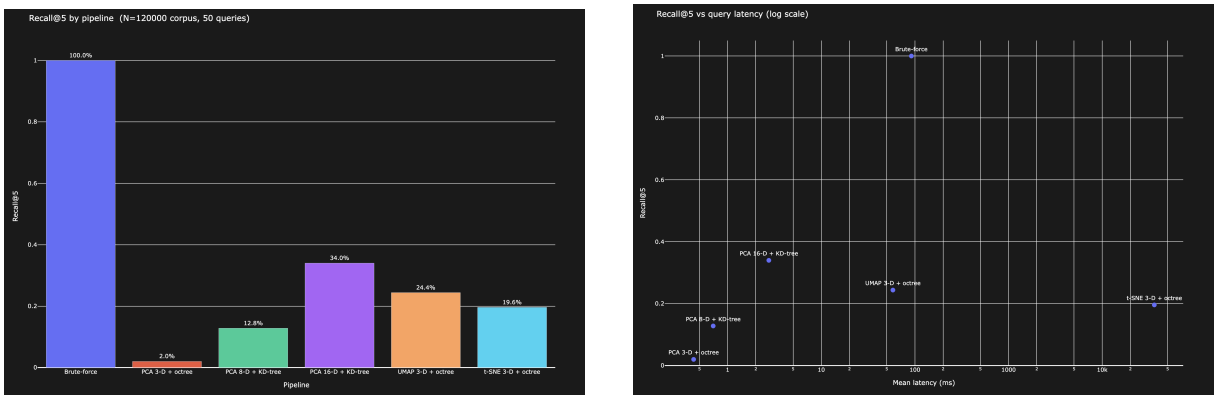

Figure 1: Left: recall@5 by pipeline on 120k AG News titles. Right: recall@5 versus mean query latency. Brute-force cosine (top-right) dominates on both axes at this corpus size.

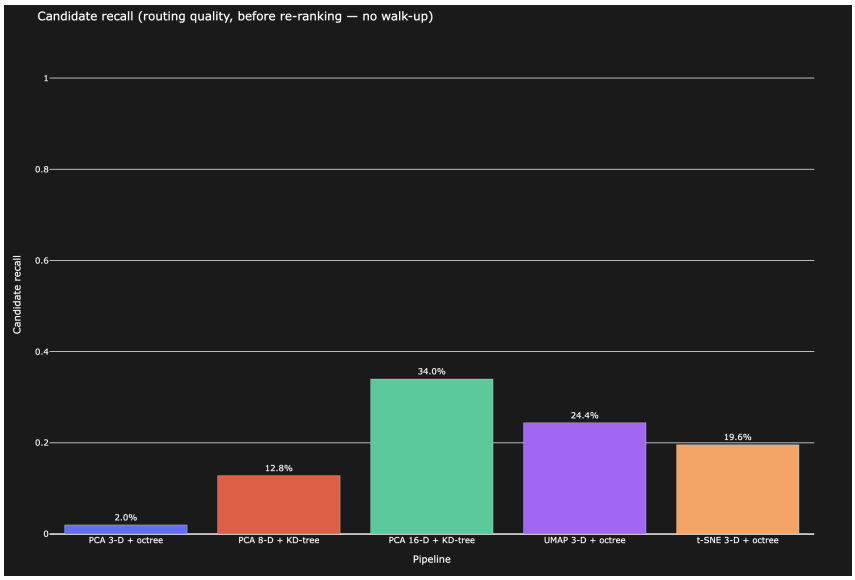

Figure 2: Candidate recall by pipeline — the fraction of true top-5 neighbours present in the candidate set before cosine re-ranking. Low candidate recall is the primary failure mode: the routing step does not retrieve the right points, so re-ranking cannot recover them.

The incremental insertion property is the primary motivation for choosing a spatial tree over IVF or HNSW for an LLM memory store. A memory store that updates as the model operates cannot afford to pause and rebuild the index on each insertion. An octree insert is a single root-to-leaf traversal: $O(\log N)$ with no global side-effects. HNSW insertion requires bidirectional edge updates across multiple graph layers, and IVF insertion requires assigning the new point to an existing centroid, which degrades in accuracy over time if the data distribution shifts. The octree avoids both of these problems, at the cost of the routing accuracy limitations documented in Section 4.2.

5.2 The Octree as a Visualisation Tool

The octree at $d = 3$ is constrained to three dimensions. That constraint is also what makes it useful for visualisation. Each internal node is an axis-aligned bounding box in a coordinate system defined by the top three principal components of the embedding space. The structure renders directly as a 3D scatter plot. A practitioner can navigate it interactively, inspect which memories cluster together, and trace why a query returned the candidates it did.

KD-trees handle arbitrary d but do not produce this kind of spatial visualisation. HNSW and IVF do not expose their index structure in a form that can be rendered geometrically.

We used the 3D octree visualisation during development to diagnose projection quality. When PCA was the projection, the octree showed that semantically unrelated sentences were landing in the same spatial cells. This was a direct indication of the routing failure before the recall metrics confirmed it.

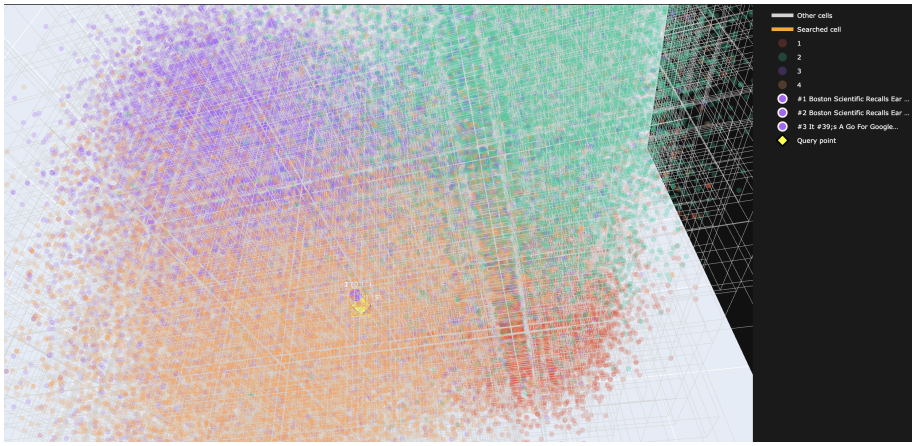

Figure 3: 3D octree query example. The query point (star) navigates to its leaf node. Candidate points from the leaf's subtree (orange) are cosine re-ranked to produce the final result. The scatter plot is the entire memory store projected via UMAP; colour indicates topic cluster.

6 Conclusion

This study implemented and evaluated a two-stage retrieval pipeline for LLM memory stores. Sentence embeddings are projected to low-dimensional space using PCA, UMAP, or t-SNE and indexed in a spatial tree. Candidates are retrieved by tree navigation and re-ranked by cosine similarity on the full vectors.

On a 25-sentence pilot corpus, PCA 16D + KD-tree achieves 100% recall@5 at 35 μs transform latency. At 120k AG News titles, all PCA pipelines fail as routers due to variance collapse. UMAP achieves 32.8% recall but is slower than brute-force cosine at this corpus size. The crossover point where an index becomes worthwhile was not found within the scope of this study.

The 3D octree is useful as a visualisation tool. The index structure renders as an interactive scatter plot that shows the semantic structure of the memory store at a glance. This is not a property that HNSW or IVF-based systems provide.

Future work could include:

- A dimensionality sweep at higher PCA dimensions (32D, 64D, 128D) to find the point at which linear projection captures enough semantic variance to be a useful routing key.
- Comparison against HNSW at the same corpus sizes to quantify the recall gap.
- Measurement of the corpus size at which indexed retrieval first becomes faster than brute-force cosine for UMAP-projected coordinates.
- Investigation of whether UMAP recall improves or degrades at corpus sizes beyond 120k.

References

[1] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87, 2020. doi: 10.1016/j.is.2019.02.006. URL https://arxiv.org/abs/1807.05614. arXiv:1807.05614. Standardised recall/latency benchmark suite across HNSW, FAISS, ANNOY, ScaNN on GloVe-100 and other datasets. See also https://ann-benchmarks.com.

[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. doi: 10.1145/361002.361007.

[3] Erik Bernhardsson. ANNOY: Approximate nearest neighbors in C++/Python. GitHub repository, 2015. URL https://github.com/spotify/annoy. No formal peer-reviewed paper exists. This is a software library released by Spotify. Cite as software.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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 (NAACL-HLT)*, volume 1, pages 4171–4186. Association for Computational Linguistics, 2019. URL https://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[5] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. URL https://arxiv.org/abs/1702.08734. arXiv:1702.08734. VERIFY publication year (some sources say 2019).

[6] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics, 2020. URL https://arxiv.org/abs/2004.04906. arXiv:2004.04906. Cited as representative two-stage retrieval (coarse BM25 + dense re-ranker).

[7] Yury A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. URL https://arxiv.org/abs/1603.09320. arXiv:1603.09320. Originally posted 2016; published in TPAMI 2020.

[8] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint*, 2018. URL https://arxiv.org/abs/1802.03426. arXiv:1802.03426. Widely cited as preprint; no primary peer-reviewed venue as of 2025.

[9] Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. openTSNE: A modular Python library for t-SNE dimensionality reduction and embedding. *Journal of Statistical Software*, 109(3): 1–30, 2024. doi: 10.18637/jss.v109.i03.

[10] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. URL https://arxiv.org/abs/1908.10084. arXiv:1908.10084.

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[12] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. URL `https://arxiv.org/abs/2002.10957`. arXiv:2002.10957. The all-MiniLM-L6-v2 model used in this work is a sentence-transformer fine-tune of MiniLM; see also [10] and the model card at https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

[13] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015. URL `https://arxiv.org/abs/1509.01626`. arXiv:1509.01626. Introduces the AG News corpus (4-class news topic classification, 120k train / 7.6k test). Used here as a medium-scale benchmark for recall@K evaluation of the PCA+KD-tree index.