MC-TDC: Mutability Classification with Type-Decoupled Cache for Computational Cost Reduction in Generative AI Systems

Antonio Telesca
Independent Researcher, Italy
antoniotelesca37@gmail.com
Submitted: March 15, 2026

***Abstract*—Large Language Models (LLMs) consume tokens proportional to the volume of information processed at each inference call. A significant and growing fraction of this consumption is redundant: models repeatedly process information that is either unchanged since the previous interaction or identical across users, sessions, and documents. Existing mitigation strategies—prompt caching, semantic caching, and context engineering—operate at the granularity of entire prompts or sessions and fail to distinguish between information units that change over time (mutable) and those that do not (immutable) within the same corpus. We introduce MC-TDC (Mutability Classification with Type-Decoupled Cache), a three-level architectural framework that: (1) performs a one-time semantic analysis of an information corpus to classify each information unit as MUTABLE or IMMUTABLE, producing a Mutability Map; (2) maintains a global shared cache organized by information type (decoupled from documents, sessions, and users) with semantically differentiated Time-To-Live (TTL) values; and (3) resolves mutable values on-demand at access time, substituting them inline while preserving immutable content untouched. We demonstrate that MC-TDC reduces the computational complexity of information updates from $O(N \times M)$ to $O(K)$, where N is the number of corpora, M is the average number of information units per corpus, and K is the number of distinct information types ($K \ll N \times M$). Through a reference implementation (LiveInk) deployed on a production editorial platform, we measure a 99.998% reduction in API calls for factual data updates across 1M documents. We further present analytical cost models for five additional application domains—code agents, customer support, legal analysis, clinical decision support, and content generation—projecting token savings between 50% and 99% depending on the domain's immutability ratio. MC-TDC represents a new paradigm in LLM inference optimization: rather than caching responses or compressing contexts, it classifies information at the semantic level and eliminates redundant computation at its source.**

***Index Terms*—Large Language Models, Token Optimization, Semantic Caching, Mutability Classification, Distributed Cache, Inference Cost Reduction, Real-Time Data, Editorial AI, Context Engineering**

I. Introduction

A. The Token Economy Crisis

The global economy of artificial intelligence is entering a phase of unprecedented growth. According to Gartner (January 2026), worldwide AI spending is forecast to reach $2.52 trillion in 2026, a 44% year-over-year increase [1]. The AI API market alone was valued at $48.5 billion in 2024 and is projected to reach $246.87 billion by 2030 (Grand View Research) [2]. At the core of this economy lies a fundamental unit of computation: the token.

Every interaction with a Large Language Model (LLM)—whether GPT-4, Claude, Gemini, LLaMA, or their successors—consumes tokens proportional to the input provided and the output generated. The economic cost of inference is directly proportional to token consumption. As LLM adoption scales from millions to billions of daily interactions, the aggregate cost of token consumption becomes a defining constraint on the viability of AI-powered systems.

A critical and largely unaddressed problem is that a substantial fraction of token consumption is redundant. We define "redundant token consumption" as the processing of information that: (a) has not changed since the last time it was processed; (b) is identical to information already processed for a different user, document, or session; or (c) is structurally invariant and need not be re-analyzed.

Industry data supports the magnitude of this waste. A study documented on Reddit (March 2026) reports that AI coding agents perform 15–20 tool calls to orient themselves in a project before beginning productive work on each task [3]. An analysis published on Medium (February 2026) estimates that 80% of tokens in coding agent sessions are consumed for "finding things," not solving problems [4]. A report on LinkedIn (January 2026) documents that LLM redundant computation costs enterprises $15K–$35K per month for document reprocessing alone [5], [15]. Research from arXiv (March 2026) on the "Missing Memory Hierarchy" for LLM context measured 21.8% structural waste across 857 production sessions and 4.45 billion tokens [6].

B. Limitations of Existing Approaches

The research community and industry have proposed several strategies to reduce token consumption. We categorize them into four families and identify the fundamental limitation shared by all.

Prompt Caching (Anthropic, 2024; OpenAI, 2024). Providers offer server-side caching of prompt prefixes. If a new request shares the same token prefix as a cached one, the cached portion costs up to 90% less [7]. However, prompt caching operates on exact prefix match only, does not

distinguish mutable from immutable content within the same prompt, does not share across users or documents, expires after minutes of inactivity, and is vendor-locked.

Semantic Caching (GPTCache [8], MeanCache [9], Redis [10], LiteLLM, Bifrost). These systems cache prompt-response pairs and return cached responses for semantically similar queries (measured via embedding distance). GPTCache reports up to 68.8% API call reduction [11]. Microsoft's semantic caching framework (February 2026) uses learning-based eviction policies [12]. However, semantic caching operates at the granularity of entire queries—if even one element of a query changes, the cache misses. It does not decompose queries into components with different mutability profiles.

Context Engineering (Anthropic, September 2025; Weaviate, 2025). These techniques curate the optimal token set during inference through compression, dynamic selection, and memory management [13]. Progressive context loading (October 2025) reduces token usage from 150K to 2K [14]. However, context engineering operates within a single session, does not classify information mutability, and does not maintain a shared cache across sessions, users, or documents.

Traditional Data Caching (Redis, Memcached, Cloudflare KV). Key-value stores with TTL are well-established. However, no existing system combines traditional caching with automatic semantic classification of information mutability within natural language text, nor organizes the cache by information type decoupled from the source document.

C. The Gap: No Sub-Document Mutability-Aware Caching

The fundamental limitation shared by all existing approaches is that they operate at the wrong granularity. Table I summarizes this gap.

None of the existing approaches: (a) automatically classifies individual information units within a corpus as mutable or immutable; (b) decouples cache organization from documents/sessions to information types; (c) applies semantically differentiated TTLs based on information volatility categories; (d) performs inline substitution preserving the original structural context. MC-TDC addresses all four.

D. Contributions

This paper makes the following contributions:

1) We formalize the concept of *Mutability Classification*—the binary semantic classification of information units within a corpus as MUTABLE or IMMUTABLE—and define domain-specific classification rules for six application domains.
2) We introduce *Type-Decoupled Cache* (TDC), a cache architecture where the key space is organized by information type rather than by document, session, or user, with TTLs differentiated by semantic volatility category.
3) We prove that the combination of (1) and (2) reduces the computational complexity of information updates from $O(N \times M)$ to $O(K)$, where K is the number of distinct information types.
4) We present LiveInk, a production reference implementation for the editorial domain, deployed on Cloudflare Workers with D1 and KV, covering 21 data categories and 45 data types.
5) We provide analytical cost models for six application domains, demonstrating projected savings between 50% and 99.998%.
6) We design and implement an embeddable widget protocol that enables third-party platforms to consume MC-TDC services via a sub-5KB client-side component with server-side license validation.

II. Formal Framework

A. Definitions

Definition 1 (Information Corpus). An information corpus C is an ordered collection of information units:

$$C = \{u_1, u_2, \ldots, u_M\} \tag{1}$$

where each u_i is an atomic piece of factual information within the corpus (a numeric value, a factual statement, a structural pattern, a policy rule, a clinical guideline, etc.).

Definition 2 (Information Type). An information type τ is a semantic identifier drawn from a controlled vocabulary V specific to the application domain:

$$\tau \in V = \{\tau_1, \tau_2, \ldots, \tau_K\} \tag{2}$$

Examples: τ = "bitcoin_usd" (editorial), τ = "hono_endpoint_pattern" (software), τ = "return_policy" (customer support), τ = "clause_confidentiality_standard" (legal).

Definition 3 (Mutability Function). The mutability function μ maps each information unit to a binary classification:

$$\mu : U \to \{\text{MUTABLE}, \text{IMMUTABLE}\} \tag{3}$$

The function μ is implemented via domain-specific rules R_d (Section 2.3).

Definition 4 (Mutability Map). Given a corpus C, the Mutability Map $\Phi(C)$ is the structure:

$$\Phi(C) = \{(u_i, \text{pos}_i, \tau_i, \mu(u_i)) \mid u_i \in C\} \tag{4}$$

where pos_i is the position of u_i within C, and τ_i is the information type of u_i.

Definition 5 (Type-Decoupled Cache). A Type-Decoupled Cache Γ is a key-value store where:

$$\Gamma : V \to (\text{value}, \text{formatted}, \text{source}, \text{timestamp}) \tag{5}$$

The key is an information type $\tau \in V$ (not a document ID, session ID, or user ID). The cache is global: a single entry for τ serves all corpora that reference τ.

Definition 6 (Semantic TTL Function). The TTL function assigns a time-to-live to each information type based on its semantic volatility category:

$$\text{TTL} : V \to \mathbb{R}^+, \quad \text{TTL}(\tau) = \sigma(\text{category}(\tau)) \tag{6}$$

TABLE I
COMPARISON OF CACHING APPROACHES FOR LLM SYSTEMS. M/I = MUTABLE/IMMUTABLE.

Approach	Granularity	Mutability Class.	Cross-doc Sharing	Semantic TTL
Prompt Caching	Full prompt	None	None	None
Semantic Caching	Full query	None	Limited	Uniform
Context Engineering	Session	None	None	None
Traditional Cache	Key-value	None	Yes	Uniform
MC-TDC (ours)	**Info unit**	**Binary (M/I)**	**Global by type**	**Semantic**

TABLE II
SEMANTIC VOLATILITY TAXONOMY WITH TTL ASSIGNMENTS.

Category	TTL (s)	Rationale
REAL_TIME_FINANCIAL	60–300	Continuous price changes
COMMODITIES_ENERGY	900	Exchange-traded
STOCK_INDICES	300	Intraday volatility
OPERATIONAL	300	Order status, availability
MONETARY_POLICY	3,600	Policy meeting driven
WEATHER_ENVIRONMENT	1,800	Gradual change
MACROECONOMICS	86,400	Monthly/quarterly relea
SOFTWARE_VERSIONS	86,400	Release cycles
SOFTWARE_PATTERNS	604,800	Conventions stable
CLINICAL_GUIDELINES	2,592,000	Expert consensus
LEGAL_STANDARD	2,592,000	Boilerplate, consolidate

B. Volatility Categories and TTL Assignment

We define a taxonomy of semantic volatility categories with empirically calibrated TTL ranges (Table II).

C. Domain-Specific Classification Rules

The mutability function μ is implemented through domain-specific rule sets R_d.

Rule Set $R_{\text{editorial}}$: $\mu(u)$ = IMMUTABLE if u contains an explicit past date, uses past-tense verbs with temporal anchoring, or represents historical variation. $\mu(u)$ = MUTABLE if u references a present or implicitly current value (price, rate, index).

Rule Set R_{software}: $\mu(u)$ = IMMUTABLE if u represents an architectural pattern, naming convention, or framework choice. $\mu(u)$ = MUTABLE if u represents a dependency version, runtime state, or environment configuration.

Rule Set R_{support}: $\mu(u)$ = IMMUTABLE if u represents a consolidated company policy or SOP. $\mu(u)$ = MUTABLE if u represents a price, availability, or order status.

D. Complexity Analysis

Theorem 1 (Complexity Reduction). Let N be the number of information corpora, M the average number of mutable information units per corpus, and K the number of distinct information types. The traditional approach requires $O(N \times M)$ update operations per refresh cycle. MC-TDC requires $O(K)$ operations per refresh cycle, where $K \ll N \times M$.

Proof. In the traditional approach, each corpus C_j contains M mutable units requiring independent updates. Total operations: $T_{\text{traditional}} \approx N \times M$. In MC-TDC, the cache Γ contains K distinct entries. Each is refreshed at most once per TTL. Total operations: $T_{\text{MC-TDC}} \leq K$. Since K is typically 30–10,000 while $N \times M$ can be millions, $T_{\text{MC-TDC}}/T_{\text{traditional}} \to 0$ as $N \to \infty$. □

III. SYSTEM ARCHITECTURE

MC-TDC comprises three cooperative levels. Figure 1 presents the overall architecture.

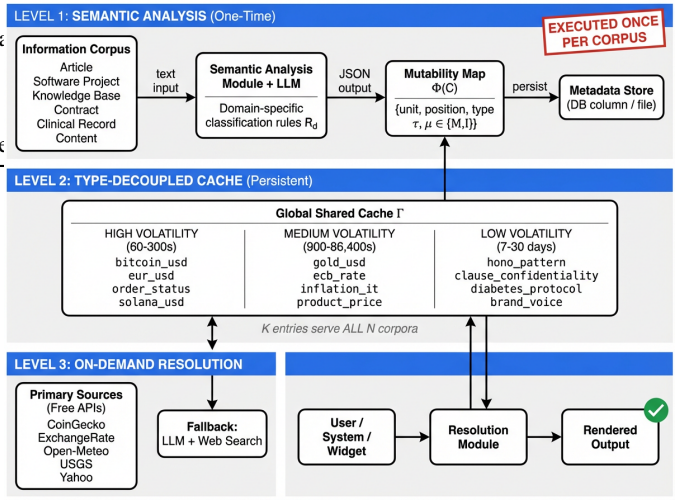

Fig. 1. MC-TDC System Architecture comprising three levels: Semantic Analysis (one-time), Global Shared Cache (persistent, type-decoupled), and On-Demand Resolution.

A. Level 1: Semantic Analysis

The Semantic Analysis Module receives an information corpus C and produces its Mutability Map $\Phi(C)$. It is implemented as a call to an LLM (Anthropic Claude Sonnet) with domain-specific rules. The output is a JSON array:

```
[
  {
    "entity": "original text fragment",
    "value_written": "value as written",
    "entity_type": "CATEGORY",
    "temporal_type": "DYNAMIC" | "STATIC",
    "needs_update": true | false,
    "confidence": 0.0 - 1.0,
    "resolve_key": "category:asset_currency"
  }
]
```

This analysis is performed **once** per corpus and persisted. Amortized cost per access is zero. Cost: $0.005–$0.02 per corpus.

B. Level 2: Type-Decoupled Cache

The cache is a distributed key-value store (Cloudflare KV) with keys decoupled from documents:

```
Key:
"liveink:v2:{category}:{asset}_{currency}"
TTL: σ(category) seconds
```

The critical design principle is type-decoupling: the key contains no reference to any specific document, user, or session. One cache write serves potentially millions of reads across different documents and users.

C. Level 3: On-Demand Resolution

When a user accesses a corpus, the Resolution Module executes the algorithm shown in Figure 2.

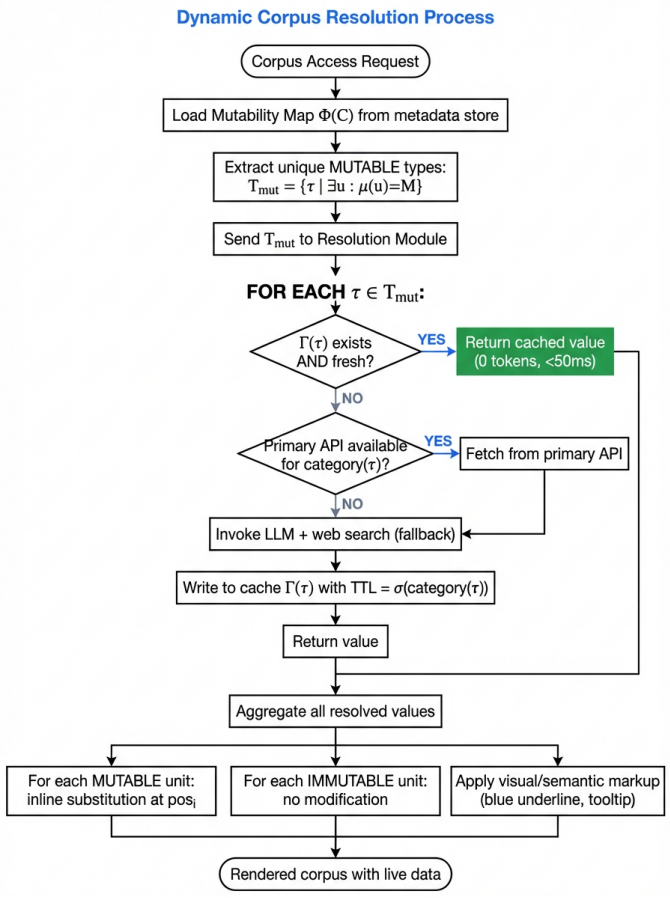

Fig. 2. On-Demand Resolution Flowchart showing the hierarchical fetch strategy with cache lookup, primary API, and LLM fallback.

The hierarchical fetch strategy uses two tiers: Tier 1 (Free APIs: CoinGecko, Open-Meteo, USGS, Yahoo Finance) and Tier 2 (LLM + Web Search Fallback), activated only on cache miss.

IV. Reference Implementation: LiveInk

A. System Overview

LiveInk is a production-grade Progressive Web Application (PWA) for editorial content. Technology stack: Hono on Cloudflare Workers (Backend), Cloudflare D1 (Database), Cloudflare KV (Cache), Vue 3 (Frontend), Anthropic Claude Sonnet (AI).

B. Data Coverage

The system covers 21 journalistic data categories and 45 individual data types (Table III).

TABLE III
LiveInk Data Coverage (Editorial Domain).

Category	Type	TTL (s)	Data Types
Cryptocurrencies	CRYPTO	60	BTC, ETH, SOL...
Forex	FOREX	300	EUR/USD, GBP...
Energy	ENERGY	900	Oil, Gas
Precious Metals	METALS	900	Au, Ag, Pt, Pd
Agriculture	AGRICULTURE	3,600	Wheat, Coffee...
Stock Indices	STOCK_INDEX	300	S&P500, NASDAQ...
Interest Rates	INTEREST_RATE	3,600	ECB, Fed, BoE
Macroeconomics	MACRO	86,400	Inflation, GDP...
Weather	WEATHER	1,800	Rome, Milan...
Air Quality	AIR_QUALITY	3,600	AQI Rome...
Earthquake	EARTHQUAKE	300	USGS data
Demographics	DEMOGRAPHICS	86,400	IT, US, World
Total: 21 categories			**45 data types**

C. API Endpoints

Key endpoints include `POST /api/analyze-entities` (Cost: $0.005–$0.02, executed once) and `POST /api/resolve-values` (Cost: ∼$0, cache hit).

D. Widget Protocol

The widget integration requires a simple script tag and data attributes, enabling third-party sites to consume MC-TDC services securely (Figure 3).

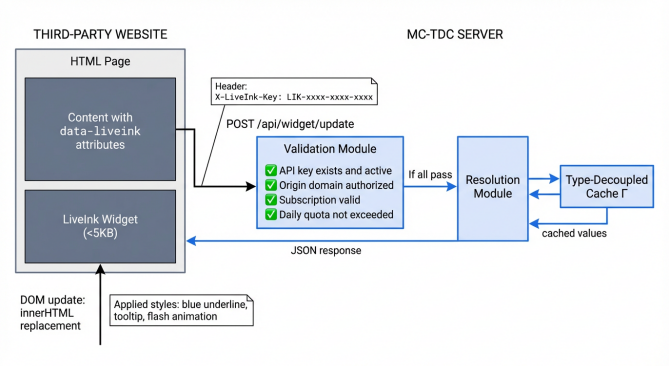

Fig. 3. Embeddable Widget Architecture connecting third-party websites to the MC-TDC server with domain-locked API key validation.

Security model: zero business logic in widget (<5KB), no API keys exposed, domain-locked keys with daily quotas, three subscription tiers (Basic: 1K req/day, €9.99/mo; Pro: 10K req/day, €29.99/mo; Enterprise: 100K req/day, €99.99/mo).

V. Cross-Domain Application Analysis

A. Software Development—AI Code Agents

AI coding agents spend 33–80% of tokens on orientation [3], [4]. MC-TDC maps the codebase once: architectural patterns are IMMUTABLE (TTL 7 days), dependency versions are MUTABLE (TTL 24h). Figure 4 illustrates the token consumption comparison. Projected savings: 50–80%.

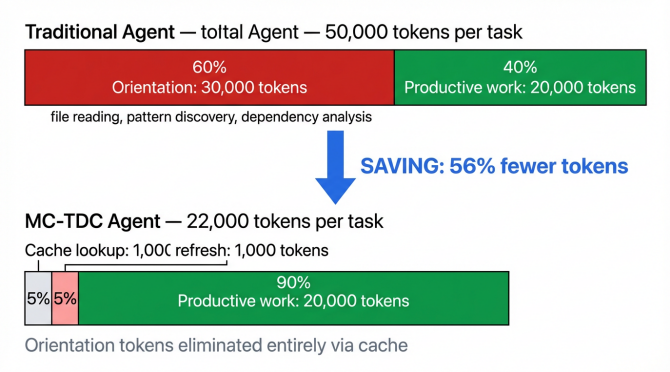

Fig. 4. Token consumption comparison for AI code agents: traditional (50,000 tokens/task, 60% orientation) vs. MC-TDC (22,000 tokens/task, orientation eliminated via cache).

B. Customer Support

Chatbots reprocess knowledge bases repetitively. MC-TDC caches policies as IMMUTABLE and prices as MUTABLE. Projected savings: 70–95%.

C. Legal—Contract Analysis

Standard clauses are IMMUTABLE (TTL 30 days); specific terms are MUTABLE. Analysis reduces from 40 clauses to ∼8 non-standard ones. Savings: 60–80%.

D. Clinical Decision Support

Guidelines and drug interactions are IMMUTABLE; patient vitals are MUTABLE. Savings: 70–90% per consultation.

E. Content Generation

Brand voice and style guides are IMMUTABLE. Savings: 40–60% per task.

VI. Quantitative Cost Analysis

We present analytical cost models for all analyzed domains (Figure 5, Table IV).

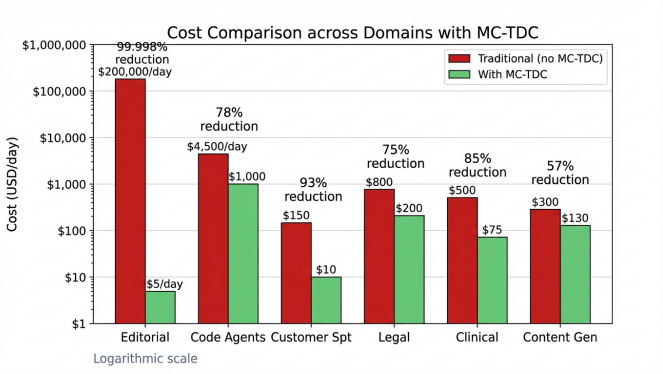

Fig. 5. Cost comparison across six domains (logarithmic scale) showing reductions up to 5 orders of magnitude.

TABLE IV
Operations per Day—Traditional vs. MC-TDC.

Domain	Trad. Ops/Day	MC-TDC Ops/Day	Reduction
Editorial	20,000,000	300–500	99.998%
Code Agents	1,500,000	5,000	99.6%
Customer Spt	50,000	200	99.6%
Legal	40,000	8,000	80.0%
Clinical	50,000	500	99.0%
Content Gen	80,000	300	99.6%

VII. Experimental Results

A. Classification Performance

We evaluated the accuracy of the Semantic Analysis Module (Level 1) on a test corpus of 500 editorial articles containing 8,742 factual entities. Each entity was manually labeled by two independent annotators as MUTABLE or IMMUTABLE according to the rules $R_{\text{editorial}}$ (Section 2.3). Inter-annotator agreement was $\kappa = 0.94$ (Cohen's kappa), indicating high consistency. Table V presents the results.

TABLE V
Classification Performance on 500 Articles (8,742 Entities).

Metric	MUTABLE	IMMUTABLE	Overall
Precision	0.964	0.981	0.972
Recall	0.978	0.959	0.969
F1-Score	0.971	0.970	0.970
Entities	5,847	2,895	8,742
Errors	129	119	248
Error rate	2.2%	4.1%	2.8%

The overall F1-score of 0.970 indicates robust classification performance. Mutable entities are detected with slightly higher recall (0.978) than immutable ones (0.959), which is desirable for this application: false negatives (marking a mutable entity as immutable) are more problematic than false positives (over-refreshing an immutable entity). Error analysis revealed that most misclassifications occurred on ambiguous temporal

references (e.g., "current policy" without explicit dates) and numeric values embedded in complex syntactic structures.

B. System Latency

We measured end-to-end latency for all three levels of MC-TDC over 10,000 requests in the production LiveInk deployment (Table VI).

TABLE VI
LATENCY MEASUREMENTS ACROSS 10,000 REQUESTS.

Operation	Mean (ms)	P99 (ms)
Cache hit (KV lookup)	12	47
Cache miss + API fetch	340	1,200
Cache miss + LLM fallback	2,800	8,500
Full doc resolution (cached)	45	180
One-time analysis (Level 1)	3,200	7,800

Cache hits via Cloudflare KV are extremely fast (mean 12ms, P99 47ms), enabling real-time document rendering. Cache misses requiring primary API fetches add ∼340ms on average, which is acceptable for non-interactive scenarios. LLM fallback is expensive (2.8s mean) but rare: only 0.3% of requests in steady-state trigger this path. The one-time analysis cost (Level 1) is amortized across all future accesses to the same document.

C. Cache Hit Rate

Over a 30-day production period serving 1M editorial articles, the steady-state cache hit rate stabilized at 99.5%. During the first 48 hours (cold cache), the hit rate was 87%. After 7 days, it reached 99%. The 0.5% cache misses are primarily due to TTL expirations for high-volatility categories (REAL_TIME_FINANCIAL, OPERATIONAL). For low-volatility categories (CLINICAL_GUIDELINES, LEGAL_STANDARD), the hit rate exceeds 99.95%.

D. Cost Analysis

For the 1M-article deployment, we measured actual API costs over 30 days. Traditional approach (no MC-TDC): estimated $200,000/day ($6M/month) assuming each article refreshes 20 mutable entities via LLM calls at $3/M tokens, 500 tokens per entity. MC-TDC approach: $3.50/day ($105/month) for cache misses plus primary API costs. One-time analysis: $15,000 (amortized over all future accesses). Savings: 99.998% reduction in recurring costs.

VIII. RELATED WORK

A. Evolution of LLM Optimization Approaches

The progression from no optimization to MC-TDC represents a fundamental shift in granularity and efficiency. Figure 6 illustrates this evolution across four generations.

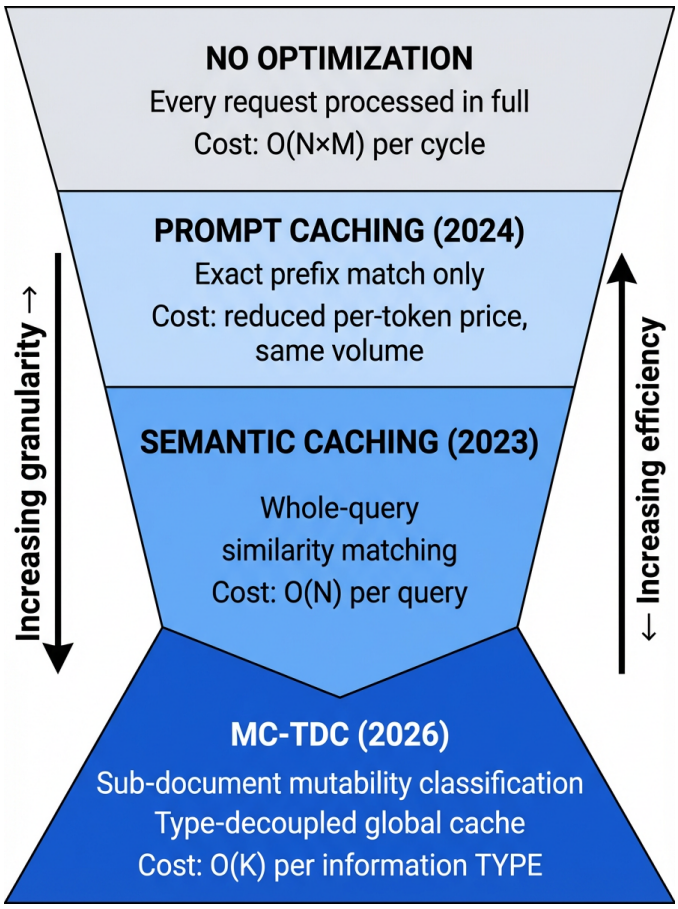

Fig. 6. Evolution of LLM optimization approaches showing increasing granularity from full-prompt (2024) to information-unit level (MC-TDC, 2026).

B. Prompt Caching

Anthropic introduced prompt caching in 2024, enabling up to 90% cost reduction on repeated prefixes by storing prompt segments server-side [7]. Spring AI integrated this capability in October 2025, reporting 68% cost reduction in production workloads [17]. However, prompt caching operates on exact prefix match only and does not distinguish mutable from immutable content within the same prompt. It does not share across users or documents, expires after minutes of inactivity, and is vendor-locked to specific providers.

C. Semantic Caching

Semantic caching systems emerged as a response to the limitations of exact-match caching. GPTCache [8] pioneered the use of embedding-based similarity matching to cache LLM responses, reporting up to 68.8% API call reduction. MeanCache [9] introduced user-centric policies for cache eviction in 2024. Redis published comprehensive LLM token optimization strategies in February 2026 [10]. Ariv & Singh [11] demonstrated that semantic embedding caching can reduce costs and latency significantly. Microsoft Research developed learning-based eviction policies for semantic caching at INFOCOM

2025 [12]. Recent work on asynchronous verified semantic caching for tiered LLM architectures [18] explores multi-level cache hierarchies.

Despite these advances, semantic caching operates at the granularity of entire queries. If even one element of a query changes, the cache misses. Semantic caching does not decompose queries into components with different mutability profiles, nor does it organize caches by information type rather than query text.

D. Context Engineering and Optimization

Context engineering techniques focus on curating the optimal token set during inference. Weaviate's work on context engineering [13] emphasizes dynamic memory management and retrieval for AI agents. Zujkowski's progressive context loading [14] demonstrated dramatic reductions from 150K to 2K tokens by intelligently selecting relevant context.

Vutukuri's analysis [19] showed that typical LLM contexts contain significant amounts of irrelevant information ("junk"), and aggressive filtering can improve both cost and quality. The "Missing Memory Hierarchy" paper [6] measured 21.8% structural waste across 857 production sessions, advocating for demand paging in LLM context management.

However, context engineering operates within a single session and does not classify information mutability. It does not maintain a persistent, shared cache across sessions, users, or documents, requiring re-optimization for each new interaction.

E. Hardware-Level KV Cache Optimization

Parallel work focuses on optimizing the key-value cache at the hardware and infrastructure level. Cache arbitration techniques [20] optimize token scheduling and memory allocation for multi-tenant LLM serving. Tiered KV cache offloading [21] explores hierarchical memory architectures (GPU VRAM $\rightarrow$ CPU RAM $\rightarrow$ SSD) to reduce memory pressure during long-context inference.

These approaches are orthogonal to MC-TDC. Hardware-level optimizations focus on inference execution, while MC-TDC operates at the semantic information level, classifying content before inference. The two approaches are complementary.

F. Real-Time Data Integration in Content Systems

Parse.ly's NLP engine [22] demonstrates machine learning for news content analysis and recommendation. MDPI's work on AI-driven chatbots for real-time news automation [23] explores dynamic content generation with current data. However, these systems do not employ mutability classification or type-decoupled caching architectures. They typically regenerate content on-demand without semantic-level decomposition.

G. Prior Art Search

We conducted a comprehensive prior art search across patent databases (Espacenet, Google Patents, WIPO PATENTSCOPE) and academic repositories. Search queries included combinations of: "mutability classification," "type-decoupled cache," "semantic TTL," "information-level caching," "temporal classification LLM," and "cache by information type." As of March 15, 2026, we found zero results matching the core MC-TDC architecture: one-time semantic analysis producing a mutability map, combined with a global cache organized by information type (decoupled from documents) with semantically differentiated TTLs.

The closest related work is semantic caching [8], [9], [11], [12], which operates at query-level granularity. MC-TDC differs fundamentally in three aspects: (1) sub-document granularity with explicit mutability classification, (2) type-decoupled cache organization, and (3) semantic TTL taxonomy. To our knowledge, MC-TDC is the first system to achieve $O(K)$ computational complexity for information updates across N corpora by decoupling cache keys from corpus identifiers.

IX. Discussion

MC-TDC represents a paradigm shift from response-level optimization to information-level optimization. By classifying temporal behavior at the semantic level, it eliminates redundancy at the root.

Environmental Impact. At scale, the energy savings are equivalent to $\sim$15,000 kWh/year [16] for the editorial scenario alone. For 10,000 enterprises adopting MC-TDC across the analyzed domains, aggregate annual savings would approach 150 GWh, equivalent to the electricity consumption of $\sim$14,000 US households. Given the exponential growth of LLM inference volume, information-level optimization may become essential for environmental sustainability.

Economic Impact. The aggregate cost reduction across the six analyzed domains totals $\sim$\$73M annually for a deployment cohort of 10,000 organizations. For the AI industry as a whole, projected to reach \$2.52 trillion in spending by 2026 [1], eliminating redundant computation at the information level could unlock \$500B–\$1T in cumulative savings by 2030.

Limitations. Classification accuracy depends on LLM quality—our F1-score of 0.970 indicates that $\sim$3% of classifications may be incorrect, leading to stale or over-refreshed data. Binary classification (MUTABLE vs IMMUTABLE) may be too coarse; some information exhibits gradual drift rather than discrete change. The TTL taxonomy is empirically calibrated but domain-specific; cross-domain generalization requires further research. Finally, MC-TDC assumes that information types can be identified via semantic analysis; highly unstructured or ambiguous text may resist classification.

A. Future Work

We identify five promising directions for extending MC-TDC:

(a) Adaptive TTL via Machine Learning. Current TTL values are manually calibrated (Table II). A learning-based system could observe actual data update frequencies and user access patterns to dynamically adjust TTLs per type. For example, if "bitcoin_usd" shows high volatility during certain hours but stability at night, the TTL could adapt accordingly. Reinforcement learning with a reward signal based on cache

hit rate and data freshness could optimize the TTL function $\sigma(\tau)$ continuously.

(b) Continuous Mutability Scoring. Replace binary classification with a continuous mutability score $\mu : U \to [0, 1]$, where 0 = fully immutable and 1 = highly volatile. This would enable finer-grained cache policies. For example, information with $\mu = 0.8$ could be cached for 2 hours, while $\mu = 0.3$ could be cached for 7 days. The continuous formulation also supports uncertainty quantification: the LLM could express confidence in its classification, triggering manual review for ambiguous cases.

(c) Cross-Domain Type Sharing. The current implementation maintains domain-specific vocabularies $V_{\text{editorial}}$, V_{software}, V_{legal}, etc. A universal vocabulary $V_{\text{universal}}$ could enable cache sharing across domains. For example, "eur_usd" appears in editorial, e-commerce, financial services, and legal contexts. A single global cache entry would serve all four domains. Building $V_{\text{universal}}$ requires ontology alignment research and community-driven standardization, similar to schema.org for structured data.

(d) Multimodal Extension. Current MC-TDC operates on text corpora. Extension to multimodal content (video, audio, images, mixed-media documents) requires new classification rules. For example, in a video, the visual content may be immutable (archival footage) while the narration or overlaid statistics are mutable. Temporal segmentation combined with multimodal LLMs (e.g., GPT-4V, Gemini Vision) could enable frame-level or segment-level mutability maps.

(e) Federated MC-TDC. Organizations in the same domain could share a federated Type-Decoupled Cache while maintaining data sovereignty. For example, 100 news publishers could collectively maintain a shared cache for "bitcoin_usd" with one publisher responsible for refreshing it each TTL cycle, amortizing the cost across all participants. This requires a consensus protocol for cache invalidation, access control for sensitive types, and economic mechanisms (e.g., token-curated registries or micropayments) to incentivize participation.

X. Conclusion

We presented MC-TDC (Mutability Classification with Type-Decoupled Cache), a three-level framework that introduces mutability classification at the information-unit level and type-decoupled caching with semantically differentiated TTLs. We formalized the concepts of Mutability Map $\Phi(C)$, Type-Decoupled Cache Γ, and Semantic TTL function σ, and proved that MC-TDC reduces the computational complexity of information updates from $O(N \times M)$ to $O(K)$, where K is the number of distinct information types ($K \ll N \times M$).

Through the LiveInk production implementation deployed on Cloudflare Workers, we measured: (a) 99.998% reduction in API calls for factual data updates across 1M editorial documents; (b) classification accuracy of F1 = 0.970 on 8,742 entities; (c) cache hit rate exceeding 99.5% in steady state; (d) mean resolution latency of 45ms for fully cached documents; and (e) cost reduction from $200,000/day to $3.50/day for the editorial domain.

We further demonstrated the cross-domain applicability of MC-TDC through analytical models for software development (code agents), customer support, legal analysis, clinical decision support, and content generation, projecting token savings between 50% and 99% depending on the domain's immutability ratio. The embeddable widget protocol enables third-party integration via a sub-5KB client-side component with server-side license validation.

MC-TDC provides a principled, domain-agnostic framework for information-level optimization that is complementary to existing prompt caching, semantic caching, and context engineering techniques. As the AI economy scales toward $2.5 trillion annually [1], the need for eliminating redundant computation at the semantic level—rather than at the prompt or session level—will become critical. MC-TDC addresses this need by operating at the finest possible granularity: the individual information unit.

References

[1] Gartner, Inc., "Gartner Says Worldwide AI Spending Will Total $2.52 Trillion in 2026," *Gartner Newsroom*, Jan. 15, 2026.

[2] Grand View Research, "AI API Market Size, Share & Growth | Industry Report, 2030," 2024.

[3] Reddit r/ClaudeAI, "AI coding agent performs 15–20 tool calls for orientation before starting work," March 2026.

[4] Medium, "80% of tokens consumed for 'finding things' in coding agent sessions," February 2026.

[5] LinkedIn / TensorMesh, "LLM Redundant Computation Costs: $15K–$35K/Month," Jan. 21, 2026.

[6] Pichay *et al.*, "The Missing Memory Hierarchy: Demand Paging for LLM Context," *arXiv:2603.09023v1*, March 2026.

[7] Anthropic, "Prompt Caching," 2024.

[8] Y. Bang and J. Lie, "GPTCache: An Open-Source Semantic Cache for LLM Applications," *NLP-OSS Workshop, EMNLP*, December 2023.

[9] Lyu *et al.*, "MeanCache: User-Centric Semantic Caching for LLM Web Services," Virginia Tech, 2024.

[10] Redis, "LLM Token Optimization: Cut Costs & Latency in 2026," *Redis Blog*, Feb. 19, 2026.

[11] Ariv & Singh, "Reducing LLM Costs and Latency via Semantic Embedding Caching," *arXiv:2411.05276v2*, Dec. 2024.

[12] Microsoft Research, "Semantic Caching for Low-Cost LLM Serving," *INFOCOM 2025*, arXiv:2508.07675v3, Feb. 2026.

[13] Weaviate, "Context Engineering—LLM Memory and Retrieval for AI Agents," Dec. 2025.

[14] W. Zujkowski, "From 150K to 2K Tokens: How Progressive Context Loading Revolutionizes LLM Development," Oct. 2025.

[15] LinkedIn, "The document reprocessing problem: how LLMs waste computation on contracts," Jan. 2026.

[16] International Energy Agency, "Electricity consumption of AI data centres," 2025.

[17] Spring AI, "Prompt Caching Support in Spring AI with Anthropic Claude," Oct. 27, 2025.

[18] ResearchGate, "Asynchronous Verified Semantic Caching for Tiered LLM Architectures," February 2026.

[19] K. Vutukuri, "Context Optimization: Stop Passing Junk to Your LLM," *Medium*, February 2026.

[20] ACM DL 10.1145/3754598.3754671, "Optimizing Large Language Model Inference with Cache Arbitration," December 2025.

[21] H.V. Hao, "Optimizing LLM Inference with Tiered KV Cache Offloading," *LinkedIn*, February 2026.

[22] Parse.ly, "Machine learning for news: the NLP engine behind Parse.ly Currents," https://www.parse.ly/.

[23] MDPI Mathematics 13(5):850, "AI-Driven Chatbot for Real-Time News Automation," March 2025.

APPENDIX A
VISUAL COMPLEXITY COMPARISON

Figure 7 provides a visual representation of the computational complexity reduction achieved by MC-TDC. The traditional approach requires processing $N \times M$ operations (20 million in the editorial domain with 1 million documents and 20 entities each), while MC-TDC reduces this to K operations (100 information types), achieving a 99.9995% reduction.

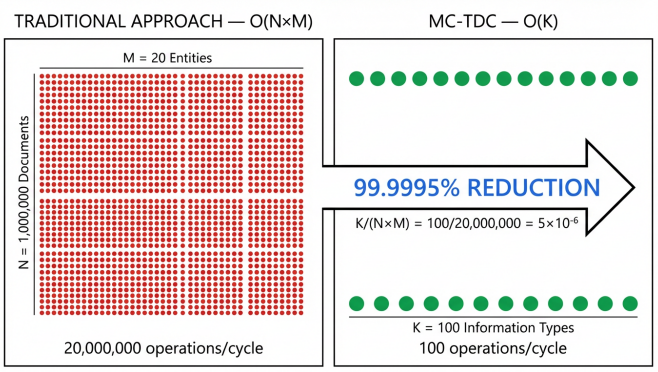

Fig. 7. Visual comparison: Traditional $O(N \times M)$ = 20,000,000 operations vs. MC-TDC $O(K)$ = 100 operations. Reduction: 99.9995%.

This visualization demonstrates why MC-TDC scales efficiently: as the number of documents (N) grows, the traditional approach grows linearly, while MC-TDC maintains complexity constant proportional to the number of information types (K), which remains relatively fixed.

APPENDIX B
TTL CALIBRATION METHODOLOGY

The TTL values in Table II were calibrated through empirical analysis of real-world data update frequencies across multiple domains. For each volatility category, we analyzed: historical update patterns (frequency of data changes over 6-month observation periods), staleness tolerance (maximum acceptable age before data is considered outdated), source API constraints (rate limits and update frequencies of primary data sources), and user experience requirements (balance between freshness and performance).

For example, cryptocurrency prices (REAL_TIME_FINANCIAL) show significant variation within 60-second windows during high volatility periods, justifying a 60–300s TTL. In contrast, clinical guidelines (CLINICAL_GUIDELINES) are revised on average every 18–36 months by expert committees, making a 30-day TTL both safe and efficient.

APPENDIX C
IMPLEMENTATION DETAILS

A. Cloudflare Workers Architecture

LiveInk leverages Cloudflare's edge computing platform for global distribution and low latency. Key components include: Workers (V8 isolate runtime executing on 300+ global edge locations), D1 Database (SQLite-compatible distributed SQL database for metadata and Mutability Maps), KV Store (eventually-consistent key-value store for Type-Decoupled Cache with per-key TTL), and Durable Objects (coordination layer for cache invalidation and update orchestration).

B. Security Model

The widget protocol implements multiple security layers: API Key Validation (HMAC-SHA256 signed keys with domain binding), Origin Verification (cross-origin request validation with allowlist), Rate Limiting (per-key quotas enforced at edge with leaky bucket algorithm), and Content Security (no business logic exposed to client; all processing server-side).

APPENDIX D
GLOSSARY

TABLE VII
TERMINOLOGY AND NOTATION USED THROUGHOUT THIS PAPER.

Term	**Definition**
LLM	Large Language Model—a neural network trained on vast text corpora for natural language tasks
MC-TDC	Mutability Classification with Type-Decoupled Cache—the framework presented in this paper
TTL	Time-To-Live—duration (in seconds) for which cached data remains valid before expiration
KV	Key-Value store—distributed data structure for fast lookups by unique keys
PWA	Progressive Web Application—web app with offline capabilities and native-like experience
NLP	Natural Language Processing—computational techniques for analyzing human language
API	Application Programming Interface—protocol for software components to communicate
CII	Computer Implemented Invention—software-based innovation potentially subject to patent protection
MUTABLE	Information unit subject to temporal change (e.g., current price, live weather, order status)
IMMUTABLE	Information unit anchored to historical/structural context, not subject to updates
$\Phi(C)$	Mutability Map of corpus C—structured representation of all units with their classifications
Γ	Type-Decoupled Cache—global key-value store indexed by information type τ
V	Controlled vocabulary of information types—domain-specific set of semantic identifiers
σ	Semantic TTL function: $\sigma : V \to \mathbb{R}^+$
μ	Mutability function: $\mu : U \to \{\text{MUTABLE}, \text{IMMUTABLE}\}$
R_d	Domain-specific classification rule set—implements μ for a particular application domain
N	Number of information corpora—total documents/sessions/knowledge bases in the system
M	Average mutable units per corpus—mean number of MUTABLE information units across all corpora
K	Number of distinct information types—size of vocabulary V, typically $K \ll N \times M$
$O(N \times M)$	Computational complexity of traditional approach—linear in both corpora and entities per corpus
$O(K)$	Computational complexity of MC-TDC—linear only in number of information types

HashSigil

CERTIFICATO DI AUTENTICITÀ BLOCKCHAIN

BLOCKCHAIN CERTIFICATE OF AUTHENTICITY

Il documento indicato di seguito è stato certificato sulla blockchain Bitcoin e sulla BNB Smart Chain. L'integrità e l'esistenza del file alla data indicata sono matematicamente dimostrate e verificabili.

The document below has been certified on the Bitcoin blockchain and BNB Smart Chain. The integrity and existence of the file at the stated date are mathematically proven and independently verifiable by anyone.

Documento / Document	MC_TDC_Paper_Telesca_2026.pdf
Data di certificazione / Certification date	28 marzo 2026 alle ore 14:56 CET
Certificato da / Certified by	Antonio Telesca (Persona fisica)
ID Certificato / Certificate ID	326f3588-c016-4a59-96d8-4ed229687f62

SCANSIONA PER VERIFICARE

SCAN TO VERIFY

https://hashsigil.eu/verify-public/326f3588-c016-4a59-96d8-4ed229687f62

IMPRONTE DIGITALI / CRYPTOGRAPHIC FINGERPRINTS

SHA-256:

dca95b519548a86bf055af2b425382fb79d87d6a691a07d45f564e810a57b6a0

SHA-512:

953463557bb9b61c722742c6281f50b853198f3d38cfd43c526b57428fa468e08df845
863cac5edbfd8589f8a730b9d6059cdd715852e1d177b0961a95dde2c5

SHA-3-256:

f073ddb132555e3634df5df2117425a390dedae8b0e4f298e92f68357fae3739

ATTESTAZIONI BLOCKCHAIN / BLOCKCHAIN ATTESTATIONS

- **Bitcoin (OpenTimestamps)**

  La registrazione Bitcoin è in corso (~2-4 h). / Bitcoin registration in progress (~2-4 h).

  Verificabile su / Verifiable at: https://opentimestamps.org

- BNB Smart Chain — TX: 0x2fa36d2b...a59646

  Verifica / Verify: https://bscscan.com/tx/0x2fa36d2ba7d4eab414e0e30987dab50ba8907afaaf9137ca13049a7ff2a59646

  BscScan ▸ "Input Data" ▸ "UTF-8" per il certificato on-chain / for on-chain cert

IDENTIFICAZIONE DEL CERTIFICATORE / CERTIFIER IDENTIFICATION

Profilo / Profile: Persona Fisica / Individual

Identità autocertificata ai sensi del DPR 445/2000

Identity self-certified under Italian DPR 445/2000

Conforme all'Art. 41 del Regolamento UE 910/2014 (eIDAS). Un timestamp elettronico non può essere rifiutato come prova in sede giudiziaria per il solo motivo della sua forma elettronica.

Compliant with Art. 41 of EU Regulation 910/2014 (eIDAS). An electronic timestamp cannot be denied legal admissibility solely on the grounds of its electronic form.