Semantic Condensation of High-Cardinality Time Series for LLM-Driven Observability

Akila Balasubramanian

Apr 2026



The Challenge

LLMs and Raw Telemetry: A Poor Fit

🔢 Token Overload

Thousands of time series passed directly to LLMs exhaust token budgets before meaningful analysis can begin.

✂️ Signal Loss

Truncation strips away precisely the anomalies and change points that matter most to on-call engineers.

⚠️ Inconsistent Outputs

Without structured context, LLM responses become unpredictable undermining troubleshooting reliability at scale.

What Actually Breaks in Production

Input (Raw telemetry)

Time Series (last 10 min):

service-A latency (ms): 120, 118, 122, 319, 421, 202, 135
service-B latency (ms): 95, 97, 102, 480, 920, 1200, 1480
service-C latency (ms): 80, 82, 81, 79, 80, 100
pod-D restarts: 0, 0, 1, 2, 1, 0, 0
error-rate service-B (%): 0.2, 0.3, 0.2, 2.5, 4.8, 5.1, 4.6

Total: 2,000+ time series (truncated)

LLM Output (What we got)

Root Cause: Unclear

Reasoning:
- Multiple services show minor variations
- No dominant anomaly detected

Confidence: Low

Observations:

- 👎 Misses the actual spike with service-B
- 👎 Misses the restart signal with pod-D
- 👎 Downplays anomaly
- 👎 Spreads attention across signals

The Scale Problem

Modern observability platforms generate telemetry across thousands of entities at once hosts, services, containers, endpoints. At that scale, direct LLM ingestion is not just inefficient; it is fundamentally incompatible with reliable context processing, as signal is buried under volume, cardinality, and churn.

Volume

Data throughput quickly exceeds what can fit in a useful prompt.

Cardinality

Thousands of entities and dimensions explode the context space.

Velocity

Rapid change makes any static snapshot stale almost immediately.



Raw Data is Not LLM-Ready

Stop *Fitting* Data *Into* LLMs

Start *Shaping* Data *For* LLMs

Shift in Approach

Semantic Condensation

Semantic Condensation is a **token-aware transformation layer** that **turns raw high-cardinality time series** into **structured summaries** built for LLM context windows. It addresses the mismatch between raw observability volume and model capacity, preserving diagnostic meaning where downsampling and naive aggregation fail.

Token-aware

Optimized for limited context, not just storage.

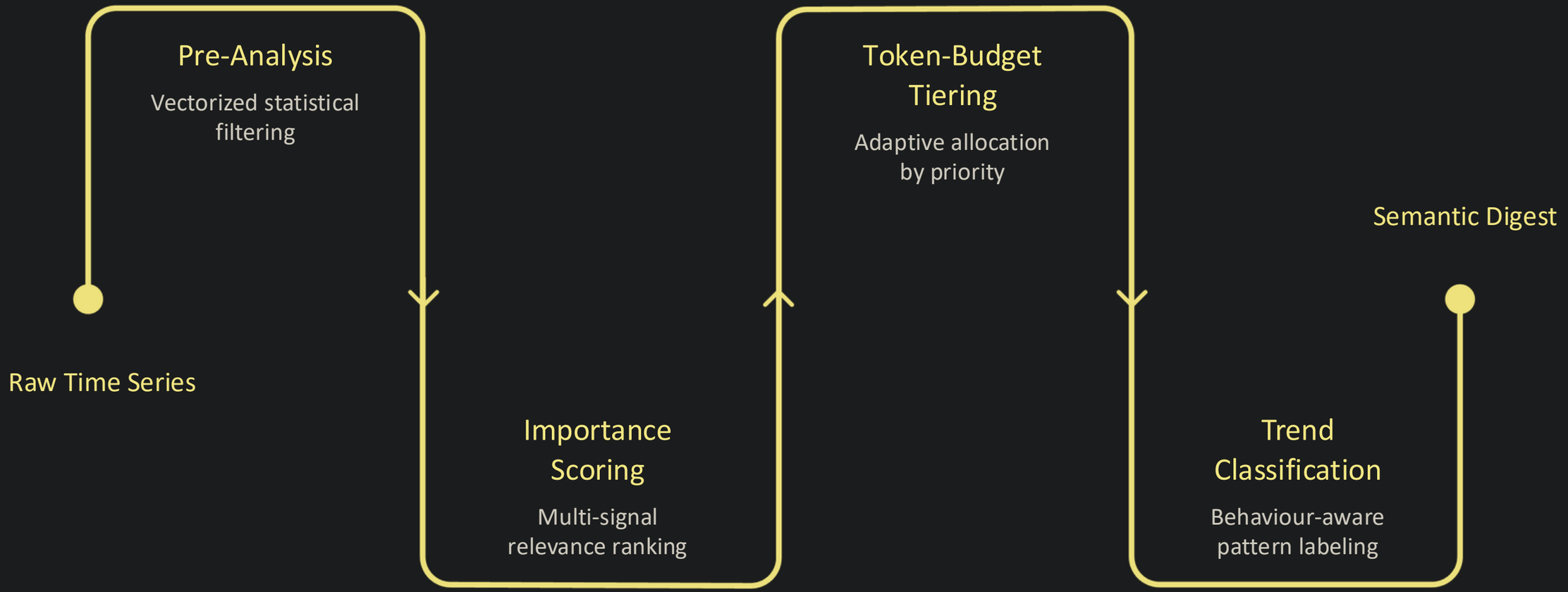
Semantic coherence

Preserves relationships and event meaning.

Operational fidelity

Retains the detail needed for diagnosis and action.

How We Make Data LLM-Ready



Each stage **strips noise** while **preserving operational semantics,** ensuring that what reaches the LLM is **dense with signal**, not diluted by volume.

Key Idea #1: Remove computation from LLM

Vectorised Statistical Pre-Analysis

LLMs struggle with raw statistical signals—these are better handled before inference.

Baseline characterisation

Mean, variance, and autocorrelation extracted per series.

Anomaly pre-flagging

Outlier windows identified before scoring, reducing downstream computation.

Latency-safe

Designed to run within strict production latency budgets.

Key Idea #2: Guide attention

Multi-Signal Importance Scoring

LLMs don’t know what to focus on—we make that decision.

What gets scored?

- Anomaly severity and rarity
- Rate-of-change and volatility
- Cross-signal correlation strength
- Historical incident co-occurrence
- Business-criticality of the emitting entity

Scoring is computed across all signals jointly, not independently ensuring that correlated anomalies across services are surfaced together.



Key Idea #3: Fits context window

Adaptive Token-Budget Tiering

LLMs are constrained by context windows—so we allocate tokens deliberately.

Not all time series deserve equal representation. Token budgets are allocated dynamically based on importance scores, operational context, and available context window.

1

Tier 1 - Detailed

Highest-importance series receive complete statistical summaries, trend annotations, and anomaly narratives.

2

Tier 2 - Compressed

Mid-ranked series are represented as compact structural descriptions with key inflection points preserved.

3

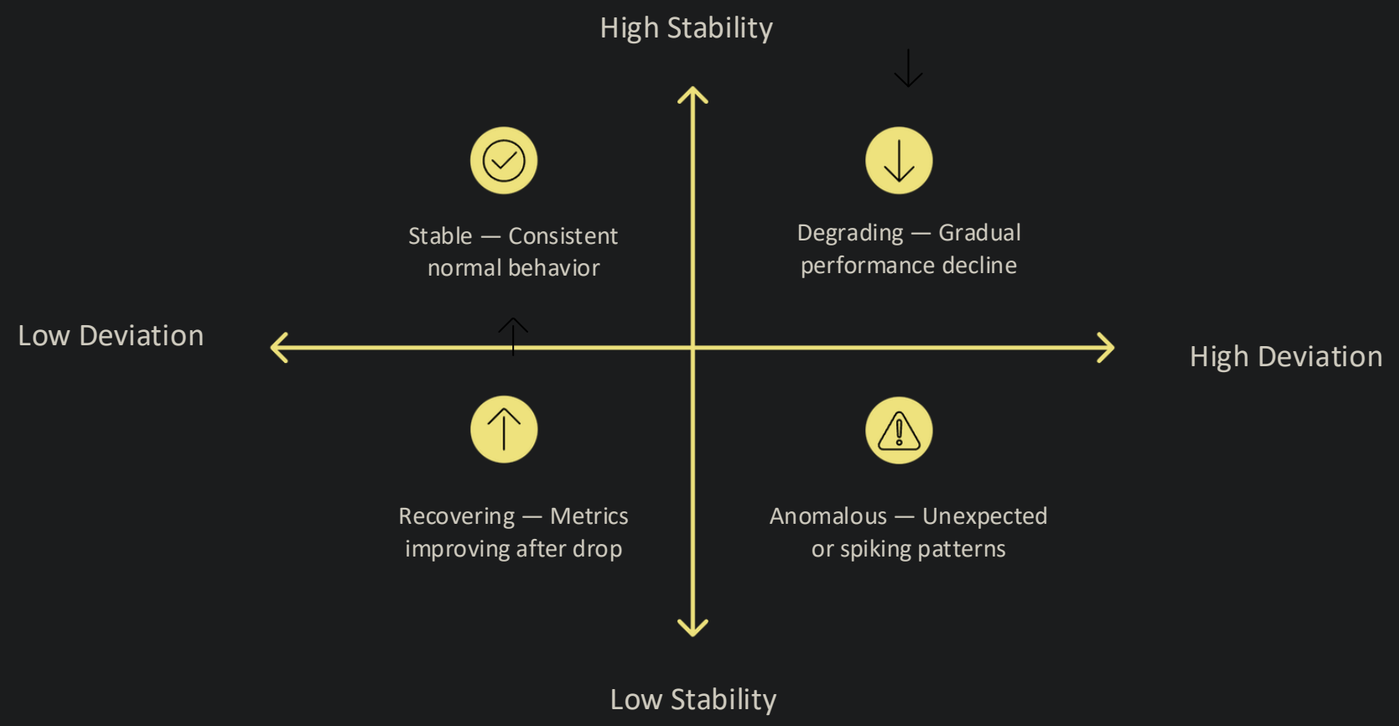
Tier 3 - Minimal

Low-importance series are reduced to a single-line status retained for completeness but consuming near-zero budget.

Key Idea #4: Enable reasoning

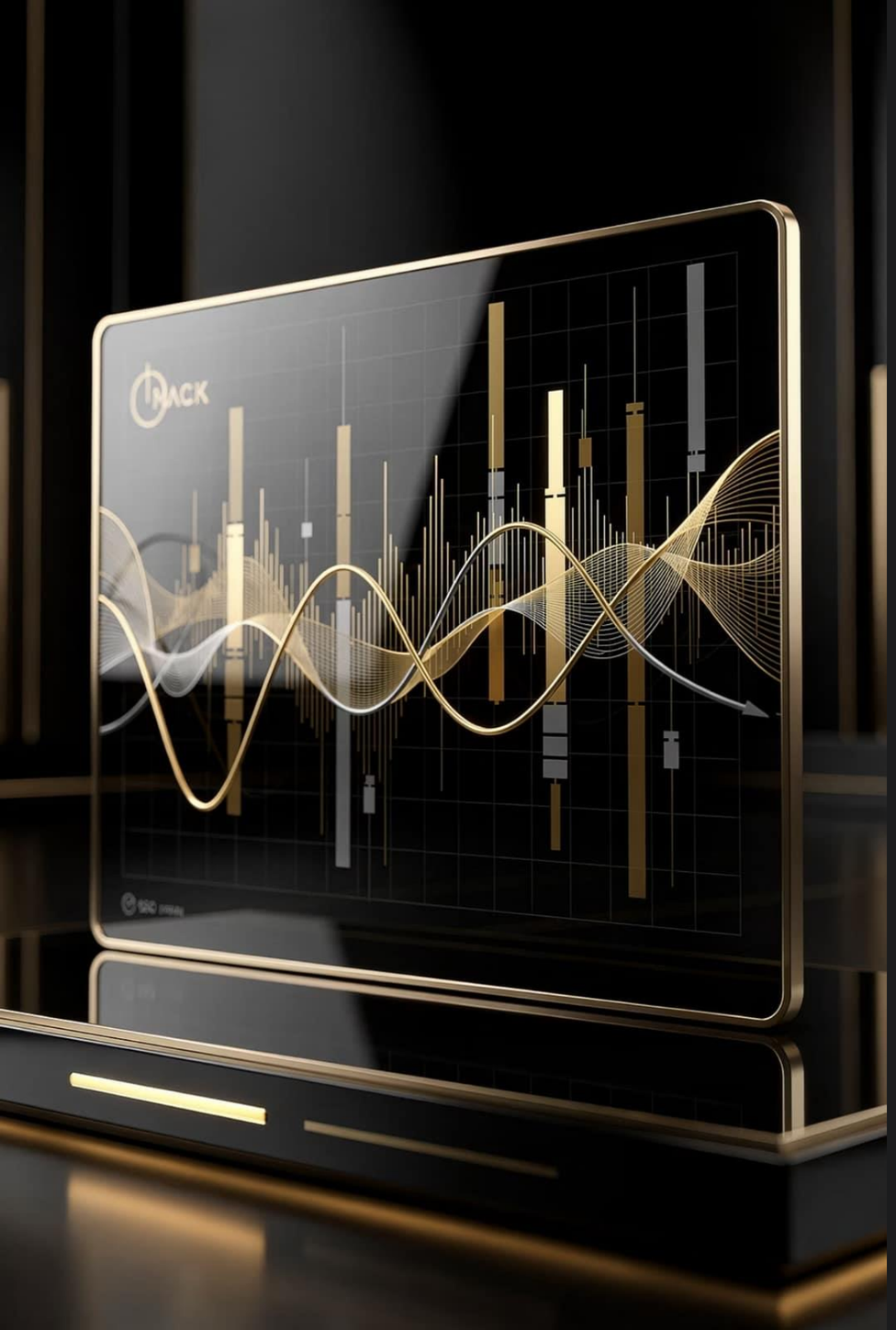
Behaviour-Aware Trend Classification

LLMs reason in language, not numbers—so we translate signals into semantic labels.



Rather than encoding raw numeric sequences, each time series window is assigned a **semantic behaviour class** a human- and LLM-readable label that encodes operational meaning.

This classification step is the bridge between numerical telemetry and natural-language reasoning, enabling LLMs to draw on linguistic inference patterns rather than numerical interpolation.



What Gets Preserved

What the LLM sees is smaller—but smarter.

Anomalies

Spike, dip, and distributional shifts are always retained → anomalies remain visible to the LLM

Change Points

Structural breaks are explicitly annotated → LLMs can detect transitions

Operational Patterns

Patterns like recovery and periodicity are preserved → LLMs can perform coherent reasoning over time

Before vs After

Same model. Same prompt. Only the input changed.

Input – Raw time series (~120K tokens)

Time Series (last 10 min):

service-A latency (ms): 120, 118, 122, 319, 421, 202, 135
service-B latency (ms): 95, 97, 102, 480, 920, 1200, 1480
service-C latency (ms): 80, 82, 81, 79, 80, 100
pod-D restarts: 0, 0, 1, 2, 1, 0, 0
error-rate service-B (%): 0.2, 0.3, 0.2, 2.5, 4.8, 5.1, 4.6

Total: 2,000+ time series (truncated)

LLM Output

Root Cause: Unclear

Reasoning:
- Multiple entities show minor variations
- No dominant anomaly detected

Confidence: Low

Input – Semantic Digest (~4.5K tokens)

service-B:
- latency spike (100ms → 1480ms)
- error rate increase (0.2% → 5.1%)
- correlated pod restarts (pod-D)

service-A:
- transient increase, no sustained anomaly

service-C:
- stable baseline behavior

Global Observations:
- primary anomaly in service-B
- correlated with pod-D restarts
- signals align across latency, error rates, and restarts
- no system-wide degradation

LLM Output

Root Cause: service-B degradation

Reasoning:
- Significant latency spike (>10x increase)
- Concurrent error rate increase
- pod-D restarts indicate downstream impact from service-B instability
- Other services remain stable

Confidence: High

Why Not Just Apply Traditional Approaches

Traditional Approaches

Downsampling

Reduces volume but discards short-lived spikes the events most likely to indicate failures.

Aggregation

Averages obscure variance; rolled-up metrics lose the temporal structure LLMs need for reasoning.

Raw Truncation

Context windows fill with low-signal data, pushing critical events outside the LLM's view.

Semantic Condensation

- Designed for LLM reasoning, not dashboards
- Preserves anomalies, transitions, and patterns
- Maintains a coherent operational narrative.

The goal isn’t just to compress — it’s to preserve meaning.

Performance at Scale

- **Time Series**

  Handles thousands of concurrent time series within strict production latency budgets.

- **Token Reduction**

  Reduces input size by orders of magnitude while preserving critical signals required for reasoning.

- **Accuracy Gain**

  Improves troubleshooting accuracy by up to 25 percentage points in real production scenarios.

Designing LLM-Ready Telemetry Pipelines

Key architectural principles for production-grade implementation:

01

Pre-compute at ingest

Run statistical pre-analysis as telemetry arrives never at query time. Latency budgets depend on it.

02

Score contextually

Importance scores should incorporate live incident context, not just signal statistics in isolation.

03

Tier dynamically

Token budgets should adapt to the active context window and the LLM model in use.

04

Validate coherence

Test summaries for internal contradiction before LLM dispatch incoherent inputs generate unreliable outputs.

A New Abstraction Layer for Observability

Semantic Condensation is not a compression trick — it is a **purpose-built semantic interface** between telemetry infrastructure and AI reasoning systems.

Semantically Consistent

Summaries are internally coherent, enabling reliable LLM chain-of-thought reasoning.

Token-Efficient

Orders-of-magnitude reduction in representation cost without sacrificing critical signals.

Production-Ready

Designed for real latency constraints, not just research benchmarks.



Key Takeaways

What to Take Back to Your Team

- **Raw telemetry is not LLM-ready**

  High-cardinality time series require a dedicated transformation layer before reaching any language model.

- **Semantic coherence beats compression**

  The goal is not just fewer tokens it is internally consistent, operationally meaningful summaries.

- **Importance scoring is the key lever**

  Multi-signal, contextual scoring determines what gets preserved and therefore what the LLM can reason about reliably.

- **This is a new interface layer**

  Semantic Condensation defines a repeatable abstraction between observability infrastructure and AI-driven troubleshooting.

Thank You!