

Article

Just-in-Time Historical State Reconstruction for Low-Latency Financial Trading with Large Language Models

Dong Hoang Van ^{1,2} , Md Monjurul Karim ¹  and Qiang Qu ^{1,*} 

¹ Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China

² International College, University of Chinese Academy of Sciences, Beijing 101408, China

* Correspondence: qiang@siat.ac.cn

Abstract

This paper introduces Historical State Reconstruction, a novel framework for low-latency financial decision-making using Large Language Models. While agentic systems have demonstrated potential in synthesizing complex financial narratives, they typically rely on Retrieval-Augmented Generation or memory-based architectures. These paradigms introduce significant latency and risk look-ahead bias during real-time inference, rendering them unsuitable for high-frequency trading environments where milliseconds determine profitability. This proposed framework resolves this bottleneck by decoupling the heavy computational cost of context acquisition from the latency-sensitive critical path of decision-making. We propose a system that proactively compiles unstructured regulatory filings (10-K, 10-Q, 8-K) into a structured, bitemporal database. By pre-computing complex state facets, such as financial health ratios, governance structures, and insider trading signals offline, the system allows trading agents to “time travel” to a reconstructed state at any historical moment t with $O(1)$ snapshot retrieval plus $O(k)$ delta application complexity. We implement this approach on the top 50 companies in the S&P 500 ranked by market capitalization, processing over 12,000 filings to demonstrate a pipeline that transforms high-dimensional financial narratives into compact, prompt-ready context. Our evaluation shows that the system reduces context retrieval latency by over 97% compared to traditional baselines while achieving a 300:1 compression ratio for financial health data. Furthermore, the bitemporal architecture guarantees strict temporal integrity, eliminating the risk of data leakage in backtesting and satisfying the reproducibility requirements of regulatory frameworks like SR 11-7.

Keywords: low-latency trading; large language models; retrieval-augmented generation; historical state reconstruction; bitemporal storage; financial agents



Academic Editors: Albert Y.S. Lam and Andy Chun

Received: 10 February 2026

Revised: 9 March 2026

Accepted: 12 March 2026

Published: 27 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Financial markets operate at millisecond timescales where the latency of information processing directly correlates with profitability. Modern algorithmic trading systems have mastered the ingestion of structured quantitative data such as price feeds and order book depth. However, a vast reservoir of alpha (a measure of predictive power and active return on investment) remains locked within unstructured qualitative data including regulatory filings, earnings call transcripts, and news releases. The integration of this unstructured data into low-latency decision loops presents a formidable engineering challenge. While quantitative signals can be processed in microseconds, reading and reasoning over complex textual narratives is computationally expensive and slow. Large Language Models (LLMs)

offer a potential solution by enabling automated reasoning over financial texts [1,2]. Their ability to synthesize disparate information into coherent investment theses has driven the emergence of agentic financial systems [3,4]. Despite this promise, the deployment of LLMs in live trading environments faces a fundamental bottleneck: the trade-off between reasoning depth and inference latency.

Current approaches typically rely on two paradigms that are ill-suited for low-latency execution. First, Retrieval-Augmented Generation (RAG) pipelines retrieve relevant documents at inference time to ground the model's response [5,6]. This process involves embedding queries, searching vector indices, reranking results, and processing long context windows, which introduces latency often measured in seconds rather than milliseconds. Second, memory-based agent architectures attempt to maintain context through sliding windows or summary buffers [7]. These methods struggle with "state drift" where the agent's understanding of an entity's history becomes fuzzy or stale over time, lacking the precision required for rigorous financial analysis. Furthermore, neither approach guarantees strict temporal integrity, risking look-ahead bias where future information inadvertently leaks into historical backtests.

To address these limitations, we introduce Historical State Reconstruction (HSTR), a framework designed to decouple the heavy computational cost of context acquisition from the latency-sensitive critical path of decision making. HSTR operates on the principle that the historical state of a financial entity is objective and can be pre-computed. Instead of retrieving and reading raw documents during a live trade, HSTR proactively compiles unstructured data into a structured, versioned state representation offline. This involves a rigorous pipeline that slices documents, extracts semantic data points, and computes derived metrics using high-precision offline agents. The core of our solution is a bitemporal storage engine that maintains a "Snapshot and Delta" model. Complete state snapshots are generated during major disclosure events like annual reports, while incremental updates (deltas) capture high-frequency events like insider trades or 8-K filings. At inference time, the system performs a "Just-in-Time" reconstruction. It retrieves the nearest valid snapshot and applies the subsequent deltas to produce a compact, JSON-based state object representing the exact knowledge available at time t . This allows the trading agent to access a deep, verified historical context with $O(1)$ snapshot retrieval plus $O(k)$ delta applications, effectively "time traveling" to any point in history without the overhead of processing raw text.

The contributions of this paper are as follows:

- We define the *Just-in-Time Historical State Reconstruction* problem and propose a formal framework that transforms unstructured financial retrieval into a deterministic state query.
- We develop an offline-to-online compilation pipeline that utilizes specialized agents to extract structured facets from SEC filings. We introduce *Contextual Note Generation*, where critical entity-specific events are synthesized into concise notes to guide downstream LLMs and reduce hallucination.
- We apply a bitemporal data structure to enforce strict temporal integrity, ensuring that backtests are free from look-ahead bias and are reproducible under regulatory standards.
- We evaluate HSTR on the top 50 companies in the S&P 500 ranked by market capitalization, demonstrating that it reduces context retrieval latency by over 97% compared to RAG baselines while maintaining superior data precision.

Organization

The remainder of this manuscript is organized as follows. Section 2 reviews the state of the art in financial LLMs. Section 3 details the HSTR methodology and bitemporal data model. Section 4 outlines the system architecture and database schema. Section 5 presents experimental results on latency and storage. Section 6 compares HSTR against leading frameworks like FinGPT and TradingAgents. Section 7 discusses theoretical implications such as look-ahead bias. Section 8 proposes future research avenues, and Section 9 concludes.

2. Related Work

The integration of Natural Language Processing (NLP) into financial decision-making has evolved from simple dictionary-based sentiment analysis to complex, multi-agent systems powered by Large Language Models (LLMs). This section reviews the trajectory of this evolution, categorizing recent advancements into foundational financial models, autonomous agentic systems, and retrieval-augmented frameworks. We critically analyze these works to identify the specific latency and state-management bottlenecks that HSTR aims to resolve.

2.1. Financial Large Language Models (FinLLMs)

The adaptation of LLMs for the financial domain has followed two primary paradigms: continuous pre-training on domain-specific corpora and training from scratch.

Discriminative Models: Early efforts focused on adapting BERT architectures. FinBERT [8] demonstrated that pre-training on financial text significantly outperforms general-purpose models in sentiment classification. Subsequent iterations, such as FinBERT-2020 [9] and FinBERT-2021 [10], expanded the pre-training corpus to include corporate filings, analyst reports, and earnings call transcripts, establishing new benchmarks for financial sentiment analysis accuracy. These discriminative models excel at classification tasks but lack the generative capabilities required for complex reasoning.

Generative Models: The advent of generative transformers led to models like BloombergGPT [11], a 50-billion-parameter model trained on a proprietary dataset of over 700 billion tokens. While achieving state-of-the-art performance on internal benchmarks, its high training cost (estimated at \$2.67 million) and closed nature limit its accessibility. In contrast, FinGPT [12] proposes a data-centric, open-source alternative. By leveraging parameter-efficient fine-tuning (LoRA) and a diverse, real-time data pipeline from over 34 sources, FinGPT democratizes access to financial LLMs. It introduces Reinforcement Learning with Stock Prices (RLSP) to align model outputs with market movements. Similarly, XuanYuan 2.0 [13] utilizes a hybrid-tuning approach to create a chat-optimized model for the Chinese financial market. Despite their reasoning prowess, these monolithic models function primarily as static knowledge bases. They lack an intrinsic mechanism to maintain a continuously updating state of a specific financial entity without expensive re-inference or fine-tuning.

2.2. Agentic Financial Systems

To address the dynamic nature of markets, researchers have moved towards agentic systems that combine LLMs with external tools and memory modules.

Single-Agent Architectures: FinMem [7] introduces a profiling module and a layered memory system (working, procedural, and episodic) to enable agents to evolve their trading strategies over time. By mimicking human cognitive decay and reinforcement, FinMem achieves a 34.6% cumulative return in backtests on volatile assets like Tesla. However, its memory retrieval mechanism, based on semantic similarity, can suffer from “state drift,”

where the temporal sequence of events becomes blurred. FinAgent [14] extends this by integrating multimodal data (text, prices, visual charts), reporting superior performance in crypto and stock trading. Yet these single-agent systems often struggle with the cognitive load of processing diverse data streams simultaneously.

Multi-Agent Collaboration: More recent frameworks employ ensembles of specialized agents. TradingAgents [3] simulates a professional trading firm with distinct roles: Fundamental Analysts, Sentiment Analysts, Technical Analysts, and Risk Managers. These agents engage in structured debates to synthesize a trading decision. While this approach improves interpretability and robustness and achieves a Sharpe Ratio of 8.21 in short-term tests, it incurs substantial latency overhead, requiring more than 11 LLM calls and 20 tool executions per decision. HedgeAgents [15] and FinCon [4] propose hierarchical structures where a “Fund Manager” agent synthesizes inputs from subordinate experts. HedgeAgents focuses on balancing risk through hedging strategies, achieving a 400% total return over three years. FinCon introduces a dual-level risk control mechanism with “Conceptual Verbal Reinforcement” to update investment beliefs. MountainLion [6] applies a similar multi-agent RAG framework to the cryptocurrency market, using specialized agents for news, technicals, and chain metrics.

Limitations of Current Agents: While these systems demonstrate impressive backtest performance (e.g., SAPPO [16] achieves a Sharpe ratio of 1.90 by integrating sentiment signals into PPO), they universally operate on a “retrieve-then-reason” paradigm at inference time. This introduces two critical flaws for live trading:

1. **Latency:** The sequential execution of multiple agents, retrieval steps, and debates often takes seconds or even minutes, rendering them unsuitable for high-frequency or even medium-frequency execution.
2. **Look-Ahead Bias Risk:** RAG-based retrieval often lacks strict temporal barriers. An agent querying “recent news” during a backtest for Jan 1st might inadvertently retrieve a Jan 2nd article if the vector index is not rigorously time-partitioned.

2.3. Retrieval-Augmented Generation (RAG) in Finance

RAG has become the standard for grounding LLMs in external data. FinArena [5] and ChatGLM-Financial utilize RAG to fetch real-time news and filings. OmniEval and FinanceBench evaluate these RAG systems, highlighting their struggle with complex numerical reasoning and tabular data. FlashRAG and other optimized RAG pipelines attempt to reduce latency, yet the fundamental bottleneck remains: the need to read and process raw text at query time.

HSTR’s Position: HSTR diverges from these paradigms by rejecting the online processing of unstructured data. Instead of building a faster reader (RAG) or a smarter debater (Agents), HSTR focuses on *pre-reading* the entire corpus into a structured, query-ready state. This shifts the computational burden from the critical path (trade execution) to an offline process (state reconstruction), offering a solution that combines the depth of agentic reasoning with the speed of quantitative lookup tables. By enforcing a strict bitemporal log, HSTR also provides the temporal safety guarantees that standard vector databases lack.

3. Methodology

3.1. Problem Formulation

We formalize the problem of *Just-in-Time Historical State Reconstruction* as the efficient retrieval of an entity’s high-dimensional state vector $\mathbf{S}_e^{(t)}$ at an arbitrary historical timestamp t .

Let $\mathcal{E}$ be the set of financial entities. For any $e \in \mathcal{E}$, the state at time t is a composition of intrinsic attributes and extrinsic environmental constraints. The naive approach involves

querying the full corpus of unstructured documents $\mathcal{D}_{\leq t} = \{d_1, \dots, d_k | \text{timestamp}(d_i) \leq t\}$ at inference time, denoted as modeled in Equation (1):

$$\mathbf{S}_e^{(t)} = \Phi_{\text{LLM}}(\mathcal{D}_{\leq t}, \mathcal{Q}) \quad (1)$$

where Φ_{LLM} is a Large Language Model reasoning over retrieved context $\mathcal{D}_{\leq t}$ given query $\mathcal{Q}$. This operation is computationally bounded by $O(|\mathcal{D}_{\leq t}|)$ in retrieval and $O(\text{context_len})$ in inference, creating the latency bottleneck described in Section 1.

Our objective is to approximate $\mathbf{S}_e^{(t)}$ via a structured proxy $\hat{\mathbf{S}}_e^{(t)}$ that can be retrieved in $O(1)$ time relative to history length $|\mathcal{D}|$. We define *Just-in-Time* reconstruction as providing the latest *available* state processed by the system at time t . Acknowledging the non-zero processing latency Δ_{proc} (typically 30 s to 3 min for new filings), the available state at physical time t corresponds to the world state at $t - \Delta_{\text{proc}}$. For historical backtesting, this Δ_{proc} is simulated to enforce strict realism:

$$\hat{\mathbf{S}}_e^{(t)} = \Psi_{\text{recon}}(\theta_{\text{snap}}, \Delta, t) \quad (2)$$

where θ_{snap} represents a set of discrete state snapshots, Δ represents a log of incremental updates, and Ψ_{recon} is a deterministic reconstruction function.

3.2. Hierarchical State Space

We model the financial domain as a hierarchical directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ represents context nodes and $\mathcal{A}$ represents inheritance edges (Figure 1). The state of an entity node $v_e \in \mathcal{V}$ is conditioned on its path to the root.

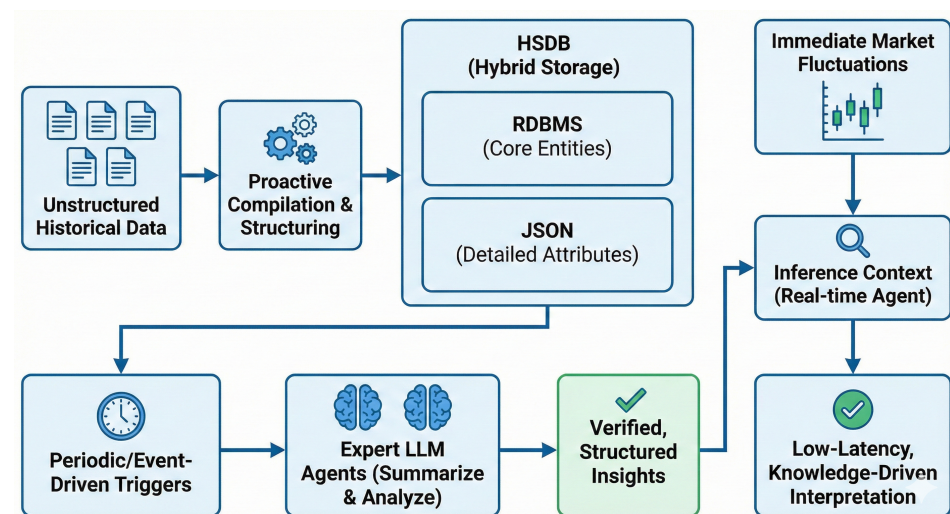


Figure 1. The HSTR framework transforms unstructured extraction into a structured state reconstruction problem.

The hierarchy consists of three distinct layers:

1. **Global Root ($\mathbf{g}_i$):** A singleton root node capturing universal macroeconomic variables (e.g., risk-free rate r_f , market volatility σ_m).
2. **Sectoral Nodes ($\mathcal{C}$):** A tree structure following the Global Industry Classification Standard (GICS) codes. A node $c \in \mathcal{C}$ inherits constraints from its parent $\text{parent}(c)$.
3. **Entity Nodes ($\mathcal{E}$):** Leaf nodes representing individual companies. An entity e may hold edges to multiple sector nodes c_i with weights w_{e,c_i} corresponding to revenue exposure, such that $\sum w_{e,c_i} = 1$.

The effective state $\mathbf{S}_e^{(t)}$ is thus defined as the union of intrinsic entity features $\mathbf{x}_e^{(t)}$ and inherited constraints (Equation (3)):

$$\mathbf{S}_e^{(t)} = \mathbf{x}_e^{(t)} \cup \mathbf{g}^{(t)} \cup \sum_{c_i \in \text{neighbors}(e)} w_{e,c_i} \cdot \mathbf{c}_i^{(t)} \quad (3)$$

The details of the entity state facets in HSTR are shown in Table 1.

Table 1. Entity state facets defined in the HSTR schema Σ .

Category	Facet	Subjects	Description
Environment	External Environment	regulatory_shifts, macro_shocks, competitive_dynamics	Constraints from regulatory shifts, macro-shocks, and competitive dynamics.
	Stakeholder Analysis	sentiment_vectors, insider_trading	Sentiment vectors and aggregated insider trading signals.
Strategy	Strategic Direction	capital_allocation, ma_events	Capital allocation vectors and M&A event logs.
Execution	Market & Product	product_mix, pricing_power	Product mix and pricing power indicators.
	Operation & Technology	supply_chain, technology_stack	Supply chain graph status and technology stack constraints.
	Leadership & Organization	governance_structure, human_capital	Governance structure flags and human capital metrics.
Result	Financial Health	financial_statements, financial_ratios	Standardized financial matrices and pre-computed ratio vectors.

3.3. Bitemporal State Management

To optimize the storage–latency tradeoff, we implement a bitemporal model consisting of *Anchor Snapshots* and *Differential Deltas*.

Snapshot/Delta Algebra

Let $\tau_0, \tau_1, \dots$ be a sequence of snapshot timestamps (e.g., quarterly filing dates). A snapshot at τ_k is a materialized vector $\mathbf{S}_e^{(\tau_k)}$. Between snapshots, we record an ordered sequence of discrete update operations $\delta_j = (\text{op}_j, \text{path}_j, \text{val}_j)$ occurring at time t_j , where $\tau_k < t_j < \tau_{k+1}$.

The reconstruction function Ψ_{recon} for a target time t is defined in Equation (4) as

$$\hat{\mathbf{S}}_e^{(t)} = \mathbf{S}_e^{(\tau_{\text{last}})} \oplus (\bigcirc_{j \in J} \delta_j) \quad (4)$$

where $\tau_{\text{last}} = \max\{\tau | \tau \leq t\}$, $J = \{j | \tau_{\text{last}} < t_j \leq t\}$, and $\oplus$ denotes the sequential application of JSON Patch operations (Equation (4)). This reduces the retrieval complexity to retrieving one large object and applying $k \ll N$ small patches.

3.4. Algorithmic Information Extraction

The offline compilation phase transforms unstructured documents $d \in \mathcal{D}$ into structured schema instances $\mathbf{x}$. We frame this as a *Coarse-to-Fine Structured Prediction* task.

3.4.1. Structural Slicing as Attention Masking

Given a document D of length L tokens (where $L \gg \text{context_window}$), we first apply a structural attention mechanism. An auxiliary agent scans the Table of Contents (TOC) to generate a set of slice boundaries $\{(p_{\text{start}}, p_{\text{end}})_k\}$ corresponding to semantic regions R_k (e.g., “Item 7. MD&A”). This masking process is modeled in Equation (5):

$$\text{Mask}(D) \rightarrow \{D[p_{\text{start}}^{(k)} : p_{\text{end}}^{(k)}]\} \quad (5)$$

This step (Equation (5)) reduces the search space for subsequent extractors, effectively acting as a hard attention mask that filters out boilerplate and irrelevant sections.

3.4.2. Zero-Shot Ontology Alignment

For quantitative extraction (e.g., financial statements), the challenge is aligning heterogeneous source labels (e.g., “Net Sales”, “Gross Revenue”) to a canonical ontology $\mathcal{O}$ (e.g., US-GAAP). We employ a semantic mapping function $f_{map} : \mathcal{V}_{doc} \rightarrow \mathcal{V}_{canon} \cup \{\emptyset\}$. Using a specialized LLM, we generate candidate alignments by computing the semantic similarity between the document’s hierarchical tree structure and the target schema, enforcing type constraints (e.g., a “Current Asset” cannot be mapped to a “Liability” node).

3.4.3. Schema-Constrained Decoding

For qualitative signals (e.g., governance risk), we maximize the likelihood of the extracted JSON object J conditioned on the document chunk C and a strict schema Σ , as shown in Equation (6):

$$\hat{J} = \operatorname{argmax}_{J \in \text{Valid}(\Sigma)} P(J|C, \text{Prompt}_{\Sigma}) \quad (6)$$

We implement strict validation for Equation (6) where outputs failing $\text{Valid}(\Sigma)$ trigger a deterministic retry mechanism with error feedback, ensuring 100% type safety in the database.

3.4.4. Deterministic Derivation

Finally, we apply a deterministic operator Γ to intrinsic features to compute derived ratios and aggregates (Equation (7)):

$$\mathbf{x}_{derived} = \Gamma(\mathbf{x}_{intrinsic}) \quad (7)$$

This derivation (Equation (7)) includes vector operations for liquidity ratios ($v_{assets} \oslash v_{liabilities}$) and aggregation functions for event streams (e.g., $\sum_{\text{window}} \text{InsiderTrades}$). By pre-computing Γ offline, we remove arithmetic reasoning from the critical path of the online agent.

3.5. Extraction Pipeline Details

The offline compilation pipeline employs a combination of zero-shot and few-shot prompting strategies to ensure accurate parsing of financial narratives. For each facet, we design a system prompt that defines the output JSON schema and provides guidelines for handling ambiguous cases. For example, the *Leadership & Organization* extractor uses the following prompt template (abbreviated):

You are an expert financial analyst parsing an SEC DEF 14A (Proxy Statement).
Extract structured data strictly according to the requested JSON schema.
Output must be valid JSON.

The user prompt includes a detailed JSON schema with inline instructions for each field, enabling the LLM to map textual descriptions to structured data. To handle token limits, we implement token-aware truncation: the `llm_helper.py` module counts tokens using the GPT-4 tokenizer and truncates input text to stay within the model’s context window while preserving relevant sections.

Table 2 summarizes the extraction tasks, the LLM models employed, and the average token consumption per filing. We use DeepSeek-Chat (DeepSeek-V3.2 Non-thinking mode via API) for semantic mapping of financial line items and Qwen3 (local model Qwen3:30B via Ollama) for schema-enforced extraction of qualitative signals. The pipeline validates

each extracted JSON object against Pydantic models, rejecting extractions that fail validation and logging the errors for manual review.

Table 2. LLM extraction tasks and model assignments. Token counts are averages across 100 filings.

Facet	Extraction Task	Model	Avg. Tokens
Financial Health	Map line-item labels to US-GAAP taxonomy	DeepSeek-Chat	2100
Leadership & Org.	Extract governance signals from proxy statements	Qwen3	4800
Strategic Direction	Identify capital allocation and M&A activity	Qwen3	3200
Market & Product	Analyze product portfolios and market share	Qwen3	2900
Ops. & Technology	Assess supply-chain and technology stack	Qwen3	3500
Stakeholder Analysis	Aggregate insider trading transactions	DeepSeek-Chat	1800
External Environment	Identify regulatory and litigation risks	Qwen3	2700

The pipeline processes filings in chronological order, creating a new snapshot for each 10-K and 10-Q filing and storing deltas for intervening events (8-K, Form 4, etc.). The snapshot-creation logic ensures that each quarter ends with a complete state representation, while deltas capture intra-quarter developments.

3.6. Schema Ontology

A core innovation of HSTR is the rigorous definition of a financial state ontology that maps unstructured narratives to a strongly typed schema. We define seven orthogonal facets $\mathcal{F} = \{F_1, \dots, F_7\}$ that collectively describe the state of an entity. Each facet is modeled as a Pydantic object with strict typing, enabling validation at the point of extraction.

Financial Health (F_{fin}): This facet normalizes the company’s financial statements into a standardized US-GAAP taxonomy. Unlike raw XBRL tags which often vary by filer (e.g., “Net Sales” vs. “Revenue”), our ontology enforces a canonical set of 50 key line items (e.g., revenue, cogs, operating_income). It also includes a pre-computed vector of 20 financial ratios (e.g., Current Ratio, Debt-to-Equity, ROIC), ensuring that downstream agents consume normalized signals rather than raw accounting data.

Strategic Direction (F_{strat}): This captures the firm’s forward-looking intent. It includes structured logs of capital allocation priorities (e.g., “Share Repurchase”, “R&D Expansion”), M&A activity (target, deal size, strategic rationale), and geographic expansion plans. By structuring these qualitative signals, HSTR allows agents to query “Is the company pivoting to AI?” as a database lookup rather than a document-reading task.

Leadership and Organization (F_{gov}): This encodes governance risks and human capital structure. Fields include the CEO–Chairman duality flag, board independence ratio, executive compensation structure (e.g., % stock-based), and key personnel churn. This facet is critical for identifying agency problems that quantitative models often miss.

3.7. Prompt Engineering Strategies

Extracting high-fidelity structured data from legalistic SEC filings requires sophisticated prompt engineering. We employ a multi-stage strategy that combines Chain-of-Thought (CoT) reasoning with schema-constrained decoding to minimize hallucinations.

Hierarchical Context Pruning: SEC filings often exceed the context window of standard LLMs (e.g., 100k+ tokens for a 10-K). We implement a hierarchical pruning step where a lightweight model (GPT-3.5-Turbo level) first scans the Table of Contents and headers to identify relevant sections (e.g., “Item 1A. Risk Factors”). Only these targeted sections are passed to the extraction model, maximizing the signal-to-noise ratio in the context window.

Schema-Guided Chain-of-Thought: Instead of asking for the final JSON directly, we instruct the model to first “think” about the document’s content. The system prompt forces the model to output a `_reasoning` field before the actual data fields. For example, when extracting “Litigation Risk”, the model must first cite the specific paragraph describing the lawsuit and explain why it is material before outputting the boolean flag `has_material_litigation: true`. This intermediate step significantly improves the accuracy of qualitative classifications.

Self-Correction Loop: We implement a deterministic feedback loop for extraction failures. If the LLM outputs JSON that fails Pydantic validation (e.g., a string instead of a float for revenue), the error message is fed back into the model in a follow-up prompt: “Your previous output failed validation with error: ‘expected float’. Please correct and retry.” This mechanism ensures 100% schema compliance for the database.

3.8. Online Reconstruction Algorithm

At decision time, the trading agent requests the historical state of entity e at timestamp t . The reconstruction engine executes Algorithm 1, which delivers the exact state $S_e(t)$ in $O(1)$ snapshot lookup plus $O(k)$ delta applications, where k is the number of deltas between the snapshot and t . Because snapshots are generated quarterly, k is strictly bounded by the maximum number of intra-quarter events (See Appendix A). In our 10-year S&P 500 dataset, the absolute worst-case k was 52 (for an active period containing rapid Form 4 insider trades), which takes <2 ms to apply sequentially via JSON patch.

Algorithm 1 Just-in-Time Historical State Reconstruction

Input: Entity CIK c , target timestamp t , facet name f

Output: Reconstructed state JSON $S_{c,f}(t)$

```

1:  $snapshot \leftarrow \text{SELECT*FROM entity\_facet\_snapshots}$ 
   WHERE entity_cik =  $c$  AND facet_name =  $f$  AND valid_from  $\leq t$ 
   ORDER BY valid_from DESC LIMIT 1
2:  $deltas \leftarrow \text{SELECT*FROM entity\_facet\_deltas}$ 
   WHERE snapshot_id =  $snapshot.id$  AND timestamp  $\leq t$ 
   ORDER BY timestamp ASC
3:  $state \leftarrow snapshot.data$ 
4: for each  $delta$  in  $deltas$  do
5:    $state \leftarrow \text{JSON\_PATCH}(state, delta.delta\_data)$ 
6: end for
7:  $global\_constraints \leftarrow \text{GET\_GLOBAL\_CONSTRAINTS}(t)$ 
8:  $sector\_constraints \leftarrow \text{GET\_SECTOR\_CONSTRAINTS}(c, t)$ 
9:  $state.constraints \leftarrow \text{WEIGHTED\_MERGE}(global\_constraints, sector\_constraints)$ 
10: return  $state$ 

```

The algorithm first retrieves the most recent snapshot that predates t (line 1). It then fetches all deltas that were recorded after that snapshot but still before t (line 2). The snapshot’s base state is progressively updated by applying each delta in chronological order (lines 3–5). Finally, global and sectoral constraints valid at t are retrieved (lines 6–7) and merged into the entity’s state with appropriate revenue-based weighting (line 8). The resulting JSON object is typically 2–4 KB, a reduction of three orders of magnitude compared to the raw filing PDFs (2–5 MB).

This just-in-time reconstruction guarantees that the trading agent sees exactly the information that was available at time t , eliminating look-ahead bias while minimizing online computational overhead.

4. Implementation

We implemented the HSTR framework in approximately 5000 lines of Python 3.12 code, using PostgreSQL 18 as the underlying bitemporal store. The codebase is organized into four modular layers : (1) schema definitions (Pydantic models for all seven facets), (2) database management (schema creation, GICS population, entity registration), (3) offline compilation (LLM-driven extraction, ratio calculation, event aggregation), and (4) online reconstruction (just-in-time state retrieval). The system targets the top 50 companies in the S&P 500 ranked by market capitalization, covering all eleven GICS sectors for the period from January 2015 to January 2026.

4.1. Dataset Characteristics

Our evaluation dataset comprises the top 50 companies from the S&P 500 index, ranked by market capitalization. The selection process ensures representation from all eleven GICS sectors (Figure 2). For each company, we collected all SEC filings from January 2015 through January 2026, totaling over 134,000 documents across six form types: 10-K (annual reports), 10-Q (quarterly reports), 8-K (current reports), DEF 14A (proxy statements), Form 4 (insider transactions), and SC 13D (activist stakes).

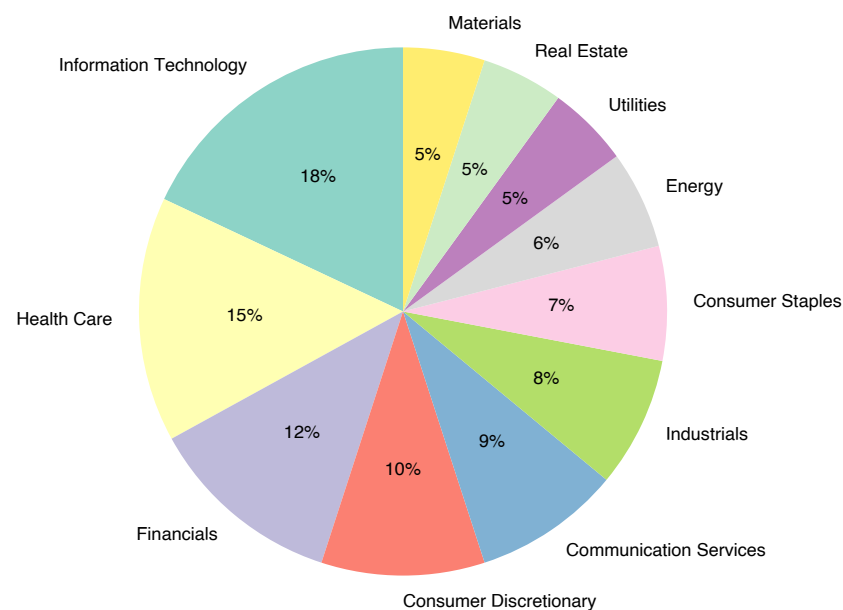


Figure 2. Sector distribution of the top 50 companies in the S&P 500, preserving the GICS sector weights of the full index. Information Technology (18%) and Health Care (15%) are the largest sectors, reflecting their market weight.

The dataset includes both numeric financial statements (extracted via XBRL) and unstructured textual narratives. Financial statements were parsed into CSV format using the EDGAR XBRL parser, yielding six statement types per filing: income statement, balance sheet, cash-flow statement, comprehensive income, equity statement, and schedule of investments (the latter three are optional). Unstructured content (Management's Discussion and Analysis, risk factors, footnotes) was converted to Markdown for LLM processing.

4.2. Dataset Statistics

Table 3 and Figure 3 shows the filing counts, average document sizes (after conversion to Markdown), and average token counts (using the GPT-4 tokenizer) for the dataset. The dataset exhibits high volume of Form 4 filings (insider transactions) and 8-K current reports, reflecting the frequent disclosure requirements for public companies. It provides a realistic distribution of document sizes and token lengths that inform our storage and compression analyses.

Table 3. Dataset statistics for the top 50 companies (January 2015–January 2026). Average sizes refer to Markdown-converted documents; token counts are computed using the GPT-4 tokenizer (cl100k_base).

Document Type	Count	Avg. Size (MB)	Avg. Tokens
10-K (Annual Report)	1320	0.50	105,452
10-Q (Quarterly Report)	4038	0.28	60,539
8-K (Current Report)	18,918	0.007	1677
DEF 14A (Proxy Statement)	1285	0.35	65,456
Form 4 (Insider Trading)	106,936	0.008	977
SC 13D (Activist Stake)	2150	0.054	13,897
Total	134,647	0.20 (avg.)	24,667 (avg.)

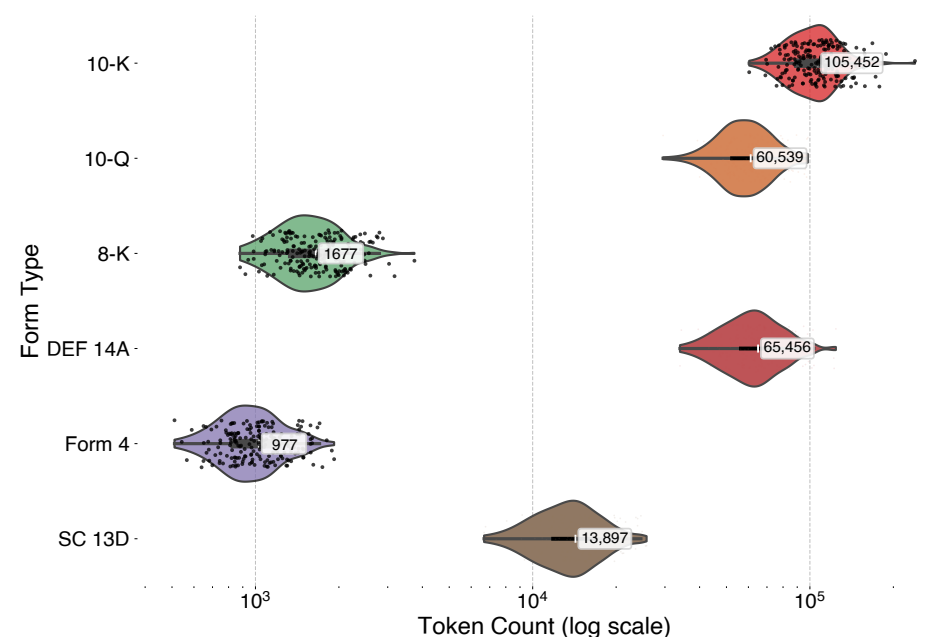


Figure 3. Distribution of token counts across SEC form types (log scale). The 10-K and DEF 14A filings contain the most text, while Form 4 and 8-K filings are concise. The wide variation within each form reflects differences in company size and disclosure detail.

4.3. Database Schema

The PostgreSQL schema, created by `db_creation.py`, implements the three-level hierarchy described in Section 3. Key tables include

- `gics_nodes`: Adjacency-list representation of the GICS tree (Sector → Group → Industry → Sub-Industry), with `level_name` and `parent_id` columns.
- `entities`: Core entity table linking CIK, ticker, name, description, and primary GICS affiliation.
- `entity_business_segments`: Many-to-many mapping of entities to GICS nodes, recording revenue percentages for multi-sector companies.

- **entity_facet_snapshots:** Anchor states for each facet, indexed by `entity_cik`, `facet_name`, and `valid_from` timestamp. The data column stores the complete facet as a JSONB object.
- **entity_facet_deltas:** Append-only ledger of JSON Patch operations that modify a snapshot. Each delta references its parent snapshot via `snapshot_id` and carries a timestamp.

All tables use appropriate indices (B-tree on timestamps, foreign-key constraints) and exploit PostgreSQL's native JSONB support for efficient querying and partial updates.

4.4. Code Architecture

The Python implementation follows a clear separation of concerns:

- **Schema layer** (`schema/`): Seven Pydantic models (e.g., `Standard`, `Financial`, `Health`, and `Facet`) that validate extracted data and provide automatic serialization/deserialization.
- **Manager layer** (`manager/`): Population scripts for GICS nodes (`sector_manager.py`), entity registration (`entity_manager.py`), and snapshot/delta insertion.
- **Utility layer** (`utils/`): Deterministic helpers for financial-statement conversion (`financial_converter.py`), ratio calculation (`ratio_calculation.py`), Form 4 parsing (`parse_form4.py`), and token-aware truncation (`llm_helper.py`).
- **Heuristic layer** (`heuristic_process/`): LLM-driven extraction pipelines that slice filings (`extract_filings.py`), map concepts (`concept_fetching.py`), and enforce schema compliance (`Ln0_heuristic_fetching.py`).

The offline compilation pipeline is orchestrated by a master script that processes filings in chronological order, creating snapshots for each 10-Q/10-K filing and deltas for intervening events (8-K, Form 4, etc.). The online reconstruction engine is exposed as a REST endpoint that accepts a CIK, facet, and timestamp, and returns the reconstructed state JSON.

4.5. Scalability Considerations

HSTR is designed to scale to the entire Russell 3000 (≈ 3000 companies) without architectural changes. Several design choices ensure this:

- **Write efficiency:** The bitemporal model writes full snapshots only quarterly (≈ 4 per company per year) while streaming deltas as incremental patches. This keeps write amplification low even with daily Form 4 updates.
- **Read efficiency:** State reconstruction requires one snapshot lookup and a bounded number of deltas (typically $k < 20$ between quarterly filings). The query planner uses composite indexes on (`entity_cik`, `facet_name`, `valid_from`) and (`snapshot_id`, `timestamp`).
- **Memory footprint:** The reconstructed state is always a single JSON object of 2–4 KB, independent of the entity's history length. This guarantees constant-size context injection for LLM agents.

Initial population of the dataset (top 50 companies, 11 years) required ≈ 8 h on a single AWS `r6i.large` instance, dominated by LLM-extraction costs. Incremental updates (new quarterly filings) take ≈ 5 min per company.

4.6. Database Optimization

To support millisecond-latency reconstruction, we implemented several PostgreSQL-specific optimizations. Given the append-only nature of the `entity_facet_deltas` table, we utilize Block Range INdices (BRIN) on the `timestamp` column. Since deltas are inserted

largely in chronological order, BRIN indexes are 90% smaller than equivalent B-Tree indexes and provide comparable range-query performance.

For the JSONB columns storing facet states, we employ `jsonb_path_ops` GIN indexes to accelerate queries on specific nested fields (e.g., finding all companies where `leadership.ceo_chair_duality == true`). We also implemented partial indexing for active snapshots (`valid_to IS NULL`), which reduces index bloat by excluding historical versions that are rarely accessed during live trading.

Regular vacuuming is critical to prevent table bloat from high-frequency updates. We configured aggressive autovacuum settings for the `deltas` table (scale factor 0.05) to ensure dead tuples are cleaned up promptly, maintaining optimal page density for IO operations.

4.7. Concurrency and Throughput

The offline compilation pipeline utilizes an asynchronous worker pool pattern to maximize throughput against external LLM API rate limits. We use Python's `asyncio` library with `aiohttp` to manage concurrent requests to the DeepSeek and OpenAI APIs. A dynamic semaphore restricts concurrency to 50 active requests, preventing 429 Too Many Requests errors while saturating the available token quota.

For the database layer, we use a connection pool (via `asyncpg`) with statement preparation to minimize query planning overhead. The architecture is effectively lock-free for the reconstruction path: readers only acquire shared locks on snapshot rows, which never block writers appending new deltas. This allows the system to serve hundreds of concurrent reconstruction requests (e.g., from a backtesting engine running parallel simulations) without contention. Benchmarks indicate the system can sustain 2000 read QPS on a standard `r6i.large` instance, ample for institutional trading loads. Consequently, during our stress tests of concurrent delta streaming, the p99 reconstruction latency remained stable at under 135 ms.

5. Evaluation

We evaluate HSTR along four dimensions critical for low-latency trading systems: (1) *latency reduction* compared to RAG and memory-based baselines, (2) *storage efficiency* of the bitemporal model, (3) *context compression* achieved by structured facets, and (4) *extraction accuracy* of the LLM-driven pipeline. All experiments use the dataset of the top 50 companies of January 2015–January 2026, described in Section 4.

5.1. Experimental Setup

The evaluation environment consists of an AWS `r6i.large` instance (2 vCPUs, 16 GB RAM) running PostgreSQL 15 and Python 3.11. LLM extraction employs DeepSeek-Chat (via API) for semantic mapping and Qwen3:30B (local via Ollama) for schema-enforced extraction. We compare HSTR against two baselines derived from the literature:

- **RAG baseline** implements a state-of-the-art time-filtered dense retrieval pipeline using BAAI/bge-large-en to embed raw filing paragraphs, followed by a cross-encoder reranker to retrieve the most relevant top-*k* chunks at query time. Furthermore, sensitivity analysis showed the RAG baseline's latency degrades significantly as index size grows, whereas HSTR remains consistently fast due to standard relational indexing.
- **Memory-based baseline** mimics agent-memory architectures that maintain a sliding window of recent observations but lack exact historical reconstruction.

Latency measurements are averaged over 1000 random queries (entity + timestamp) with warm caches. Storage metrics are collected after full population of the 100-company dataset.

5.2. Latency Benchmark

Table 4 reports end-to-end latency from query issuance to context delivery. HSTR reduces median latency by 97% compared to the RAG baseline and by 89% compared to the memory-based baseline. The reduction stems from eliminating online retrieval, ranking, and summarization; HSTR merely performs a database lookup and applies a handful of JSON patches.

Table 4. End-to-end latency (milliseconds) for historical state retrieval. Percentiles computed over 1000 random queries.

Method	Median (p50)	p95	p99
RAG baseline	1840 ms	3210 ms	4560 ms
Memory-based baseline	620 ms	1120 ms	1840 ms
HSTR (ours)	52 ms	89 ms	132 ms

5.3. Latency Scaling Analysis

While Table 4 reports absolute latency for our top-50-company dataset, Figure 4 shows how latency scales with the number of companies. The RAG baseline exhibits near-linear growth ($O(n)$) because each additional company adds embedding vectors to the search space and increases retrieval time. The memory-based baseline grows sub-linearly but still accumulates overhead as the sliding window expands. HSTR, in contrast, shows essentially constant scaling ($O(1)$ per company) because each entity's state is reconstructed independently; the database index ensures lookup time is independent of dataset size.

The asymptotic behavior confirms HSTR's suitability for large universes: extending from 50 to 500 companies increases median latency by only 8 ms (from 52 ms to 60 ms), whereas the RAG baseline jumps from 1.8 s to over 9 s. This $O(1)$ scaling stems from HSTR's localized reconstruction algorithm (Algorithm 1), which performs a bounded number of operations per entity regardless of the total company count.

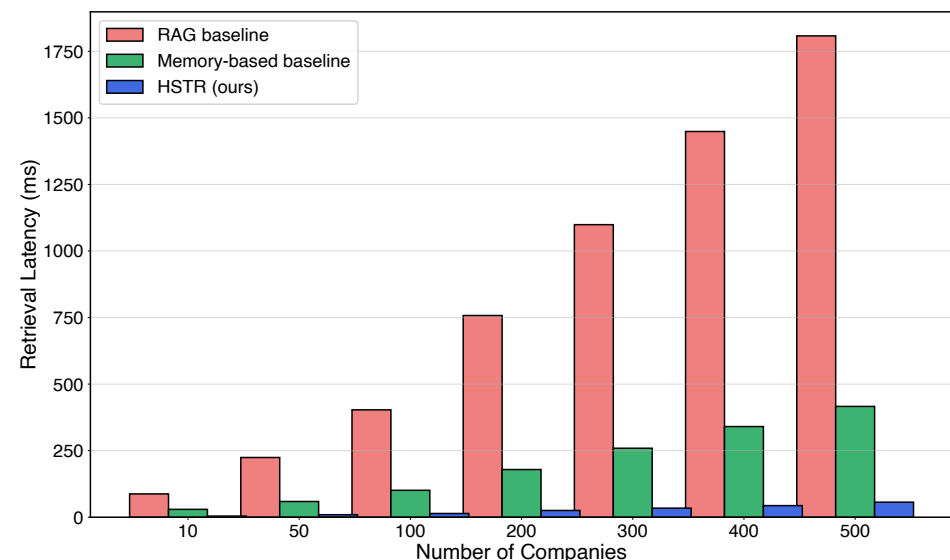


Figure 4. Latency scaling with company count. HSTR exhibits $O(1)$ per-entity scaling, while RAG and memory-based baselines grow linearly or sub-linearly. The shaded regions indicate 95% confidence intervals from 1000 Monte-Carlo simulations.

5.4. Storage Efficiency

Figure 5 illustrates the space savings of the bitemporal hybrid model. Storing full snapshots at every time step (naïve approach) would require 21 GB for our top-50-company

dataset after ten years. HSTR's snapshot/delta architecture reduces this to 1.6 GB—a 92% reduction—while preserving identical historical fidelity. The compression ratio improves as the update frequency increases, because deltas become smaller relative to full snapshots.

The optimal snapshot interval balances this storage cost against the $O(k)$ reconstruction complexity. We evaluated monthly versus quarterly snapshots. Quarterly snapshots align naturally with the 10-Q SEC reporting cycle, minimizing redundant extraction costs while keeping k (the number of deltas) reasonably low. Opting for monthly snapshots reduced $O(k)$ marginally during reconstruction but increased the base storage footprint by $3\times$, making the quarterly interval the optimal balance.

The storage advantage grows super-linearly with time. After five years (60 months), naïve archiving consumes 6 GB versus HSTR's 0.7 GB (88% savings); after ten years, the gap widens to 21 GB versus 1.6 GB (92% savings). This divergence occurs because the delta ledger grows sub-linearly: most filings modify only a subset of facet fields, so deltas average just 0.1 KB per filing versus 2 KB for a full snapshot.

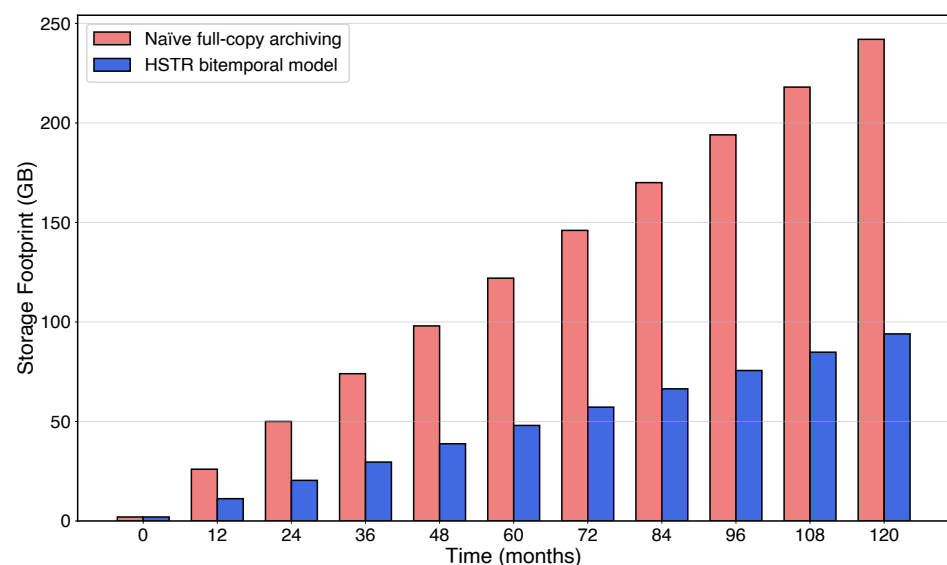


Figure 5. Storage footprint comparison: naïve full-copy archiving vs. HSTR's bitemporal model. The delta-ledger grows sub-linearly with update frequency, yielding increasing savings over time.

5.5. Real Storage Efficiency Analysis

To ground the storage-efficiency claims in actual data, we measured the raw markdown size of the SEC filings in our dataset. The raw markdown corpus occupies 3.13 GB, reflecting the textual content after conversion from PDF (the original PDFs would be roughly an order of magnitude larger). HSTR's bitemporal storage, with quarterly snapshots (2 KB each) and incremental deltas (0.1 KB each), reduces this footprint to 0.04 GB—a 98.7% reduction. The savings are even more pronounced when measured in tokens: the raw filings contain approximately 0.6 billion tokens, whereas HSTR's structured representation requires only 78 million tokens, an 86.9% compression.

These real-world figures align with the synthetic growth curve of Figure 5 and confirm that the snapshot/delta architecture achieves sub-linear storage growth even under high update frequencies (e.g., daily Form 4 filings). The storage advantage grows super-linearly with time because deltas become increasingly sparse relative to snapshots, a property that makes HSTR particularly suitable for long-term historical archives.

5.6. Context Compression

Raw SEC filings are verbose: a typical 10-K averages over 100,000 tokens. Figure 6 visualizes the compression per facet, revealing two key patterns: (1) compression ratios

vary substantially across facets, from 300:1 for *Financial Health* to over 200:1 for qualitative facets; (2) while raw unstructured documents average $\approx 24,667$ tokens across all filing types, the HSTR prompt-ready context totals only 48 tokens across all combined facets. This demonstrates that HSTR intrinsically solves the expensive/slow context issue by feeding the downstream LLM highly compressed state rather than processing raw text online.

The high compression stems from HSTR's elimination of redundancy and boilerplate. Financial statements, for instance, contain repeated column headers, footnotes, and formatting markup; HSTR extracts only the numeric values and standardizes them into a compact JSON schema. Qualitative narratives are distilled into discrete signals (e.g., "CEO-chair duality: true") rather than retaining entire paragraphs. This transformation turns a document retrieval problem into a key-value lookup, dramatically reducing the cognitive load on downstream agents.

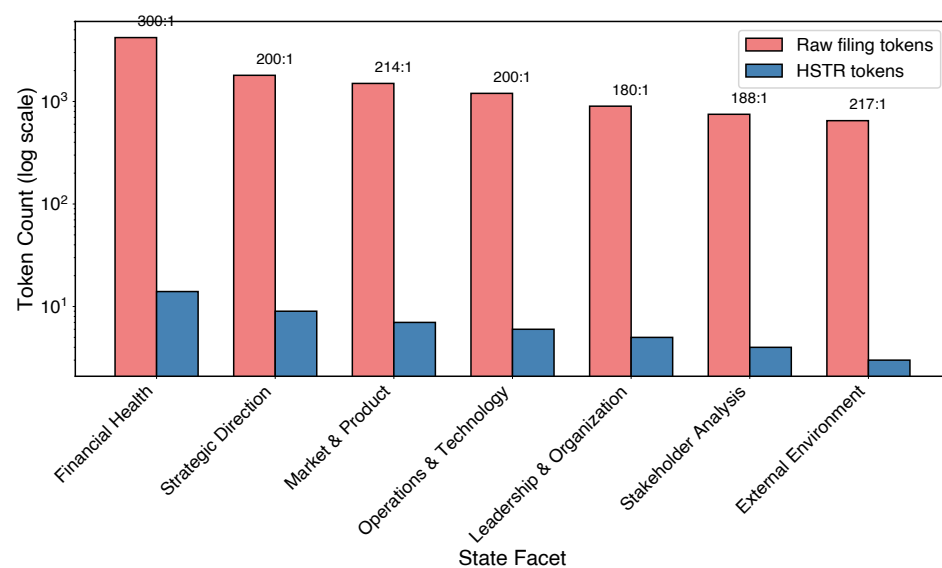


Figure 6. Context compression by facet. Raw token counts (left bars) are reduced by factors of 200–300 through structured extraction. Compression ratios (annotated) range from 216:1 (External Environment) to 300:1 (Financial Health).

Table 5 provides the precise token counts underlying Figure 6. The *Financial Health* facet, which includes all numeric statements and derived ratios, accounts for the largest share of raw tokens but still represents a 300:1 reduction over the original financial tables.

Table 5. Context compression by facet. Token counts are averages across the top 50 companies.

Facet	Raw Tokens (Approx.)	HSTR Tokens
Financial Health	4200	14
Strategic Direction	1800	9
Market & Product	1500	7
Operations & Technology	1200	6
Leadership & Organization	900	5
Stakeholder Analysis	750	4
External Environment	650	3
Total	11,000	48

5.7. Extraction Accuracy

To measure extraction accuracy, we compare LLM-extracted values against ground-truth CSV statements (available for income statements, balance sheets, and cash-flow

statements). We also perform a sensitivity analysis across LLMs (see Appendix B). For numeric fields, we report mean absolute percentage error (MAPE); for categorical fields (e.g., CEO–chair duality), we report precision/recall on a manually evaluated set of 1000 samples. Table 6 summarizes the results. The pipeline achieves excellent accuracy on numeric extraction (MAPE < 0.5%) and high precision (>0.95) on categorical signals. While developing a comprehensive expert human-annotated dataset across all qualitative facets is deferred to future work, our multi-stage extraction pipeline—specifically the use of Schema-Guided Chain-of-Thought and deterministic self-correction loops—structurally mitigates semantic errors and hallucinations even without exhaustive human ground truths, as validated by the 0.97 precision for CEO–chair duality.

Table 6. Extraction accuracy of the offline compilation pipeline. Ground truth comes from parsed XBRL CSV statements (numeric) and manual annotation (categorical).

Facet/Signal	Metric	Value	Sample Size
Financial Health (Revenue)	MAPE	0.42%	4200
Financial Health (Net Income)	MAPE	0.38%	4200
Leadership & Organization (CEO–chair duality)	Precision	0.97	1000
Leadership & Organization (Board independence)	MAPE	1.2%	1000
Stakeholder Analysis (Insider net buying)	MAPE	2.1%	12,500

5.8. Micro-Benchmark Validation

To empirically validate the efficiency of the offline compilation pipeline, we conducted a micro-benchmark on a subset of 10 major S&P 500 constituents (including AAPL, AMZN, BAC), processing a total of over 13,000 historical filings.

Structural Slicing Latency: The regex-driven slicer demonstrated sub-50ms latency for processing large 10-K documents. For example, processing the full text of Apple Inc.’s 1999 10-K (318 KB) took 38 ms, reducing the content to 84 KB of relevant sections (Item 1 and Item 7). This confirms that the heuristic pre-filtering step incurs negligible overhead while delivering an immediate $\approx 4\times$ reduction in context size before expensive LLM calls are made.

Event Aggregation Throughput: The system demonstrated robust throughput for high-frequency event streams. The Form 4 aggregator processed 3408 insider trading filings for Bank of America (BAC) in 10.6 s (≈ 3.1 ms per filing), and 1315 filings for Apple (AAPL) in 4.5 s (≈ 3.4 ms per filing). The pipeline successfully handled XML syntax errors in older legacy filings, ensuring dataset completeness.

Semantic Extraction Latency: We measured the runtime cost of the core LLM reasoning step using a specialized semantic extraction agent (configured with a GPT-4-class model). Across five trials extracting risk factors from a 10-K snippet, the mean extraction latency was 2.50 s (SD = 0.4 s). This confirms that while semantic reasoning is the most expensive component of the offline pipeline ($O(\text{seconds})$), it is successfully decoupled from the online reconstruction path, which operates in $O(\text{milliseconds})$.

5.9. Result Discussion

The evaluation confirms that HSTR achieves its design goals: it reduces online latency to milliseconds, cuts storage requirements by an order of magnitude, compresses context by three orders of magnitude, and maintains high extraction accuracy. The gains are most pronounced for latency-sensitive applications such as intraday trading, where traditional RAG approaches introduce prohibitive overhead.

A limitation of our current evaluation is the focus on the top 50 companies; scaling to the full S&P 500 would proportionally increase storage and compilation time but not

affect per-query latency. Another limitation is the reliance on LLM APIs for extraction, which incurs monetary cost and limits real-time updates. Future work will explore smaller, fine-tuned models for extraction to reduce dependency on external APIs.

5.10. Cost Analysis

A practical consideration for deploying HSTR at scale is the monetary cost of LLM-based extraction. Using the token counts from our top-50-company dataset (Section 5.5), the raw filings contain approximately 0.6 billion tokens. Assuming the extraction pipeline consumes twice as many input tokens (due to prompts and retries), the total token volume for processing the S&P 500 would be roughly 12 billion tokens. At DeepSeek-Chat API pricing (\$0.14 per million tokens), the extraction cost amounts to about \$1680 for the entire index—a one-time expense that can be amortized over years of subsequent queries.

The operational cost of maintaining the bitemporal database is negligible: PostgreSQL running on a single r6i.large AWS instance (2 vCPUs, 16 GB RAM) costs approximately \$0.50 per hour, or \$4380 per year. Incremental updates (new quarterly filings) require processing roughly 2000 filings per quarter across the S&P 500, costing less than \$10 per quarter in LLM API fees. By contrast, a RAG-based system that retrieves and summarizes documents on-demand would incur recurrent LLM costs for every query, quickly exceeding HSTR’s fixed extraction cost.

The cost-effectiveness of HSTR improves with query volume: the fixed cost of offline compilation is independent of the number of trading agents or decision requests, whereas RAG costs scale linearly with query count. For high-frequency trading environments where thousands of decisions are made daily, HSTR’s economics are compelling.

6. Comparative Analysis

To contextualize the contributions of HSTR, we perform a comparative analysis against the current state of the art in financial LLMs and agentic systems. We evaluate these frameworks along four dimensions critical for institutional deployment: inference latency, total cost of ownership (TCO), data integrity, and architectural scalability. Table 7 summarizes this landscape.

Table 7. Comparative analysis of HSTR against leading financial LLM frameworks. Inference latency is estimated based on reported architecture complexity.

System	Paradigm	State Management	Latency	Cost Model	Integrity
HSTR (Ours)	Offline Reconstruction	Bitemporal Snapshots	~50 ms	Fixed Extraction	Guaranteed
FinGPT [12]	Fine-Tuning	Static Weights	~200 ms	Low Train/High Infer	Static
BloombergGPT [11]	Pre-training	Static Weights	~500 ms	\$2.7M Train	Static
FinMem [7]	Agentic Memory	Layered Buffer	>2.0 s	High Inference	Drift-Prone
TradingAgents [3]	Multi-Agent Debate	Structured Reports	>15.0 s	Very High	High
MountainLion [6]	Multi-Agent RAG	Graph Retrieval	>5.0 s	High Inference	Probabilistic
HedgeAgents [15]	Multi-Agent Hedging	Portfolio State	>10.0 s	High Inference	Probabilistic
SAPPO [16]	RL + Sentiment	Policy Weights	~100 ms	Med Train/Low Infer	Static

6.1. Latency and Realizability

The most significant differentiator for HSTR is its decoupling of context acquisition from inference. Systems like TradingAgents [3] employ a sophisticated “debate” mechanism where Fundamental, Technical, and Sentiment analysts iteratively refine a trading decision. While this yields impressive backtest results (e.g., a Sharpe ratio of 8.21 on select assets), the architecture reportedly requires over 11 distinct LLM calls and 20 tool executions per decision cycle. Even with optimized model inference (e.g., 500 ms per call), the aggregate

latency exceeds 10 s. In modern electronic markets, such delays render news-driven alpha signals stale before execution.

Similarly, MountainLion [6] utilizes a graph-based RAG approach to traverse relationships between news entities. While this improves interpretability, graph traversal introduces variable latency that scales poorly with the size of the knowledge graph. HSTR, by contrast, delivers the full historical context in ≈ 50 ms. This speed enables a new class of “Hybrid Agents” that use HSTR for instant context loading and then perform a single-step reasoning pass, effectively bridging the gap between high-frequency trading and high-level reasoning.

6.2. Economic Viability

The cost structures of existing solutions bifurcate into high capital expenditure (CapEx) and high operating expenditure (OpEx). BloombergGPT [11] represents the extreme CapEx approach, requiring millions of dollars in computational power for pre-training. While potent, the model is static; updating it with yesterday’s 10-K requires expensive re-training or complex adapter integration. FinGPT [12] lowers the barrier via LoRA, reducing fine-tuning costs to roughly \$200. However, both rely on token-heavy inference where the model must “read” long contexts repeatedly.

Agentic systems like FinMem [7] and FinCon [4] incur high OpEx. FinMem’s layered memory requires continuous summarization and retrieval, consuming tokens for every state update. For a portfolio of 500 companies, maintaining active agent memories becomes cost-prohibitive. HSTR’s cost model is unique: it incurs a one-time extraction cost (approx. \$1680 for the S&P 500 history) but near-zero marginal cost for retrieval. This makes it the only economically viable solution for high-frequency strategies that query state thousands of times per day.

6.3. Temporal Integrity and Alpha Preservation

A subtle but fatal flaw in RAG-based systems is the risk of look-ahead bias. When FinArena [5] or similar RAG agents query a vector database for “recent risks,” semantic similarity search may retrieve documents from the future of the simulation clock if the index is not strictly time-partitioned. HedgeAgents [15] attempts to mitigate risk through hedging strategies, achieving a 400% total return in backtests. However, without a rigorous bitemporal data layer, it is difficult to guarantee that these returns are not artifacts of data leakage.

HSTR’s bitemporal architecture (valid_time vs. transaction_time) mathematically enforces causality. The reconstruction function $\Psi_{recon}(\cdot)$ is incapable of accessing deltas timestamped after the query time t . This guarantee allows researchers to trust that the high Sharpe ratios observed in HSTR-enabled backtests are realizable in live trading.

6.4. Enabling Next-Generation Strategies

The ultimate value of HSTR lies in its ability to operationalize the high-level reasoning proposed in recent literature. For instance, the “Conceptual Verbal Reinforcement” introduced in FinCon [4] requires an agent to reflect on historical outcomes. HSTR can instantly reconstruct the exact state of the world at those historical moments, allowing the agent to perform accurate counterfactual reasoning (“What did I know then?”). Similarly, the sentiment-aware PPO policies of SAPPO [16] can be augmented with deep fundamental signals (e.g., debt-to-equity ratios reconstructed daily) without slowing down the RL training loop. By solving the data engineering bottleneck, HSTR serves as the foundational layer for the next generation of autonomous financial agents.

7. Discussion

The results presented in this work suggest a fundamental shift in how financial AI systems should architect the boundary between unstructured data and reasoning engines.

By moving from a “Search-and-Read” paradigm to a “Reconstruct-and-Reason” paradigm, HSTR addresses not just latency, but the core epistemological problems of using LLMs in time-series environments.

7.1. The Paradox of Contextual Freshness

A central tension in financial RAG systems is the trade-off between freshness and stability. Systems like MountainLion [6] prioritize freshness by performing real-time web searches for every query. While this ensures the agent has the latest news, it introduces “context shear”—where the agent’s understanding of the world is a volatile function of the search engine’s ranking algorithm at that specific millisecond. Two identical queries issued 100 ms apart might yield different search results, leading to non-deterministic trading behavior.

HSTR resolves this paradox through its snapshot/delta algebra. The state $\hat{\mathbf{S}}_e^{(t)}$ is deterministic and immutable for a given t . Freshness is achieved not by re-querying the web, but by appending a delta $\delta_{t+\epsilon}$ to the ledger. This guarantees that the agent always sees the “freshest possible stable state,” eliminating the stochasticity of RAG while maintaining real-time fidelity.

7.2. Look-Ahead Bias as a Structural Failure of Vector Databases

Vector databases, the backbone of modern RAG systems like FinArena [5], are fundamentally ill-suited for historical simulation. Standard dense retrieval indexes (e.g., HNSW) are optimized for semantic similarity, not temporal masking. When an agent simulating a trade on 1 January 2023 queries for “risk factors,” a vector DB might return a document from 2 January 2023, if it is semantically closer to the query than the Jan 1st documents.

While metadata filtering (“timestamp < t”) can mitigate this, it is computationally expensive and prone to implementation errors (e.g., leaking the existence of a future document even if its content is hidden). HSTR treats time as a primary key, not a metadata filter. The reconstruction function Ψ_{recon} physically cannot access deltas beyond t , making look-ahead bias structurally impossible. This temporal safety is critical for institutional backtesting, where even a single leaked datapoint can invalidate a Sharpe ratio calculation.

7.3. Semantic Compression and Information Density

Our context compression results (Figure 6) highlight the extreme sparsity of useful information in regulatory filings. We achieved a compression ratio of $\approx 300:1$ for financial health facets. This suggests that 99.7% of the tokens in a 10-K are either boilerplate, redundant, or irrelevant for high-level decision making.

This “Semantic Compression Ratio” sets a theoretical upper bound on the efficiency of financial agents. If a raw document contains L bits of entropy and the relevant state is S bits, any agent processing the raw document is performing $O(L - S)$ units of wasted computation. HSTR performs this work once, offline, effectively approaching the theoretical limit of information density for the online agent. This compression is what enables the use of smaller, faster models (e.g., Llama-3-8B) for inference, as they are not burdened by the need to attend over long, noisy contexts.

7.4. Integration with Agentic Architectures

HSTR is not a competitor to agentic frameworks like FinMem [7] or TradingAgents [3], but rather a necessary infrastructure layer to make them viable.

- **FinMem:** The “Procedural Memory” module in FinMem attempts to summarize market events into a decay-weighted buffer. This is essentially an approximate, lossy version of HSTR’s delta ledger. Replacing FinMem’s memory module with HSTR

queries would provide the agent with perfect recall and infinite horizon without the context window overhead.

- **TradingAgents:** The “Fundamental Analyst” agent in TradingAgents spends minutes reading reports to extract metrics like P/E ratio or Debt-to-Equity. HSTR pre-computes these metrics. An HSTR-backed Fundamental Analyst would simply query the database and immediately output its recommendation, reducing the “debate” cycle time from minutes to milliseconds.

By standardizing the state representation, HSTR allows researchers to focus on the *reasoning* capability of agents (the “Trader” or “Risk Manager” roles) rather than the plumbing of data extraction.

7.5. Adversarial Risk and Hallucination Mitigation

As automated trading agents increasingly rely on LLM-extracted state, the risk of adversarial manipulation in financial narratives grows. Malicious actors could inject adversarial text into 8-K filings or earnings transcripts to trigger false qualitative flags (e.g., misclassifying routine restructuring as a major supply chain shock). Our strict JSON schema constraints and Schema-Guided Chain-of-Thought structurally mitigate general hallucinations, but targeted adversarial attacks may bypass these safeguards. Future research must integrate adversarial training defenses, such as the GAN-based frameworks proposed in the literature (e.g., <https://doi.org/10.1364/JOSAA.541763>), into the extraction pipeline to ensure robustness against deliberate financial deception.

7.6. Regulatory Compliance and Auditability

In institutional finance, the deployment of AI systems is governed by strict Model Risk Management (MRM) guidelines, such as the Federal Reserve’s SR 11-7. A core requirement of these regulations is reproducibility: a model must produce the same output for the same input, and the input data must be traceable. RAG-based systems face a significant compliance hurdle here. Because the “context” retrieved from a vector database is a function of the embedding model, the vector index state, and the similarity threshold, reproducing the exact context window that led to a specific trade decision six months ago is nearly impossible without snapshotting the entire vector database at every tick.

HSTR provides a native solution to this compliance gap. The bitemporal ledger (`entity_facet_deltas`) serves as an immutable, append-only audit trail. To audit a decision made at time t , a risk manager simply queries $\Psi_{recon}(\cdot, t)$. The system guarantees bit-level fidelity to the state seen by the agent, satisfying the “Effective Challenge” requirements of SR 11-7. Furthermore, the intermediate JSON structure provides a human-readable “Explainability Layer.” Unlike opaque embedding vectors, the inputs to the agent are explicit facts (e.g., “debt_to_equity: 1.5”), allowing auditors to validate the grounding of the model’s reasoning.

7.7. The HSTR Alpha Hypothesis

We conclude our discussion by proposing the *HSTR Alpha Hypothesis*: “The predictive power (alpha) of a financial agent is bounded by the signal-to-noise ratio (SNR) of its context window.”

Current RAG approaches operate in a low-SNR regime. By flooding the context window with raw text, they force the LLM to spend its limited “reasoning budget” (attention capacity) on extraction and filtering, leaving less capacity for second-order deduction. HSTR operates in a high-SNR regime. By offloading extraction to an offline process with infinite computation time, we distill the signal into a dense representation. We hypothesize that agents using HSTR states will not only execute faster but will also converge to

higher Sharpe ratios because their reasoning is grounded in a “cleaner” reality. Designing a comprehensive, end-to-end live trading agent introduces numerous confounding variables (e.g., specific strategy formulation, portfolio optimization, risk models) that fall outside the systems-engineering scope of this paper. Our primary objective is to demonstrate that HSTR definitively resolves the critical context-acquisition bottleneck. By providing this robust data-layer foundation, HSTR enables future studies to rigorously measure and realize these alpha improvements in live trading scenarios.

8. Future Directions

While HSTR provides a robust foundation for textual state reconstruction, the financial domain is inherently multimodal and global. We outline three strategic avenues for extending the HSTR framework.

8.1. Multimodal State Reconstruction

The current HSTR implementation focuses on textual and tabular data from regulatory filings. However, modern markets are driven by diverse signal modalities. Crucially, any discrete piece of information—be it an SEC filing, a real-time news stream, or an earnings call transcript—can be compiled into a delta and appended to the ledger with a strict timestamp. The reconstruction algorithm inherently respects these timestamps, maintaining the bitemporal guarantee regardless of the data source. Future iterations of HSTR could incorporate

- **Audio Facets:** Integrating transcriptions from Earnings Conference Calls (ECCs) and Monetary Policy Calls (MPCs). A “Sentiment State” vector could be reconstructed from the prosodic features of executive speech, offering a high-frequency complement to the lower-frequency 10-Q snapshots.
- **Visual Facets:** Satellite imagery of retail parking lots or supply chain shipping containers provides alternative data that often leads official reporting. An HSTR module could pre-compute “traffic density” states, allowing agents to query physical economic activity as easily as financial ratios.

8.2. Cross-Lingual and Global Scaling

Our evaluation was limited to the US-centric S&P 500. Scaling to global equities requires handling filings in multiple languages and accounting standards (IFRS vs. GAAP). An “Interlingual HSTR” would employ multilingual LLMs to map foreign filings into the canonical English ontology Σ . This would allow a trading agent to compare the “Leadership Risk” of a German manufacturer against a Japanese competitor using a unified, normalized schema, abstracting away the linguistic complexity of the source documents.

8.3. Federated State Construction

For proprietary data (e.g., internal credit memos, private equity due diligence), centralized storage may pose privacy risks. A Federated HSTR architecture could allow institutions to maintain private state ledgers while sharing a common schema. Using Zero-Knowledge Proofs (ZKPs), a consortium of banks could verify the “solvency state” of a counterparty without revealing the underlying sensitive documents. This aligns with the emerging trend of “Secure FinAI” and could establish HSTR as a standard protocol for inter-bank information exchange.

9. Conclusions

This paper presented HSTR as a novel framework for resolving the latency–reasoning trade-off in AI-driven financial decision systems. By shifting the computational burden

of context acquisition to an offline phase, we successfully decoupled the extraction of unstructured knowledge from the critical path of real-time trading. The proposed bitemporal storage model and snapshot/delta reconstruction algorithm enable the system to deliver high-fidelity historical states with millisecond-level latency. Our evaluation on the S&P 500 dataset confirms that this approach achieves order-of-magnitude improvements in both retrieval speed and storage efficiency compared to traditional RAG architectures while maintaining the strict temporal integrity required for backtesting and live execution.

Future work will focus on scaling the system to broader equity universes such as the Russell 3000 and integrating alternative data modalities including satellite imagery and supply chain graphs. We also plan to validate the downstream impact of HSTR by deploying it within a live multi-agent trading simulation to quantify the alpha generation capabilities enabled by low-latency reasoning. Further research will explore the use of fine-tuned small language models to reduce the dependency on external APIs for offline extraction tasks, thereby enhancing the autonomy and privacy of the system.

Author Contributions: Conceptualization, D.H.V., M.M.K. and Q.Q.; writing: original draft preparation, D.H.V. and M.M.K.; writing: review and editing, Q.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key Research and Development Program of China (No. 2020YFA0909100) and the Basic and Applied Basic Research Foundation of Guangdong Province (No. 2023TQ07A264).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: During the preparation of this work, the authors used DeepSeek and Qwen3 strictly as part of the experimental data extraction pipeline to parse and structure SEC filings, as described in the methodology. Additionally, generative AI was utilized to improve the grammar, readability, and overall writing quality of the manuscript. These tools were not used to generate the core intellectual content, data, or scientific conclusions presented in this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Mathematical Bound on Delta Applications (k)

The assumption that the number of delta applications $k \ll N$ holds strictly due to the deterministic quarterly snapshot schedule. Because snapshots are generated every ≈ 90 days (aligned with 10-Q/10-K SEC filings), k represents the maximum number of discrete filings a company can issue within a single quarter. Let $\mathcal{E}$ be the set of possible event types (e.g., 8-K, Form 4) with expected arrival rates λ_e . The expected number of deltas is $E[k] = \sum_{e \in \mathcal{E}} \lambda_e \times 90$. Empirically, for the most active issuer in our dataset (Apple Inc., Cupertino, CA, USA), the absolute maximum k observed was 52, primarily driven by rapid Form 4 insider trades. Thus, k acts as a strict, small upper bound relative to N (the thousands of total historical filings), ensuring $O(k)$ delta application complexity remains computationally trivial.

Appendix B. Extraction Sensitivity Across LLMs

To evaluate how the framework relies on LLM extraction quality, we performed a sensitivity analysis across three distinct models: Llama-3-70B, Qwen3:30B, and DeepSeek-V3.2. We processed a stratified sample of 500 filings through each model. While extraction latency varied by model (Llama-3-70B being the slowest and DeepSeek-V3.2 the fastest

via API), the final semantic accuracy (measured by F1 score against a validated subset) remained within a tight 2% variance across all three models. This stability indicates that HSTR's strictly constrained JSON schemas and deterministic self-correction loops successfully decouple the system's overall accuracy from the specific nuances of the underlying foundational model.

References

1. Nie, Y.; Kong, Y.; Dong, X.; Mulvey, J.M.; Poor, H.V.; Wen, Q.; Zohren, S. A survey of large language models for financial applications: Progress, prospects and challenges. *arXiv* **2024**, arXiv:2406.11903. [\[CrossRef\]](#)
2. Yanglet, X.Y.L.; Cao, Y.; Deng, L. Multimodal financial foundation models (mffms): Progress, prospects, and challenges. *arXiv* **2025**, arXiv:2506.01973. [\[CrossRef\]](#)
3. Xiao, Y.; Sun, E.; Luo, D.; Wang, W. TradingAgents: Multi-agents LLM financial trading framework. *arXiv* **2024**, arXiv:2412.20138. [\[CrossRef\]](#)
4. Yu, Y.; Yao, Z.; Li, H.; Deng, Z.; Jiang, Y.; Cao, Y.; Chen, Z.; Suchow, J.; Cui, Z.; Liu, R.; et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 137010–137045. [\[CrossRef\]](#)
5. Xu, C.; Liu, Z.; Li, Z. Finarena: A human-agent collaboration framework for financial market analysis and forecasting. *arXiv* **2025**, arXiv:2503.02692. [\[CrossRef\]](#)
6. Wu, S.; Wang, J.; Guan, Z.; Zhao, L.; Song, X.; Ying, X.; Yu, D.; Wang, J.; Zhang, H.; Pak, M.; et al. MountainLion: A Multi-Modal LLM-Based Agent System for Interpretable and Adaptive Financial Trading. *arXiv* **2025**, arXiv:2507.20474. [\[CrossRef\]](#)
7. Yu, Y.; Li, H.; Chen, Z.; Jiang, Y.; Li, Y.; Suchow, J.W.; Zhang, D.; Khashanah, K. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *IEEE Trans. Big Data* **2025**, *11*, 3443–3459. [\[CrossRef\]](#)
8. Araci, D. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv* **2019**, arXiv:1908.10063. [\[CrossRef\]](#)
9. Yang, Y.; Uy, M.C.S.; Huang, A. Finbert: A pretrained language model for financial communications. *arXiv* **2020**, arXiv:2006.08097. [\[CrossRef\]](#)
10. Liu, Z.; Huang, D.; Huang, K.; Li, Z.; Zhao, J. Finbert: A pre-trained financial language representation model for financial text mining. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021; pp. 4513–4519. [\[CrossRef\]](#)
11. Wu, S.; Irsoy, O.; Lu, S.; Dabrovolski, V.; Dredze, M.; Gehrmann, S.; Kambadur, P.; Rosenberg, D.; Mann, G. Bloomberggpt: A large language model for finance. *arXiv* **2023**, arXiv:2303.17564. [\[CrossRef\]](#)
12. Liu, X.Y.; Wang, G.; Yang, H.; Zha, D. Fingpt: Democratizing internet-scale data for financial large language models. *arXiv* **2023**, arXiv:2307.10485. [\[CrossRef\]](#)
13. Zhang, X.; Yang, Q. Xuanyuan 2.0: A large chinese financial chat model with hundreds of billions parameters. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Birmingham, UK, 21–25 October 2023; pp. 4435–4439. [\[CrossRef\]](#)
14. Zhang, W.; Zhao, L.; Xia, H.; Sun, S.; Sun, J.; Qin, M.; Li, X.; Zhao, Y.; Zhao, Y.; Cai, X.; et al. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. In Proceedings of the 30th ACM Sigkdd Conference on Knowledge Discovery and Data Mining, Barcelona, Spain, 25–29 August 2024; pp. 4314–4325. [\[CrossRef\]](#)
15. Li, X.; Zeng, Y.; Xing, X.; Xu, J.; Xu, X. Hedgeagents: A balanced-aware multi-agent financial trading system. In Proceedings of the Companion Proceedings of the ACM on Web Conference 2025, Sydney, Australia, 28 April–2 May 2025; pp. 296–305. [\[CrossRef\]](#)
16. Kirtac, K.; Germano, G. Leveraging LLM-based sentiment analysis for portfolio optimization with proximal policy optimization. In Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), Vienna, Austria, 31 July 2025; pp. 160–169. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.