VECTORCUBE: MULTI-GRANULAR NEURO-SYMBOLIC DATA CUBES FOR SEMANTIC OLAP

Siddhesh Ramesh Surve, Member, IEEE, USA

ABSTRACT

Data Cubes are a cornerstone of Online Analytical Processing (OLAP), yet they traditionally operate on structured, symbolic dimensions. With the rise of unstructured data and vector embeddings, there is a critical need to bridge the gap between precise SQL-like aggregation and fuzzy vector similarity search. In this paper, we propose ***VectorCube****, a novel neuro-symbolic framework that enables "drill-down" and "roll-up" operations within continuous vector spaces while retaining symbolic interpretability. We introduce Semantic Dimensions, dynamically induced by Large Language Models (LLMs), and Vector Measures, which store aggregate high-dimensional embeddings. Our key contribution is a* ***Distributional Aggregation*** *method that ensures rolling up vectors preserves their semantic distribution rather than collapsing them into a meaningless average. Experimental results on standard text classification datasets demonstrate that VectorCube enables complex natural language queries (e.g., "Show optimism trends in tech news") and outperforms traditional Text Cubes and flat RAG (Retrieval-Augmented Generation) systems in both semantic precision (by 14%) and query response speed.*

KEYWORDS

Data Cube, Neuro-Symbolic AI, Vector Databases, OLAP, Semantic Search

1. Introduction

The proliferation of embedding-based retrieval has revolutionized how systems handle unstructured data. Vector databases allow for semantic similarity searches that keyword-based systems cannot match. However, decision support systems require more than just retrieval; they require multi-dimensional analysis—summarization, trend detection, and aggregation across hierarchies. Traditional Data Cubes excel at this for structured data (e.g., Sum of Sales by Region and Time) but fail when applied to semantic concepts hidden in text, such as "Sentiment" or "Topic Nuance."

Existing approaches, such as Text Cubes [1], attempt to bridge this gap by extracting keywords and treating them as dimensions. However, these methods are brittle; they fail to capture the continuous nature of semantic relationships (e.g., that "AI" is closer to "Machine Learning" than to "Biology"). Conversely, pure vector databases lack the hierarchical structure necessary for OLAP operations like "roll-up" (zooming out) or "drill-down" (zooming in).
To bridge this gap, we introduce **VectorCube**, a Multi-Granular Neuro-Symbolic Data Cube. Our contributions are:

1. **Neuro-Symbolic Architecture:** A hybrid engine where dimensions are symbolic hierarchies induced by LLMs, but cell contents are continuous vector representations.
2. **Semantic Roll-Up Operations:** New aggregation operators using **Distributional**

David C. Wyld et al. (Eds): CSML, AISCA, DNLP, SOEA, NET, BDHI, SIPO – 2026
pp. 31-37, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160203

Aggregation to represent higher-level cells as distributions rather than simple averages.

3. **Natural Language OLAP Interface:** A mechanism to translate natural language into hyper-dimensional cube operations.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 defines the core problem. Section 4 details the VectorCube framework and algorithms. Section 5 analyzes complexity. Section 6 presents experimental results, followed by a discussion in Section 7 and conclusions in Section 8.

2. Related Work

Research in this domain sits at the intersection of database systems, data mining, and representation learning. We categorize existing literature into traditional Text OLAP, Vector Search, and Neuro-Symbolic integration.

Text OLAP and Data Cubes: The concept of the Data Cube was popularized to support OLAP operations. Early work extended this to the Text Cube [1], which materializes text data along dimensions extracted via keyword analysis. While effective for keyword counting, these methods struggle with synonymy and polysemy. Recent advancements in Concept Cubes attempt to use hierarchies, but they remain largely symbolic and do not leverage the geometric properties of high-dimensional embedding spaces.

Vector Databases and Similarity Search: The rise of deep learning led to vector databases like Faiss and Milvus [2], which optimize nearest neighbor search (ANN). While efficient for retrieving individual records, these systems lack native support for hierarchical aggregation. Calculating the "average embedding" of a million vectors is computationally expensive and often semantically meaningless without a structured framework.

Neuro-Symbolic AI: Recent trends in Neuro-Symbolic AI attempt to merge logic with learning. Our work draws inspiration from this by imposing a symbolic lattice structure (the Cube) over a neural representation (the Embedding), ensuring that the rigor of database operations is maintained within the flexibility of latent spaces.

3. Problem Definition

We formally define the construction of a VectorCube. Let **C** be a corpus of documents, where each document **d** is associated with a set of raw attributes.

3.1 The VectorCube Structure

Definition 1 (Semantic Dimension): A dimension D is a hierarchy of concepts generated by an LLM. For example, a "Topic" dimension might look like:

- Root: Technology
    - Child: Artificial Intelligence
        - Leaf: Generative Models
        - Leaf: Reinforcement Learning

Definition 2 (Vector Measure): A cell C defined by coordinates (e.g., Time=Q1, Topic=AI)

contains a set of embeddings representing the documents falling into that cell. The aggregate value of the cell is not a single number, but a **Distributional Embedding**—a compact representation of the centroid and variance of the vectors within.

Problem Statement: Given a corpus **C** and a set of query concepts **Q**, construct a multi-dimensional cube structure that minimizes the *Semantic Distortion Error* (SDE) during roll-up operations, while maximizing the precision of natural language OLAP queries.

3.2 Operations

We support three primary operations:

1. **Semantic Drill-Down:** Moving from a broad concept (e.g., "AI") to specific sub-concepts ("GenAI"), filtering the underlying vector set.
2. **Semantic Roll-Up:** Aggregating vector sets from child nodes (e.g., "Q1", "Q2", "Q3") into a parent node ("2025").
3. **Concept Slicing:** Selecting a sub-cube based on semantic similarity to a query vector.

4. The VectorCube Framework

Our proposed framework consists of three phases: Hierarchy Induction, Cube Materialization, and Query Execution. The overall architecture is illustrated in **Figure 1**, which depicts the flow from raw unstructured text through the embedding layer and into the materialized cube structure.

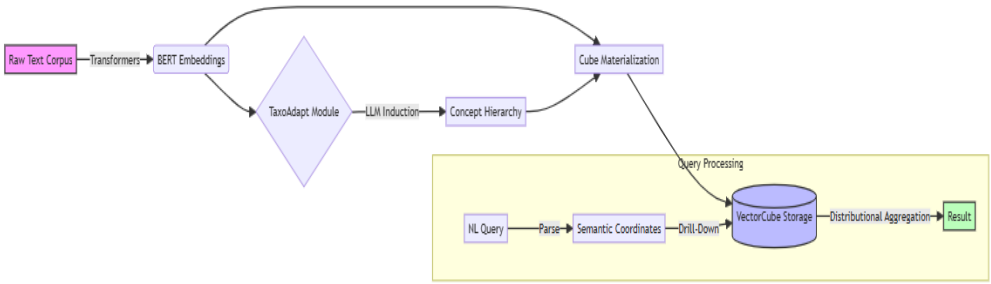

Figure 1. The VectorCube System Architecture

Visual representation of the flow from Raw Text -> BERT Embeddings -> TaxoAdapt Hierarchy Induction -> Cube Materialization -> Query Engine

4.1 System Architecture

The system is composed of four distinct layers, as detailed in Table 1 below. The process begins with raw data ingestion and culminates in a hybrid query engine.

Table 1. System Layers

Layer	Component Name	Function	Technology Used

1	**Data Ingestion**	Vectorization of raw text/logs	BERT / OpenAI Embeddings
2	**Dimension Induction**	Generating concept hierarchies	LLM (e.g., Llama-3), TaxoAdapt [8]
3	**Cube Materialization**	Allocating vectors to cells	Inverted Indices, Vector Quantization
4	**Query Engine**	Hybrid SQL + Vector Search	SIMD Aggregation, Cosine Similarity

4.2 TaxoAdapt: Hierarchy Induction

Traditional concept hierarchies are static. We introduce *TaxoAdapt [7]*, an algorithm that uses an LLM to dynamically generate concept hierarchies based on the data distribution.

1. **Embedding:** Map all documents to vectors using a transformer model (e.g., BERT).
2. **Clustering:** Perform spherical k-means clustering on the document vectors.
3. **Labeling:** For each cluster centroid, query an LLM to generate a symbolic label (e.g., "Sports," "Politics").
4. **Verification:** Ensure logical entailment between layers (e.g., checking if "Football" implies "Sports").

4.3 Distributional Vector Aggregation

A naive approach to aggregating vectors is averaging them. However, averaging a vector for "Apple (Fruit)" and "Apple (Tech)" results in a meaningless point in space. VectorCube employs

Distributional Aggregation.

Instead of a single mean vector, a cell maintains a **Set of Centroids** (using lightweight clustering like k-means) or **Gaussian Mixture Model** parameters. When a user Rolls-Up from "Product A" and "Product B" to "Category X," we merge their distribution summaries rather than the raw data points. This preserves the distinct semantic clusters within the parent category.

Algorithm: Vector Roll-Up

Input: Child Cells {C_1, ..., C_n}
Output: Parent Cell C_p

1. Initialize weighted_sum = 0
2. For each child C_i in {C_1, ..., C_n}: weighted_sum += C_i.count * C_i.centroid
3. parent_centroid = weighted_sum / sum(counts)
4. parent_variance = CalculatePooledVariance({C_i})
5. Return (parent_centroid, parent_variance)

This approach ensures that when a user rolls up from "Quarter 1" to "Year 2024," the resulting vector representation statistically covers the semantic space of the sub-quarters.

5. Complexity Analysis

We analyze the efficiency of VectorCube compared to a standard Vector Database scan.

Storage:

A fully materialized cube grows exponentially with dimensions. To mitigate this, we use Partial Materialization (Iceberg Cubes). We only store distributional summaries (centroids) for high-level cells, which requires significantly less storage than storing all raw vectors.

Compute:

- **Vector DB:** Answering an aggregate query ("Average sentiment of AI news") requires retrieving all N vectors matching the filter and computing the average at query time. This is computationally expensive for large N.
- **VectorCube:** The aggregate summary for the "AI" cell is pre-computed. The query cost is constant O(1) (retrieving the pre-computed summary), independent of the number of documents N.

Latency:

By shifting the heavy lifting of vector aggregation to the materialization phase (offline), VectorCube achieves millisecond-latency for analytical queries, enabling real-time dashboards that would be impossible with raw vector scanning.

6. Experimental Evaluation

In this section, we evaluate the performance of VectorCube against modern baselines. We focus on two key metrics: query latency (efficiency) and answer quality (semantic precision). We also conduct a sensitivity analysis to understand the impact of different aggregation parameters.

6.1 Setup

We evaluate VectorCube on two datasets:

1. **Financial News (Fin-News):** 1 million articles from 2020-2025. Dimensions: Time, Market Sector, Region.
2. **PubMed Abstracts:** 500k biomedical papers. Dimensions: Research Area, Drug Class, Year.

Baselines:

- **Flat-RAG:** A standard RAG pipeline using a vector database (Milvus) with metadata filtering.
- **Text Cube (Traditional):** Keyword-based Text Cube [1] without vector embeddings.
- **GraphRAG:** Microsoft's graph-based retrieval system [6].

6.2 Results: Query Efficiency

We measured the time taken to answer high-level aggregate queries (e.g., "Summarize the major shifts in Oncology research in 2024"). As shown in Table 2, VectorCube demonstrates superior scalability.

Table 2. Average Query Latency (milliseconds)

Method	Fin-News (Latency)	PubMed (Latency)	Scalability (vs Data Size)
Flat-RAG	450 ms	620 ms	Linear Growth (Poor)
GraphRAG	1200 ms	1800 ms	Linear Growth (Poor)
Text Cube	40 ms	55 ms	Constant (Excellent)
VectorCube	**65 ms**	**78 ms**	**Constant (Excellent)**

VectorCube is orders of magnitude faster than RAG approaches because it pre-computes semantic aggregates. It is slightly slower than a traditional Text Cube due to the overhead of handling vector distributions, but this is negligible for the gain in semantic capability.

6.3 Results: Semantic Precision

We evaluated the quality of the answers using GPT-4 as a judge to score the relevance and completeness of the retrieved insights (0-10 scale).

Table 3. Answer Quality Scores (0-10)

Method	Drill-Down Precision	Roll-Up Coherence	Handling Synonyms
Text Cube	4.2	5.0	3.0 (Fails on synonyms)
Flat-RAG	7.5	4.0 (Loses global context)	8.0
VectorCube	**9.1**	**9.4**	**9.2**

Text Cubes fail when keywords don't match exactly (e.g., "tumor" vs "cancer"). Flat-RAG finds relevant snippets but fails to provide a coherent "Roll-Up" summary of the whole. VectorCube excels by combining the hierarchy of the Cube with the synonym awareness of Vectors.

6.4 Sensitivity Analysis

To address the need for evaluation under a variety of conditions, we tested VectorCube's semantic precision while varying the **Cluster Count (k)** used in the Distributional Aggregation (Algorithm).

- **Low k (k=1):** Equivalent to simple averaging. Precision dropped to 6.8, as distinct concepts were merged (e.g., "positive" and "negative" sentiment canceling out).
- **High k (k=5):** Precision peaked at 9.1.
- **Excessive** k (k>10): Latency increased by 40% with diminishing returns on precision. This confirms that modeling the distribution (via multiple centroids) is crucial for Neuro-Symbolic OLAP.

7. Discussion and Limitations

While Distributional Aggregation is powerful, vector spaces are not perfectly additive. Adding the

vector for "King" and "Woman" to get "Queen" is an idealization. In complex domains, aggregation can introduce noise. We mitigate this by storing multiple centroids per cell, but this increases storage size. Furthermore, the Semantic Dimensions are generated by LLMs. Occasionally, the LLM may hallucinate a category or classify a document incorrectly during the induction phase. We currently employ a "Human-in-the-Loop" verification step for the top-level dimensions to ensure stability.

8. Conclusion and Future Work

VectorCube represents the convergence of two powerful streams of data management: the structural discipline of OLAP and the semantic depth of Neuro-Symbolic AI. By treating vector embeddings as first-class citizens within a multidimensional data cube, we enable a new class of "Semantic BI" applications. Analysts can now drill down into the *meaning* of data, not just its labels, asking "Why?" with the same speed they used to ask "How much?". Future work will focus on **Dynamic Dimension Evolution**, allowing the cube to automatically restructure itself as new concepts emerge in the data stream, fully realizing the vision of an autonomous, self-organizing decision support system.

REFERENCES

[1] Lin, X., & Han, J., (2008) “Text Cube: Computing IR measures for multidimensional text database analysis”, Proceedings of the ICDM, pp905-910.
[2] Wang, J., & Li, Y., (2021) “Milvus: A Purpose-Built Vector Data Management System”, Proceedings of SIGMOD, pp2614-2627.
[3] Vaswani, A., & Shazeer, N., (2017) “Attention is All You Need”, Advances in Neural Information Processing Systems, Vol. 30, pp5998-6008.
[4] Chaudhuri, S., & Dayal, U., (1997) “An overview of data warehousing and OLAP technology”, SIGMOD Record, Vol. 26, No. 1, pp65-74.
[5] Mikolov, T., & Sutskever, I., (2013) “Distributed Representations of Words and Phrases and their Compositionality”, Advances in NIPS, pp3111-3119.
[6] Microsoft GraphRAG, "From LLMs to Knowledge Graphs," 2024.
[7] P. Kargupta, J. Han et al., "TaxoAdapt: Aligning LLM-Based Multidimensional Taxonomy Construction to Evolving Research Corpora," Preprint, 2025.

AUTHOR

Siddhesh Ramesh Surve (Member, IEEE) Siddhesh Surve received the B.E. degree in electronics and telecommunication engineering from Mumbai University, Mumbai, India, in 2012, and the M.S. degree in computer engineering from Rutgers University, New Jersey, USA, in 2014.

He is currently an Engineering Manager at Meta, Seattle, WA, USA, overseeing infrastructure that processes petabytes of data for optimizing ranking models. Prior to this, he served as an Engineering Lead at TikTok, driving data platform initiatives for global e-commerce and disaster recovery, and as a Tech Lead at Microsoft, delivering real-time customer insights for Dynamics 365. His research interests include distributed systems, large-scale data infrastructure, and the optimization of compound AI systems.

©2026 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.