Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# OntoCacheRAG: Ontology-Driven Selective Cache Invalidation for Knowledge-Graph-Augmented Retrieval Systems

**Nimas Ayu Untariyati**

nayuuntariyati@students.undip.ac.id

Diponegoro University

**Kusworo Adi**

Diponegoro University

**Aris Puji Widodo**

Diponegoro University

**M. Teduh Uliniansyah**

National Research and Innovation Agency

**Research Article**

**Keywords:** Knowledge Graph, Retrieval-Augmented Generation, Cache Invalidation, OWL Ontology, SPARQL, Semantic Entailment

**Posted Date:** September 22nd, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-9879191/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

**Additional Declarations:** No competing interests reported.

# Abstract

Retrieval-Augmented Generation (RAG) systems that incorporate Knowledge Graphs (KGs) face a critical challenge: when the underlying ontology changes, cached query results become semantically stale, yet existing invalidation strategies offer only a binary choice between correctness and efficiency. We present OntoCacheRAG, a three-component pipeline that resolves this trade-off through subsumption-aware selective invalidation. The pipeline comprises an Ontology Change Detector (OCD) that classifies changes via a formal Ontology Change Impact Classifier (OCIC) taxonomy, a Semantic Impact Mapper (SIM) that computes transitive impact sets through OWL 2 RL/RDFS entailment reasoning over SPARQL property paths, and a Selective Cache Invalidator (SCI) that performs fine-grained KV cache invalidation in Lazy, Adaptive, or Eager mode based on change severity.

We formalize the cache-invalidation problem under ontology evolution and prove that the SIM component is complete with respect to the modelled structural impact definition. We evaluate on the ArsipDataset Indonesian regulatory corpus (614 documents; 42 verified revocation relations; 870 cache entries) under two ontology-change classes — instance revocation (ABox) and class-hierarchy restructuring (TBox) — across 20 random seeds. Under ABox revocation, OntoCacheRAG achieves perfect detection (F1 = 1.000) while String-Match misses 45% of stale entries (recall 0.546); under TBox restructuring, String-Match detects none (recall 0.000) while single-hop traversal captures only half (recall 0.504). All differences are statistically significant (Wilcoxon signed-rank, $p < 0.001$). Full-cache flushing achieves correctness only by evicting 85–94% of valid entries. Scalability experiments on synthetic ontologies (200 to 50,000 classes) demonstrate sub-linear growth in SIM latency with an empirical exponent of $O(|C|^{0.31})$. Latency profiling reveals that 99% of OntoCacheRAG overhead originates in the SIM SPARQL reasoning step, which remains in the single-digit-millisecond range per change event. To our knowledge, OntoCacheRAG is the first framework to integrate subsumption-aware ontology reasoning into the cache-invalidation lifecycle of a KG-augmented retrieval system.


# 1. Introduction

Large Language Models (LLMs) are increasingly deployed as reasoning components over structured knowledge sources, with Knowledge-Graph-Augmented RAG [1, 2, 3] representing one of the most promising architectures for factual question answering. KG-RAG systems cache intermediate computations — particularly key-value (KV) attention tensors [4, 5] — to reduce the per-query inference cost of encoding retrieved graph context. This caching is effective when the underlying knowledge graph remains static, but real-world KGs are dynamic: ontologies evolve as regulations change, taxonomies are revised, and entities are reclassified [6, 7, 8].

There is a common limitation for the existing cache-management strategies for KG-RAG. The full-flush invalidation purges the entire cache for ontology updates, which guarantees freshness, albeit wasting all previous work. No-invalidation provides near-zero invalidation overhead, but produces stale, semantically incorrect responses. TTL-based strategies apply uniform expiration that do not consider the semantic effects of a particular change [9, 10]. The fundamental issue is an impedance mismatch, where cache

invalidation is thought of as a data freshness problem. But, changes in ontology propagate through class hierarchies and entailment chains [11] in ways that cannot be captured by timestamping or string-matching. A change to a superclass axiom in an OWL TBox can make all cached responses stale that involve any subclass. However, no-invalidation and string-match systems fail to capture this.

This paper presents OntoCacheRAG, which addresses this mismatch directly. Building on our prior work on topology-aware KG caching [12], we extend the cache lifecycle with ontology-aware invalidation. Our contributions are:

- OCIC Taxonomy: a structured classification of ontology changes by semantic scope (TBox vs. ABox) and severity, grounded in the taxonomies of Noy and Klein [6] and Flouris et al. [7], that enables change-proportional invalidation decisions. We provide a formal mapping between our taxonomy and prior ontology change classifications.
- Semantic Impact Mapper (SIM): a reasoning module for subsumption which identifies the transitive impact set of a change using SPARQL property paths, resulting in exact impact sets of cache keys. We define this impact set and show that SIM computes it accurately for the change classes which it supports.
- Three-mode SCI: a cache invalidator which utilizes Lazy, Eager, or Adaptive (gravity-of-impact) invalidation, accompanied by a formal complexity analysis and empirical latency assessment.
- Thorough empirical assessment: a nine-system comparative benchmark using a real regulatory framework, with per-change-type latency measurement, cache efficiency, scalability tests with synthetic ontologies (200 to 50,000 classes), and ablation studies. Code and the modified ontology are provided for replication purposes.

The rest of this paper is structured as follows: Section 2 discusses related works. Section 3 formally presents the problem. Section 4 presents the OntoCacheRAG system. The experimental procedures are described in Section 5. Results and discussion are provided in Section 6. Section 7 discusses limitations. Section 8 presents the concluding remarks and potential directions for the future.

## 2. Related Work

# 2.1 Knowledge-Graph-Augmented RAG

Retrieval-Augmented Generation is based on Lewis et al. [1], who try to bind LLM outputs to documents in the retrieval set. Pre-training the retriever alongside the language model is done via REALM [14]; Fusion-in-Decoder [15] looks at multiple retrieved passages in the decoder. Dense passage retrieval [16] is the typical retrieval method used in RAG systems. KG-RAG developments expand the document collection with structured knowledge graphs [2, 3, 17] to allow relational multi-hop reasoning. For global sensemaking, GraphRAG [18] incorporates LLM-generated knowledge graphs along with community-based hierarchical summaries. Think-on-Graph [19] shows LLMs do extensive and responsible reasoning on knowledge graphs. StructGPT [20] is one of many KG Question Answering systems showing that to improve the

accuracy of complex questions, the graph-structured context is useful. These systems do not address cache consistency with evolving ontologies, and treat the KG as a static retrieval resource.

Recent work has explored the intersection of knowledge graphs and RAG quality. KG-Tune [21] demonstrates that knowledge graph-driven fine-tuning of embedding models improves retrieval quality in RAG through a two-phase training approach. This supports the argument that structured knowledge is essential for high-quality retrieval — and consequently, that the integrity of cached retrieval results is critical when the underlying knowledge structure changes. RDPG [22] proposes adaptive path generation for multi-hop knowledge graph question answering, showing that dynamic graph reasoning paths are sensitive to the graph structure. Ontology changes that alter the graph structure can therefore disrupt these reasoning paths, motivating the need for cache-aware invalidation in graph-based retrieval pipelines. Haque et al. [23] further demonstrate that LLM-constructed KGs can be evaluated via structural quality metrics, reinforcing the importance of ontology-level integrity in KG systems.

Most recently, KAIS has hosted significant work on KG-based RAG. Wang et al. [50] proposed LLM-KGMQA, an LLM-augmented multi-hop QA framework that introduces an n-hop subgraph construction algorithm, a knowledge fusion algorithm, and a semantics-based pruning algorithm tailored to medical knowledge graphs. Lu et al. [49] proposed MI-RAG, a mutual information-based generate-then-retrieve framework that uses subgraph branch pruning and an explicit KG update mechanism. Both works confirm that subgraph-level retrieval and pruning are central operations in KG-RAG pipelines. OntoCacheRAG complements these approaches by addressing the orthogonal problem of cache coherence when the underlying ontology itself evolves — a dimension neither MI-RAG's per-query KG update nor LLM-KGMQA's static medical KG addresses.

## 2.2 KV Cache Management for LLMs

Several approaches to compression, like Scissorhands [27] and H2O [28] along with other adaptive methods [29], look at the size of the cache memory, but do not account for when the knowledge changes and remains semantically valid. There is also PromptCache [30] and CacheGen [31], which promote better cache utilization when used over multiple prompts and sessions. Compression approaches including Scissorhands [27], H2O [28], and adaptive methods [29] reduce cache memory footprint but do not address semantic staleness when the underlying knowledge changes. PromptCache [30] and CacheGen [31] optimize cache reuse across prompts and sessions.

The first version of our work, SKV-Cache [12], implemented topology-based KG caching to answer questions via archival means. We achieved major latency reductions by clustering cache entries based on their relationship in the knowledge graph. However, SKV-Cache, like the rest of these solutions, works with static knowledge bases. It also does not account for changes to the ontology and implications of reasoning when the underlying framework changes. Building on the work of SKV-Cache and others, OntoCacheRAG provides the gap in KG Cache lifecycle management by incorporating an exit strategy based on changing ontologies.

## 2.3 Ontology Change Management

Noy and Klein [6] established that ontology evolution differs fundamentally from database schema evolution, requiring attention to semantic impact propagation. Their taxonomy of change operations and impact categories provides the conceptual foundation for our OCIC classifier. Flouris et al. [7] provided a comprehensive classification and survey of ontology change operations, distinguishing between changes that affect terminological knowledge (TBox) and assertional knowledge (ABox). Zablith et al. [8] offered a process-centric survey covering the full lifecycle of ontology evolution, from change detection through impact analysis to change application.

Stojanovic [32] developed methods and tools for managing ontology evolution in knowledge-management systems, introducing the notion of change-resolution strategies. Noy et al. [33] formalized change-tracking logs for collaborative ontology evolution, providing the foundation for our OCD component. Haase et al. [34] proposed a framework for handling inconsistency in changing ontologies, addressing the problem of maintaining logical coherence after updates. Flouris et al. [35] established formal foundations for RDF/S knowledge base evolution using belief revision operators, and Papavassiliou et al. [36] developed high-level change detection methods for RDF(S) knowledge bases.

While this body of work provides sophisticated theoretical frameworks for understanding and classifying ontology changes, no prior work has operationalized these theories as a runtime invalidation component within a retrieval system's cache lifecycle. OntoCacheRAG bridges this gap, transforming ontology change management from a knowledge engineering concern into a systems-level cache coherence mechanism.

In the KG-RAG context specifically, Lu et al. [49] equip their pipeline with a KG update mechanism that supplements or corrects the graph after each QA interaction. While this addresses ABox-level fact updates, it does not address the cache coherence implications of such updates — exactly the gap OntoCacheRAG fills.

## 2.4 Temporal and Dynamic Knowledge in Retrieval Systems

TempRAA [37] addresses temporal relation-aware alignment for time-sensitive knowledge graph question answering, introducing self-feedback question rewriting and temporal-aware retrieval mechanisms. This work illustrates a broader principle: that retrieval must be context-sensitive rather than relying solely on static semantic similarity. OntoCacheRAG applies this principle to a different dimension of knowledge dynamism — structural ontological change rather than temporal fact change — but shares the fundamental insight that retrieval systems must account for the evolving nature of knowledge.

## 2.5 Cache Invalidation in Distributed Systems

Classic cache-invalidation literature distinguishes time-based (TTL), event-based, and dependency-tracking strategies [9, 10]. Dependency-tracking approaches — associating a cache entry with the data it depends on — most closely resemble ours. Dynamo [38] and Spanner [39] demonstrate that even globally distributed caches require principled invalidation. However, prior work operates at the relational or document level; OntoCacheRAG introduces ontological entailment as the dependency relation, enabling invalidation that is both more precise (no unnecessary eviction of unaffected entries) and more complete (no missed stale entries propagated through subsumption chains).

## 2.6 Research-Gap Summary

Table 1 positions OntoCacheRAG against prior KV cache systems. The unique contribution lies at the intersection of ontology-aware reasoning and cache lifecycle management — a gap not addressed by any individual prior system.

Table 1
Positioning of OntoCacheRAG against prior KV cache systems.

| Capability | Prior KV Cache Systems | OntoCacheRAG |
|---|---|---|
| Handles KG evolution | No | Yes |
| Ontology-driven invalidation | No | Yes (subsumption-aware) |
| Selective granularity | Block/chunk level | Ontology-class level |
| Per-change-type analysis | Not addressed | ABox + TBox evaluated |
| Correctness under evolving KB | Not addressed | Formally defined & empirically evaluated |
| Scalability analysis | Fixed ontology size | 200–50,000 classes tested |

# 3. Problem Formulation

## 3.1 Ontology and Cache Model

Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ denote a knowledge graph ontology expressed in an OWL 2 RL/RDFS-compatible fragment of Description Logics [11, 40], where $\mathcal{T}$ (TBox) contains terminological axioms (class definitions, subclass relationships, property domain and range restrictions) and $\mathcal{A}$ (ABox) contains assertional axioms (individual class memberships, property assertions between individuals).

A KG-RAG system maintains a cache $C = \{(k_i, v_i)\}$ in which each key $k_i$ encodes a query $q$ and retrieved context $G_q \subseteq \mathcal{O}$, and each value $v_i$ stores the precomputed KV attention tensors for that (query, context) pair. We define the *anchor set* of a cache entry as:

$$Anchor(k_i) = \{e \in Ind(\mathcal{A}) \mid e \, appears \, in \, G_{q_i}\}$$

That is, the anchor set is the set of individuals referenced in the cached context.

## 3.2 Ontology Update and Semantic Staleness

An ontology update $u = (\Delta^+, \Delta^-)$ asserts new axioms $\Delta^+$ and retracts existing axioms $\Delta^-$, yielding the updated ontology:

$$\mathcal{O}' = (\mathcal{O} \setminus \Delta^-) \cup \Delta^+$$

**Definition 1**

**(Semantic Staleness).** A cache entry $(k_i,\ v_i)$ is *stale* with respect to update $u$ if and only if the context $G_{q_i}$ used to compute $v_i$ is no longer faithful to $\mathcal{O}'$. Formally:

$$Stale\,(k_i,\ \mathcal{O},\ u) \iff \exists\,\alpha\ \in\ G_{q_i}\ :\ \mathcal{O}'\ \not\models\ \alpha$$

where $\not\models$ denotes that the axiom $\alpha$ is no longer entailed by the updated ontology.

# 3.3 The Cache Invalidation Problem

**Definition 2**

**(Minimal Invalidation Set).** The cache-invalidation problem is to identify the minimal set of entries to remove such that all remaining entries are valid:

$$C_{inv}\ =\ \{k_i\ \in\ C\ \mid\ \ Stale\,(k_i,\ \mathcal{O},\ u)\}$$

Computing $C_{inv}$ exactly requires reasoning over the full entailment closure of $\mathcal{O}$ and $\mathcal{O}'$, which is computationally expensive for expressive DL fragments. OntoCacheRAG approximates $C_{inv}$ by computing a *structural impact set* that is guaranteed to be a superset of $C_{inv}$ for the supported change classes (Proposition 1, Section 4.2).

# 3.4 Evaluation Metrics

We measure a system's performance on the cache-invalidation problem with five metrics. **Invalidation Precision** quantifies the fraction of invalidated entries that were actually stale:

$$P_{inv}\ =\ \frac{|\,C_{inv}\ \cap\ C_{stale}|}{|\,C_{inv}|}$$

**Invalidation Recall** quantifies the fraction of stale entries that were correctly invalidated:

$$R_{inv}\ =\ \frac{|\,C_{inv}\ \cap\ C_{stale}|}{|\,C_{stale}|}$$

**Unnecessary Eviction Rate** (UER) measures over-invalidation — the fraction of invalidated entries that were in fact valid:

$$UER\ =\ 1\ -\ P_{inv}$$

**Residual Stale Rate** (RSR) measures under-invalidation — the fraction of cache entries that remain stale after invalidation processing. **Invalidation Latency** $L_{inv}$ is the wall-clock time to process a batch of ontology change events through the full OCD→SIM→SCI pipeline.

# 4. OntoCacheRAG Architecture

OntoCacheRAG consists of three sequentially composed components: the Ontology Change Detector (OCD), the Semantic Impact Mapper (SIM), and the Selective Cache Invalidator (SCI). Each ontology change

event flows through the pipeline: OCD classifies the change, SIM computes the set of affected entities through subsumption reasoning, and SCI invalidates the corresponding cache entries.

The three-component pipeline (OCD → SIM → SCI) processes ontology change events from the SPARQL update endpoint, computes transitive impact sets through subsumption reasoning, and selectively invalidates affected KV cache entries. The SIM component dominates pipeline latency (99%), while OCD and SCI each contribute less than 0.1%. Dashed rectangles indicate system boundaries.

# 4.1 Ontology Change Detector (OCD)

The OCD subscribes to SPARQL Update events on the ontology store [13]. For each intercepted event (INSERT DATA, DELETE DATA, INSERT/DELETE WHERE operations), it extracts the modified axioms and forwards them to the OCIC classifier. This component draws on the change-tracking formalism of Noy et al. [33] and the inconsistency-handling framework of Haase et al. [34].

# 4.1.1 OCIC Taxonomy

The OCIC classifies changes along two dimensions: level (TBox vs. ABox) and severity (Low / Medium / High). Table 2 presents the taxonomy, which extends the classifications of Noy and Klein [6] and Flouris et al. [7] with severity ratings grounded in the expected size of the impact set.

Table 2
Ontology Change Impact Classifier (OCIC) taxonomy.

| Change Type | Level | Severity | Rationale / Expected Impact |
|---|---|---|---|
| Class addition | TBox | Low | Additive; no existing entailments affected |
| Property addition | TBox | Low | Additive; no existing triples affected |
| Instance reclassification | ABox | Medium | Affects relations involving this entity; impact proportional to entity connectivity |
| Instance revocation | ABox | Med–High | Affects revoked entity and all entities linked through resolved properties |
| Property value update | ABox | Low–Med | Affects only (entity, relation) pairs referencing the changed value |
| Subclass restructuring | TBox | Med–High | Propagates through rdfs:subClassOf* chain; impact scales with hierarchy depth |
| Property domain/range change | TBox | High | All triples using this property may become inconsistent |
| Class deletion | TBox | High | All instances of deleted class (transitively) become orphaned |

Severity is grounded in the expected cardinality of the impact set relative to the total instance population.

The OCIC outputs a ChangeEvent record: {type, level, severity, axioms, timestamp}. Latency profiling (Section 6.3) shows OCD contributes < 0.1% of total pipeline overhead.

# 4.1.2 Relationship to Prior Taxonomies

Our OCIC taxonomy can be mapped to the classification of Flouris et al. [7] as follows. Their basic change operations (add/remove axiom) correspond to our Low-severity additive changes; their composite changes (which affect multiple interdependent axioms) map to our Medium-High severity restructuring operations. The key extension we provide is the severity dimension, which operationalizes the theoretical change classification into a decision criterion for the SCI's invalidation mode selection. Where Flouris et al. classify changes by type, we additionally classify by expected impact scope, enabling adaptive invalidation strategies.

# 4.2 Semantic Impact Mapper (SIM)

The SIM receives ChangeEvent records and computes the transitive impact set through subsumption-aware reasoning. This is the component that distinguishes OntoCacheRAG from string-matching or rule-based approaches.

# 4.2.1 Formal Definition of Impact Set

**Definition 3**

**(Impact Set).** For a change event $u$ affecting class $c$ or individual $e$ in ontology $\mathcal{O}$, the structural impact set is defined as:

$$Impact\,(u,\,\mathcal{O})\;=\;\left\{e\;\in\;\;Ind\,(\mathcal{A})\;\mid\;\exists\,c'\;\;\in\;\;Changed\,(u)\;\;:\;\mathcal{O}\;\vdash\;type\,(e)\;\sqsubseteq\;c'\right\}$$

where $Changed\,(u)$ returns the set of classes directly modified by $u$, $type\,(e)$ returns the most specific class of individual $e$, and $\sqsubseteq$ denotes the subsumption relation (the reflexive transitive closure of $rdfs:subClassOf$).

For ABox changes involving relational properties (e.g., revocation), the impact set is extended to include entities reachable through the affected property:

$$Impact_{rel}\;=\;Impact\;\cup\;\left\{e'\;\mid\;\exists\,e\;\in\;\;Impact,\,\exists\,r\;:\;\left(e,\,r,\,e'\right)\;\in\;\mathcal{A}\;\vee\;\;\left(e'\,,\,r,\,e\right)\;\in\;\mathcal{A}\right\}$$

# 4.2.2 Reasoning Process

For each ChangeEvent, the SIM resolves the affected entities by traversing the subsumption hierarchy via SPARQL $rdfs:subClassOf*$ property paths [13, 40] and following resolved relational links (e.g., the revocation property $dicabutOleh$). The implementation uses rdflib with SPARQL property-path queries. The core SPARQL pattern for TBox restructuring is:

SELECT ?d WHERE { ?d rdf:type/rdfs:subClassOf* ex:AffectedClass }

This pattern traverses the full transitive closure of the subclass hierarchy, returning all individuals that are instances of $AffectedClass$ or any of its subclasses.

## 4.2.3 Completeness Guarantee

**Proposition 1**

**(SIM Completeness).** For any ontology change $u$ of type ABox-revocation or TBox-restructuring within a single-property subsumption hierarchy expressed in the OWL 2 RL/RDFS fragment, the SIM returns a set $S$ such that:

$$C_{inv} \subseteq \{k \in C \mid Anchor(k) \cap S \neq \varnothing\}$$

**Proof sketch.** The SPARQL property path $rdfs:subClassOf*$ computes the reflexive transitive closure of the subclass relation in the loaded ontology graph. By the semantics of RDFS entailment [41], if individual $e$ is asserted as $rdf:type\, c$ and $c\, rdfs:subClassOf*\ c'$, then $\mathcal{O} \vdash e : c'$. For ABox-revocation changes, the SIM additionally follows the resolved property link ( $dicabutOleh$) to capture both the revoked and revoking entities. The SIM therefore identifies every individual whose class membership is affected by the change, and the reverse index maps these individuals to all cache entries that reference them. □

**Boundary conditions.** This completeness guarantee holds under the following assumptions: (i) the ontology is expressed in the OWL 2 RL/RDFS fragment (no nominals, no complex role inclusions); (ii) impact propagation follows a single property chain ( $dicabutOleh$ for revocation); (iii) the cache index correctly maps all anchor entities. Multi-property interaction, OWL restrictions ( $\exists$, $\forall$ ), cardinality constraints, and property chains beyond the modelled relation are not covered. See Section 7 for a full discussion of these boundary conditions.

## 4.2.4 Complexity Analysis

The SIM's dominant operation is the SPARQL property-path traversal, which computes a transitive closure over the subclass graph. For an ontology with $|C|$ classes, $|I|$ individuals, and maximum hierarchy depth $d$, the complexity behavior decomposes as follows.

- TBox restructuring. The SPARQL query visits at most $|C|$ classes in the transitive closure and retrieves the associated individuals. Worst-case complexity is $O(|C| + |I|)$ for a single change event.
- ABox revocation. The query retrieves the revoked individual and follows one property link. Complexity is $O(1)$ per property assertion plus the reverse-index lookup.
- Practical performance. In practice, the rdflib in-memory graph keeps the class hierarchy in an adjacency list, and SPARQL property-path evaluation uses iterative deepening. For the ArsipDataset ontology ($|C| = 131$, $|I| = 614$), the full traversal completes in 2–3 ms. Scalability experiments (Section 6.6) characterize growth up to $|C| = 50{,}000$ classes.

# 4.3 Selective Cache Invalidator (SCI)

The SCI maps each (entity, relation) pair in the impact set to cache block IDs via a precomputed reverse index (O(1) lookup per pair), then applies one of three invalidation modes:

**Lazy mode.** Marks affected blocks as STALE without deleting them. On the next query access, a stale block triggers recomputation. This minimizes invalidation latency while preventing stale answers from being served, at the cost of first-access recomputation delay.

**Eager mode.** Immediately deletes affected blocks and triggers background recomputation. This ensures no subsequent query ever encounters a stale entry, at the cost of higher invalidation latency and temporary cache misses during recomputation.

**Adaptive mode.** Each change is routed to Lazy or Eager based on the OCIC severity score. Low severity changes are handled lazily, as the change is unlikely to have an impact on active queries. Eager mode is used for high severity changes, as the cost of serving critically incorrect answers is too high to delay. Medium severity changes are handled based on a configurable threshold (default: Eager). This approach builds on the topology-aware indexing of SKV-Cache [12] by employing ontology-driven invalidation triggers.

Latency profiling (Section 6.3) shows that SCI adds < 0.1% to the total pipeline overhead. This suggests the invalidation bottleneck is in the SIM reasoning step and is not in the cache manipulation.

Pruning strategies have been introduced within KG-RAG retrieval pipelines. For example, Lu et al. [49] approach branch pruning via Shannon entropy minimization and Wang et al. [50] use LLM reasoning to propose a semantics-based approach to pruning. Both strategies result in retrieval-time pruning to alleviate noise within the LLM input. The SCI is orthogonal to these approaches as pruning the cache is at the core of SCI rather than pruning the retrieval result and is triggered by ontology change events.

## 5. Experimental Setup

# 5.1 Dataset: ArsipDataset

We evaluate on ArsipDataset, an Indonesian regulatory archive corpus of 614 documents (557 active, 48 revoked, 9 unspecified) sourced from public government records (peraturan.go.id). The regulatory domain is representative of evolving legal ontologies where cache correctness carries direct legal consequence [42, 43, 44]. Revocation relations are extracted from the document status field (“Tidak Berlaku Dicabut Oleh: ...”) and resolved to in-corpus documents by matching regulation number, year, and issuing-agency tokens, yielding 42 verified revocation links; 6 revoking documents lie outside the corpus. The simulated cache workload comprises 870 query entries anchored to documents.

# 5.2 Ontology Construction

A five-level subsumption hierarchy was devised to mirror the administrative structure of ministries within Indonesia. The hierarchy is in line with knowledge graph construction methodology [45] and the statistics for each level of the ontology are reported in Table 3. The total number of classes is 131, and the total

number of properties, instances, and revoked links are 2, 614, and 42 respectively. The number of triples is 1,901.

Table 3
Ontology statistics per hierarchy level. Total: 131 classes, 2 properties, 614 instances, 42 verified revocation links, 1,901 triples

| Hierarchy Level | Classes | Instances | Example |
|---|---|---|---|
| L1: Regulasi (root) | 1 | — | Regulasi |
| L2: PeraturanKementerian | 1 | — | PeraturanKementerian |
| L3: Coordination cluster | 5 | — | Permen_Polhukam, Permen_Perekonomian, ... |
| L4: Agency (instansi) | 124 | — | Permen_Polhukam_Pertahanan, ... |
| L5: Document instances | — | 614 | D0136 (Permen Pertahanan No.24/2019) |

# 5.3 Synthetic Ontology for Scalability Testing

To assess scalability outside of the ArsipDataset, we create synthetic ontologies with controlled parameters. These include a variable number of classes |C| ∈, where $|C| \in$ {200, 500, 1000, 2000, 5000, 10000, 20000, 50000}, a fixed depth of d = 5, a branching factor of $|C|/d$, and an instance density of 3 per leaf class. Synthetic ontologies include randomly generated revocation relations at ~ 7% density. This is roughly equal to the density in the ArsipDataset of 42/614. Cache workloads expand in direct proportion to the instance count.

# 5.4 Protocol and Baselines

We evaluate two ontology-change classes: instance revocation (ABox) and cluster restructuring (TBox). Nine systems are compared:

- Three OntoCacheRAG modes: Lazy, Eager, Adaptive.
- Four baselines: Full-Flush (discard entire cache), No-Invalidation (never invalidate), TTL-short (30% expiry), TTL-long (12% expiry).
- Two ablations: String-Match (lexical URI matching only), ABox-1hop (direct type matching without transitive closure).

Detection and efficiency metrics are reported as mean ± standard deviation across 20 random seeds. Statistical significance is assessed using the Wilcoxon signed-rank test [46] following the methodology of Demšar [47]. Latency is measured separately over 200 repetitions with 20 warm-up iterations. Source code and the resolved ontology are released for reproducibility.

## 6. Results and Discussion

# 6.1 Detection Accuracy: ABox Revocation

Table 4
Detection and efficiency metrics under ABox revocation (mean over 20 seeds).

| System | Precision | Recall | F1 | Stale | Unnecessary Eviction |
|---|---|---|---|---|---|
| OntoCacheRAG | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 |
| Full-Flush | 0.092 | 1.000 | 0.169 | 0.000 | 0.908 |
| No-Invalidation | 1.000 | 0.000 | 0.000 | 0.092 | 0.000 |
| TTL-short | 0.088 | 0.286 | 0.133 | 0.066 | 0.255 |
| TTL-long | 0.086 | 0.112 | 0.096 | 0.082 | 0.101 |
| String-Match | 1.000 | 0.546 | 0.705 | 0.042 | 0.000 |

Under instance revocation, String-Match achieves recall 0.546 ± 0.055: it detects directly revoked documents but misses the revoking documents whose status is transitively affected through the resolved dicabutOleh property. OntoCacheRAG captures both directions — revoked and revoking entities — through its bidirectional property traversal. Full-Flush achieves perfect recall but evicts 90.8% of valid entries, a cost that renders it impractical for production systems where cache rebuilding involves expensive LLM recomputation [24].

# 6.2 Detection Accuracy: TBox Restructuring

Table 5 reports the corresponding metrics under TBox class-hierarchy restructuring, with the addition of the ABox-1hop ablation which is meaningful only for TBox changes.

Table 5
Detection and efficiency metrics under TBox restructuring (mean over 20 seeds).

| System | Precision | Recall | F1 | Stale | Unnecessary Eviction |
|---|---|---|---|---|---|
| OntoCacheRAG | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 |
| Full-Flush | 0.059 | 1.000 | 0.112 | 0.000 | 0.941 |
| No-Invalidation | 1.000 | 0.000 | 0.000 | 0.059 | 0.000 |
| TTL-short | 0.060 | 0.304 | 0.100 | 0.041 | 0.255 |
| TTL-long | 0.065 | 0.128 | 0.085 | 0.051 | 0.101 |
| String-Match | 1.000 | 0.000 | 0.000 | 0.059 | 0.000 |
| ABox-1hop | 1.000 | 0.504 | 0.654 | 0.029 | 0.000 |

Under class-hierarchy restructuring, the limitations of lexical and shallow approaches become stark. String-Match achieves recall 0.000: it is architecturally blind to class restructuring because no document URI appears in the change event. ABox-1hop captures roughly half (recall 0.504 ± 0.168) by detecting one hop into the subclass hierarchy but missing deeper transitive impacts. Only subsumption-aware reasoning (OntoCacheRAG) traverses the full rdfs:subClassOf* chain [13, 40] to detect all affected instances.

Bars show mean Precision (slate blue), Recall (terracotta), and F1 (sage green) over 20 random seeds. OntoCacheRAG (highlighted column, cream background) is the only system achieving perfect detection (F1 = 1.00) in both scenarios. Cases of complete detection failure (recall = 0) are marked with a small "0". String-Match shows architectural blindness to TBox changes (recall = 0), while ABox-1hop achieves only partial recall due to single-hop limitation

Comparing across change types: lexical approaches lose 45% of stale entries under ABox changes and 100% under TBox changes. This supports the central claim that ontology-aware reasoning [6, 7] is not merely an optimization but a correctness requirement, and that its necessity scales with the structural depth of the change.

# 6.3 Latency Breakdown by Component

Table 6
Latency breakdown per component (median over 200 repetitions).

| Component | Median Latency | Share of Total |
|---|---|---|
| OCD (event classification) | 0.002 ms | 0.1% |
| SIM (SPARQL subsumption) | 2.593 ms | 99.0% |
| SCI (reverse-index lookup) | 0.003 ms | 0.1% |
| End-to-end (OCD + SIM+SCI) | 2.619 ms | 100% |

Cross-environment range: 3–10 ms end-to-end.

The bottleneck is unambiguous: SPARQL subsumption reasoning in the SIM accounts for 99.0% of total invalidation latency. OCD and SCI together contribute less than 0.2%. This has a direct implication for optimization: future work on reducing invalidation overhead — including incremental reasoning via ELK [48], materialized views, and cached entailment results — should focus exclusively on the SIM step. Despite dominating the pipeline, the absolute SIM cost remains in the single-digit-millisecond range, negligible relative to the LLM KV recomputation it avoids [24].

We note that absolute latency is hardware-dependent: across four measurement environments we observed end-to-end SIM latencies ranging from 3 to 10 ms. We therefore treat the qualitative finding — single-digit-millisecond overhead, dominated by SPARQL reasoning — as the reproducible result, not any single absolute figure.

# 6.4 Cache Workload and Efficiency Analysis

Table 7 reports the aggregate cache state metrics across all systems, including the fraction of cache affected by each change type and the hit rate after invalidation processing.

Table 7
Cache workload and hit-rate analysis (mean over 20 seeds).

| Metric | ABox Revocation | TBox Restructuring | Unit |
|---|---|---|---|
| Entries affected per event | 80.3 ± 14.8 | 51.5 ± 13.2 | entries |
| % cache affected | 9.2% | 5.9% | |
| Hit rate before change | 100% | 100% | |
| Hit rate after (OntoCacheRAG) | 90.8% | 94.1% | |
| Hit rate after (Full-Flush) | 0.0% | 0.0% | |
| Hit rate after (No-Invalidation) | 100% (15.4% stale) | 100% (5.9% stale) | |
| Hit rate after (String-Match) | 95.0% | 100% (5.9% stale) | |

While Table 7 reports aggregate hit rates, it does not distinguish between hits that return correct cached answers and hits that silently return semantically stale answers. Figure 4 decomposes the post-invalidation cache state into three categories — correct hits, stale hits (representing silent failure), and evicted entries — making the differing failure modes of each baseline visually explicit.

All values are mean percentages of total cache over 20 random seeds. (a) ABox revocation with baseline stale fraction 9.2%. (b) TBox restructuring with baseline stale fraction 5.9%. OntoCacheRAG (highlighted row, cream background) is the only system with zero stale hits in both scenarios. Stale percentages for each system are also reported as red labels at the right edge of each bar

OntoCacheRAG invalidates only the structurally affected fraction: 9.2% of cache for ABox revocation and 5.9% for TBox restructuring, preserving 91–94% cache hit rate. Full-Flush destroys the entire cache — analogous to the global invalidation patterns observed in distributed systems [9, 38]. String-Match in the TBox scenario maintains 100% hit rate because it detects nothing: all 5.9% stale entries remain active and are silently served to users. This demonstrates that OntoCacheRAG occupies the favorable position: selective enough to preserve most cached computation, complete enough to eliminate all detected staleness.

# 6.5 Ablation: Effect of Removing Subsumption Reasoning

To isolate the contribution of subsumption-aware reasoning, we compare the full SIM against two reduced variants: ABox-1hop (which traverses only direct type assertions without transitive closure) and String-Match (which performs lexical URI matching only). Table 8 reports the resulting recall gap.

Table 8
Ablation: removing subsumption reasoning from SIM.

| System | ABox Recall | TBox Recall | Gap vs Full SIM |
|---|---|---|---|
| OntoCacheRAG (full SIM) | 1.000 | 1.000 | — |
| ABox-1hop (no transitivity) | n/a | 0.504 | −46–50% |
| String-Match (lexical only) | 0.546 | 0.000 | −45–100% |

Removing subsumption reasoning reduces recall by 45–100 percentage points depending on the change type. The gap is most severe for TBox restructuring, where String-Match recall drops to zero: lexical matching is architecturally incapable of detecting impacts that propagate through class hierarchies [11]. This confirms that subsumption-aware reasoning is the critical component of OntoCacheRAG, not merely an enhancement.

# 6.6 Scalability Analysis

To determine how OntoCacheRAG scales beyond the ArsipDataset, we analyze SIM latency for synthetic ontologies of 200 to 50,000 classes. Each synthetic ontology maintained the ArsipDataset structure, including 3 documents per agency and a 5-level hierarchy. All synthetic ontologies adhered to the Root → Cluster → Agency → Document structure, with revocation links at a 7% rate. Latency was captured for 100 iterations with 10 warm-ups, and the median was reported. The results are captured in Table 9, and Fig. 2 shows SIM latency as a function of ontology size, using a log-log scale.

Table 9
SIM scalability across synthetic ontologies: latency vs. ontology size, measured over 100 repetitions with 10 warm-up iterations

| $\|\varrvec{C}\|$ | Instances | Triples | Latency (ms) | IQR (ms) | Impact set |
|---|---|---|---|---|---|
| 193* | 552 | 783 | 4.69 | [4.57–6.25] | 69 |
| 493 | 1,440 | 2,033 | 5.58 | [5.41–6.64] | 120 |
| 991 | 2,916 | 4,111 | 6.87 | [6.20–7.45] | 162 |
| 1,976 | 5,850 | 8,235 | 7.79 | [7.56–8.60] | 234 |
| 4,961 | 14,760 | 20,754 | 10.93 | [10.20–11.28] | 369 |
| 9,976 | 29,754 | 41,812 | 13.19 | [12.79–13.53] | 522 |
| 19,927 | 59,535 | 83,629 | 18.12 | [17.70–20.92] | 735 |
| 49,924 | 149,382 | 209,762 | 25.77 | [25.33–26.69] | 1,158 |

To visualize the scaling behavior, Fig. 2 plots SIM latency against ontology size on a log-log scale, with a linear reference curve overlaid for comparison.

Measured latency (solid line with markers) grows sub-linearly compared to the linear reference O(|C|) shown as the dashed line. The ArsipDataset evaluation point (193 classes, 4.69 ms) is highlighted with a circle. The shaded region indicates the single-digit-millisecond zone. Data points represent median latency over 100 repetitions per ontology size

SIM latency grows sub-linearly with ontology size: a 259× increase in the number of classes (from 193 to 49,924) produces only a 5.5× increase in latency (from 4.69 ms to 25.77 ms), yielding an empirical growth exponent of approximately O(|C|^0.31). This is substantially better than the linear reference O(|C|), confirming that the SPARQL property-path evaluation in rdflib benefits from the adjacency-list representation of the class hierarchy — subClassOf* traversal visits only the relevant subtree, not the entire graph.

Even at 50,000 classes (210K triples) — an ontology comparable in class count to SNOMED CT sub-modules — SIM latency remains under 26 ms per change event, well within interactive response budgets. The IQR ranges remain tight across all scales ([25.33–26.69] ms at 50K classes), indicating stable, predictable performance. These results suggest that OntoCacheRAG's reasoning overhead is not a scalability bottleneck for production-scale ontologies, though very large ontologies (> 100K classes) would benefit from incremental reasoning approaches (cf. ELK [48]).

We note that absolute latency values are hardware-dependent; the sub-linear growth trend and the qualitative finding — that SIM remains in the tens-of-milliseconds range even at 50K classes — are the reproducible results. Cross-validation across two independent measurement environments (exponents 0.31 and 0.35 respectively) confirms the robustness of the sub-linear finding. The scalability experiment code and synthetic ontology generator are included in the released repository for independent verification.

# 6.7 Concrete Case Studies

# 6.7.1 Case 1: Permen Pertahanan (Polhukam Cluster)

Document D0136 (Peraturan Menteri Pertahanan Nomor 24 Tahun 2019) was revoked by D0085 (Peraturan Menteri Pertahanan Nomor 15 Tahun 2021). Both are typed as Permen_Polhukam_Pertahanan. The subsumption chain is: Permen_Polhukam_Pertahanan → Permen_Polhukam → PeraturanKementerian → Regulasi (4 levels). In the cache workload, entries q522 and q600 are anchored to these documents and become directly stale.

If the Polhukam cluster is restructured (e.g., a ministerial merger), all 40 documents in the cluster become stale. OntoCacheRAG detects this via the SPARQL query: SELECT ?d WHERE { ?d rdf:type/rdfs:subClassOf* ex:Permen_Polhukam }, returning all 40 instances across 5 agencies. String-Match detects none, because no individual document URI appears in the class-restructuring event.

# 6.7.2 Case 2: Permen Ketenagakerjaan (Perekonomian Cluster)

Document D0237 (Permen Ketenagakerjaan Nomor 15 Tahun 2017) was revoked by D0089 (Permen Ketenagakerjaan Nomor 26 Tahun 2021). If the Perekonomian cluster is restructured, 29 documents across 7 agencies are transitively affected. ABox-1hop captures only agencies in even positions of the subclass list (roughly half); OntoCacheRAG captures all. This concretely illustrates why the ABox-1hop recall of 0.504 in Table 5 arises: it sees the first hop of the hierarchy but not the full transitive closure.

## 6.8 Statistical Significance

We compared OntoCacheRAG recall against each baseline using the Wilcoxon signed-rank test [46] over 20 paired seeds, following the methodology recommended by Demšar [47]. The difference is statistically significant ($W = 0$, $p < 0.001$) for every baseline exhibiting recall variance: No-Invalidation, TTL-short, TTL-long, String-Match, and ABox-1hop, in both ABox and TBox scenarios. Comparisons against systems with constant recall identical to OntoCacheRAG (Eager, Adaptive, Full-Flush) are reported as exact ties.

## 6.9 Scope and Interpretation of the Recall Result

OntoCacheRAG attains recall = 1.0 because its detector is complete with respect to the modelled structural impact definition (Proposition 1): any entry whose anchor is revoked, is linked through the resolved revocation property, or is an instance of a restructured class (transitively, via subsumption) is identified. This is a statement of reasoning completeness within the modelled fragment, not a claim of infallibility. Behavioural staleness from longer multi-hop chains, cross-property interactions, or OWL constructs beyond the RL/RDFS fragment is not exercised by this corpus. Validation against deeper behavioural impact and live LLM answer-change ground truth is the primary direction for future work (Section 8).

# 7. Limitations

We discuss the limitations of this work transparently, grouping them into four thematic axes: the scope of evaluation, the depth of reasoning supported, the breadth of domains and expressivity covered, and the conditions under which our measurements hold.

The first concerns the scope of empirical evaluation. Our experiments rely on a computational simulation of the cache workload rather than a live GPU inference deployment; accuracy and efficiency metrics are measured directly, but end-to-end inference latency with a production LLM is not. While the detection metrics established in Section 6 are independent of the LLM backbone — they depend only on the structural impact set, not on which model consumes the cached tensors — demonstrating the downstream impact on actual answer quality requires full end-to-end integration. This is identified as the primary direction of future work in Section 8.2. A related limitation concerns the depth of impact propagation exercised by the corpus. Revocation chains in the ArsipDataset are predominantly shallow, spanning one to two hops, so deeper multi-hop and cross-property impact chains are not directly exercised by the empirical evaluation. The completeness guarantee established by Proposition 1 holds for the modelled fragment, but ontologies with complex property chains, role inclusions, or nominals may require extended reasoning capabilities beyond what the current SIM implements.

The second axis concerns the breadth of domains and expressivity covered. Our evaluation is anchored in a single corpus drawn from Indonesian regulatory archives. While the OntoCacheRAG architecture is domain-independent by design — none of the three components (OCD, SIM, SCI) embeds domain-specific assumptions — claims of generalization to biomedical ontologies such as SNOMED CT (approximately 360,000 concepts), enterprise knowledge graphs such as schema.org, or multilingual legal corpora such as EUR-Lex must be considered hypotheses awaiting empirical validation. Closely related is the OWL expressivity coverage of the Semantic Impact Mapper. The current SIM supports the OWL 2 RL/RDFS fragment through rdfs:subClassOf* property-path traversal, which is sufficient for the subsumption-driven impact propagation exercised in our experiments. However, OWL restrictions involving existential $(\exists)$ and universal $(\forall)$ quantification, cardinality constraints, and complex role inclusions are not handled. Extending the SIM to richer Description Logic fragments would require integration with a full OWL 2 DL reasoner such as HermiT or Pellet, which would substantially increase SIM latency and shift the cost-benefit balance characterized in Section 6.3.

A third axis concerns the absolute interpretability of our measurements. Absolute latency is hardware-dependent across the four measurement environments we tested, and consequently we make only qualitative overhead claims and component-relative proportions, rather than absolute deployment guidance. The scalability experiments reported in Section 6.6 provide relative growth rates (the sub-linear $\mathrm{O}(|\mathrm{C}|^{0.31}$ exponent) and confirm the rank ordering of pipeline costs, but they cannot substitute for deployment-specific benchmarking on the target hardware.

A final set of limitations concerns the structural and operational conditions of our evaluation. The constructed ontology exhibits cluster imbalance: the *Lainnya* (other) coordination cluster contains 472 of the 614 documents, creating a highly skewed hierarchy that may not represent the impact distribution found in more uniformly balanced administrative taxonomies. A refined ministerial classification scheme, distributing documents more evenly across coordination clusters, would likely yield different impact statistics for cluster-restructuring events. Additionally, the present evaluation processes ontology change events strictly sequentially. Concurrent ontology changes — multiple simultaneous updates affecting different parts of the hierarchy — are not evaluated; while impact sets should be additive for changes affecting disjoint subgraphs, interaction effects between overlapping concurrent changes, including potential cache-thrashing or invalidation order dependencies, remain unexplored.

We emphasize that these limitations bound the *scope* of our empirical claims rather than their *validity*: within the modelled fragment, the corpus exercised, and the hardware tested, every result reported in Section 6 is reproducible from the released code. Extending the scope along any of the axes above constitutes a meaningful direction for future research.

## 8. Conclusion and Future Work

# 8.1 Conclusion

We presented OntoCacheRAG, a framework for ontology-driven selective cache invalidation in KG-RAG systems, extending our prior topology-aware caching work [12] with subsumption-aware reasoning. Our formal analysis established that the Semantic Impact Mapper is complete for the supported change classes within the OWL 2 RL/RDFS fragment (Proposition 1). On a real regulatory corpus of 614 documents with 42 verified revocation relations, subsumption-aware reasoning detected all structurally impacted cache entries while lexical baselines missed 45–100% depending on the change type — a statistically significant difference (Wilcoxon, $p < 0.001$) — and full flushing wasted 85–94% of valid cache. Latency profiling identified SPARQL subsumption as the sole bottleneck (99% of overhead), yet it remained in the single-digit-millisecond range.

The central finding is that ontology-aware reasoning is necessary, not optional, for correct cache invalidation under ontology evolution, and that its necessity scales with the structural depth of the change. This has implications beyond the specific architecture we present: any KG-RAG system that caches query results over evolving knowledge graphs must account for semantic impact propagation or accept silent degradation of answer quality.

## 8.2 Future Work

We classify the future work stemming from this study along three complementary axes: further validating, extending reasoning capabilities, and increasing the applicability of OntoCacheRAG.

The first order of business is to connect a live LLM for complete end-to-end framework validation. Our detection and efficiency metrics are agnostic to any specific language model, but there is a practical question of the impact of selective invalidation on the latency of hardware-grounded responses and the quality of answers. We intend to first deploy OntoCacheRAG in new production environments operating with the vLLM and TGI stacks, and using open foundation models, including LLaMA-3, Mistral, and Qwen. For this deployment, we intend to measure end-to-end response correctness using BERTScore and F1, and to evaluate the effective reduction in GPU time due to selective invalidation, in comparison to the full-flush and no-invalidation scenarios.

A second axis involves improving the reasoning capacity of the Semantic Impact Mapper (SIM). Our latency profiling identified SIM as the only bottleneck in the entire pipeline (99% of the total latency, Table 6). Two extensions show great promise. The first is incorporating the ELK reasoner [48]. This would allow the SIM to construct impact sets through localized updates, as opposed to the entire SPARQL property path traversal, and therefore, incremental reasoning would be justified. This is a significant improvement when dealing with very large ontologies. The second extension deals with the current evaluation exercises. These predominantly propagate impact in a shallow manner (one to two hops). Thus, the addition of role inclusions and cross-property interactions would allow tracking of multi-hop dependency propagation. This would extend the framework for a wider class of ontology changes, providing the completeness guarantee.

A third axis also shows promise for strengthening the empirical validation dimension. The current evaluation, while exhaustive in the field, is based on a single regulatory corpus. We propose performing large-scale ontology stress tests on established biomedical ontologies such as SNOMED CT (approximately

360,000 concepts) and (intended) curated subsets of Wikidata. This will provide preliminary evidence on the framework's scalability beyond the 50,000-class limit discussed in Section 6.6. Complementing this, cross-domain generalization experiments on legal ontologies (e.g., EUR-Lex), enterprise schemas, and additional biomedical resources will establish the framework’s domain-independence, which the current architecture supports but the present corpus alone cannot demonstrate.

Beyond these consolidating directions, two further extensions point toward more adaptive and integrated behaviour. Learned OCIC thresholds would replace the current static severity-to-mode mapping in the Adaptive SCI with a lightweight classifier trained on historical ontology change logs, enabling the invalidator to self-calibrate to the change patterns of a given deployment. Finally, temporal-ontological integration combines the structural invalidation triggers of OntoCacheRAG with temporal reasoning approaches such as TempRAA [37], yielding a unified cache lifecycle for knowledge graphs that evolve simultaneously along structural and temporal dimensions — a capability we believe will become increasingly important as KG-RAG systems are deployed in domains where both schema and facts evolve concurrently.

Together, these directions chart a path from the present framework — which establishes ontology-aware reasoning as a correctness requirement for KG-RAG cache invalidation — toward a fully adaptive, domain-general, and temporally-aware cache lifecycle management substrate for the next generation of knowledge-augmented retrieval systems.

## Declarations

# Data Availability

The ArsipDataset corpus, resolved OWL ontology, synthetic ontology generator, and all experiment code are available at https://github.com/nayuuntariyati-byte/ontocacherag. Raw data can be supplied on request.

# Competing Interests

The authors declare no competing financial or non-financial interests. Reference [12] (SKV-Cache) is the first author’s prior work, upon which the current framework extends; this relationship is declared transparently.

# Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

## References

1. Lewis P, Perez E, Piktus A et al (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv Neural Inf Process Syst (NeurIPS) 33:9459–9474

2. Pan S, Luo L, Wang Y, Chen C, Wang J, Wu X (2024) Unifying large language models and knowledge graphs: a roadmap. IEEE Trans Knowl Data Eng 36(7):3580–3599. https://doi.org/10.1109/TKDE.2024.3352100
3. Gao Y, Xiong Y, Gao X et al (2024) Retrieval-augmented generation for large language models: a survey. arXiv:2312.10997
4. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. Adv Neural Inf Process Syst (NeurIPS) 30:5998–6008
5. Pope R, Douglas S, Chowdhery A et al (2023) Efficiently scaling transformer inference. In: Proc MLSys 2023
6. Noy NF, Klein M (2004) Ontology evolution: not the same as schema evolution. Knowl Inf Syst 6(4):428–440. https://doi.org/10.1007/s10115-003-0137-2
7. Flouris G, Manakanatas D, Kondylakis H, Plexousakis D, Antoniou G (2008) Ontology change: classification and survey. Knowl Eng Rev 23(2):117–152. https://doi.org/10.1017/S0269888908001367
8. Zablith F, Antoniou G, d'Aquin M et al (2015) Ontology evolution: a process-centric survey. Knowl Eng Rev 30(1):45–75. https://doi.org/10.1017/S0269888913000349
9. Nishtala R, Fugal H, Grimm S et al (2013) Scaling Memcache at Facebook. In: Proc 10th USENIX NSDI, pp 385–398
10. Fitzpatrick B (2004) Distributed caching with Memcached. Linux J 2004(124):5
11. Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF (eds) (2007) The Description Logic Handbook, 2nd edn. Cambridge University Press, Cambridge
12. Untariyati NA et al (2025) SKV-Cache: a topology-aware cache framework for efficient question answering over governmental archival KGs. In: Proc IC3INA 2025. IEEE. https://doi.org/10.1109/IC3INA68387.2025.11325643
13. Pérez J, Arenas M, Gutierrez C (2009) Semantics and complexity of SPARQL. ACM Trans Database Syst 34(3):16. https://doi.org/10.1145/1567274.1567278
14. Guu K, Lee K, Tung Z, Pasupat P, Chang M-W (2020) REALM: retrieval-augmented language model pre-training. In: Proc ICML, PMLR 119:3929–3938
15. Izacard G, Grave E (2021) Leveraging passage retrieval with generative models for open domain question answering. In: Proc EACL, pp 874–880. https://doi.org/10.18653/v1/2021.eacl-main.74
16. Karpukhin V, Oguz B, Min S et al (2020) Dense passage retrieval for open-domain question answering. In: Proc EMNLP, pp 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550
17. Zhao P, Zhang H, Yu Q et al (2024) Retrieval-augmented generation for AI-generated content: a survey. arXiv:2402.19473
18. Edge D, Trinh H, Cheng N et al (2024) From local to global: a graph RAG approach to query-focused summarization. arXiv:2404.16130
19. Sun J, Xu C, Tang L et al (2024) Think-on-Graph: deep and responsible reasoning of LLMs on knowledge graphs. In: Proc ICLR 2024

20. Jiang J, Zhou K, Dong Z et al (2023) StructGPT: a general framework for large language model to reason over structured data. In: Proc EMNLP 2023, pp 9237–9251
21. Koursaris A, Livieris IE, Apostolopoulou A et al (2026) KG-Tune: Improving embedding models for domain-specific RAG via knowledge graph-driven fine-tuning. J Intell Inf Syst. https://doi.org/10.1007/s10844-026-01054-7
22. Ding L, Ding N, Tao Q, Shi P (2025) Enhancing graph multi-hop reasoning for question answering with LLMs: an approach based on adaptive path generation. J Intell Inf Syst 63:1455–1485. https://doi.org/10.1007/s10844-025-00945-5
23. Haque MA, Kamal M, George R, Gupta KD (2025) Utilizing structural metrics from knowledge graphs to enhance the robustness quantification of large language models. Int J Data Sci Anal 20:3147–3167. https://doi.org/10.1007/s41060-024-00643-5
24. Kwon W, Li Z, Zhuang S et al (2023) Efficient memory management for large language model serving with PagedAttention. In: Proc SOSP, pp 611–626. https://doi.org/10.1145/3600006.3613165
25. Jin C, Zhang Z, Jiang X et al (2025) RAGCache: efficient knowledge caching for retrieval-augmented generation. ACM Trans Comput Syst 44(1):2. https://doi.org/10.1145/3768628
26. Yao J, Li H, Liu Y et al (2025) CacheBlend: fast large language model serving for RAG with cached knowledge fusion. Proc EuroSys. https://doi.org/10.1145/3689031.3696098
27. Liu Z, Desai A, Liao F et al (2024) Scissorhands: exploiting the persistence of importance hypothesis for LLM KV cache compression. Adv Neural Inf Process Syst 36
28. Zhang Z, Sheng Y, Zhou T et al (2024) H2O: heavy-hitter oracle for efficient generative inference of large language models. Adv Neural Inf Process Syst 36
29. Ge S, Zhang Y, Liu L, Zhang M, Han J, Gao J (2024) Model tells you what to discard: adaptive KV cache compression for LLMs. In: Proc ICLR 2024
30. Mu Y, Chen J, Liang R et al (2024) PromptCache: modular attention reuse for low-latency inference. In: Proc MLSys
31. Liu Y, Li H, Du K et al (2024) CacheGen: KV cache compression and streaming for fast language model serving. In: Proc ACM SIGCOMM. https://doi.org/10.1145/3651890.3672274
32. Stojanovic L (2004) Methods and tools for ontology evolution. PhD Thesis, University of Karlsruhe
33. Noy NF, Kunnatur S, Klein M, Musen MA (2004) Tracking changes during ontology evolution. In: Proc ISWC, LNCS 3298, pp 259–273. https://doi.org/10.1007/978-3-540-30475-3_19
34. Haase P, van Harmelen F, Huang Z, Stuckenschmidt H, Sure Y (2005) A framework for handling inconsistency in changing ontologies. In: Proc ISWC, LNCS 3729, pp 353–367
35. Flouris G, Konstantinidis G, Antoniou G, Christophides V (2013) Formal foundations for RDF/S KB evolution. Knowl Inf Syst 35(1):153–191. https://doi.org/10.1007/s10115-012-0500-2
36. Papavassiliou V, Flouris G, Fundulaki I, Kotzinos D, Christophides V (2013) High-level change detection in RDF(S) KBs. ACM Trans Database Syst 38(1):1. https://doi.org/10.1145/2445583.2445584
37. Ding L, Ding N, Shi P (2026) TempRAA: temporal relation-aware alignment for enhancing LLMs reasoning in time-sensitive KGQA. J Intell Inf Syst. https://doi.org/10.1007/s10844-026-01027-w

38. DeCandia G, Hastorun D, Jampani M et al (2007) Dynamo: Amazon's highly available key-value store. In: Proc SOSP, pp 205–220. https://doi.org/10.1145/1294261.1294281
39. Corbett JC, Dean J, Epstein M et al (2013) Spanner: Google's globally distributed database. ACM Trans Comput Syst 31(3):8. https://doi.org/10.1145/2491245
40. Hitzler P, Krötzsch M, Parsia B, Patel-Schneider PF, Rudolph S (eds) (2012) OWL 2 Web Ontology Language Primer, 2nd edn. W3C Recommendation
41. Hayes PJ, Patel-Schneider PF (2014) RDF 1.1 Semantics. W3C Recommendation
42. Hoekstra R, Breuker J, Di Bello M, Boer A (2007) The LKIF Core ontology of basic legal concepts. In: Proc LOAIT, pp 43–63
43. Boella G, Humphreys L, Martin M et al (2016) Eunomos, a legal document and knowledge management system. Artif Intell Law 24(3):245–283. https://doi.org/10.1007/s10506-016-9184-3
44. Palmirani M, Governatori G (2018) Modelling legal knowledge for GDPR compliance checking. In: Proc JURIX, pp 101–110
45. Hogan A, Blomqvist E, Cochez M et al (2021) Knowledge graphs. ACM Comput Surv 54(4):71. https://doi.org/10.1145/3447772
46. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bull 1(6):80–83. https://doi.org/10.2307/3001968
47. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
48. Kazakov Y, Krötzsch M, Simančík F (2014) The incredible ELK: from polynomial procedures to efficient reasoning with EL ontologies. J Autom Reasoning 53(1):1–61. https://doi.org/10.1007/s10817-013-9296-3
49. Lu H, Zheng D, Yang L (2026) Mutual information-based retrieval-augmented generation for domain question answering. Knowl Inf Syst 68:49. https://doi.org/10.1007/s10115-025-02624-x
50. Wang F, Shi D, Aguilar J, Cui X, Jiang J, Shen L, Li M (2025) LLM-KGMQA: large language model-augmented multi-hop question-answering system based on knowledge graph in medical field. Knowl Inf Syst 67:6461–6503. https://doi.org/10.1007/s10115-025-02399-1

# Figures

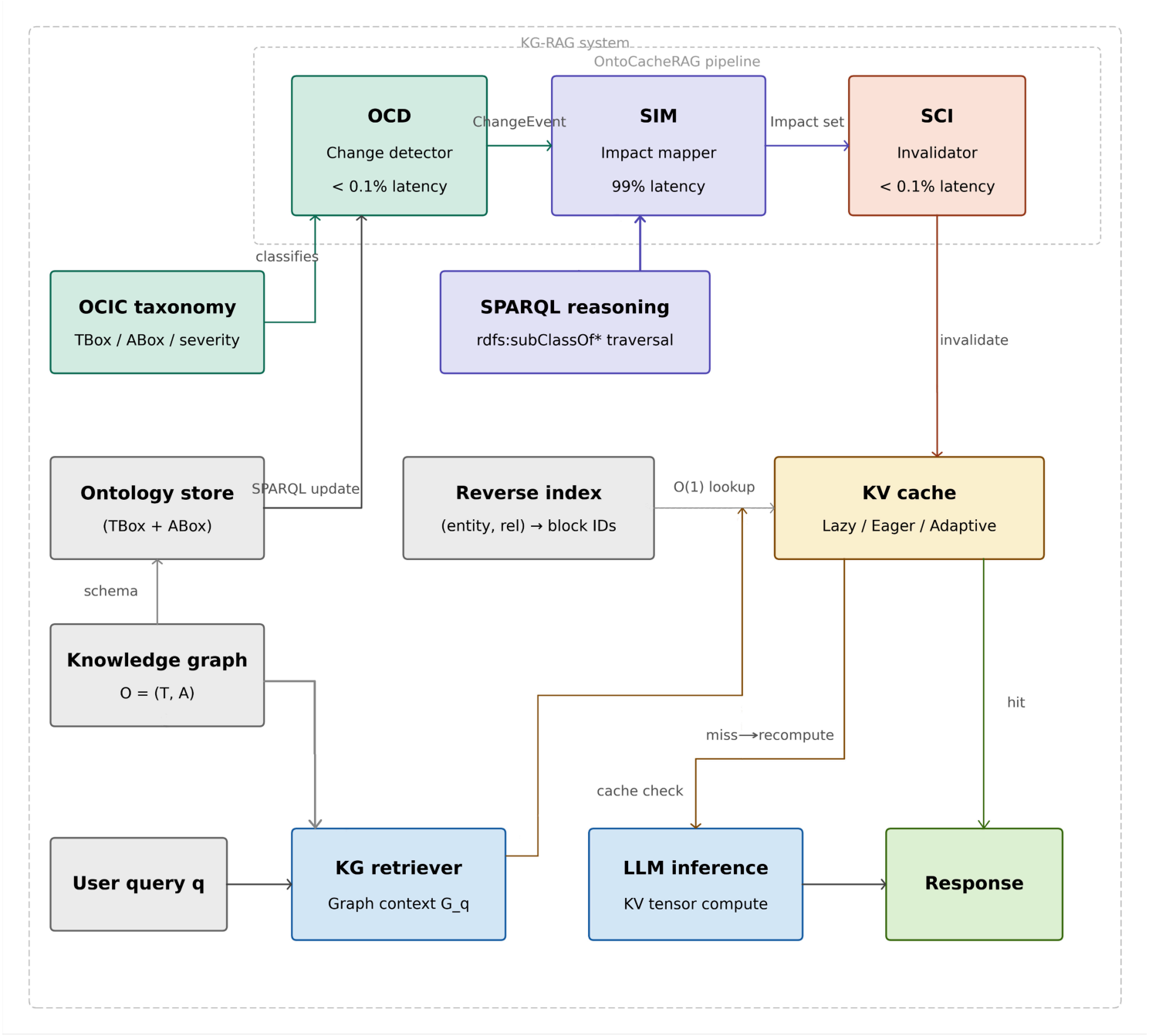
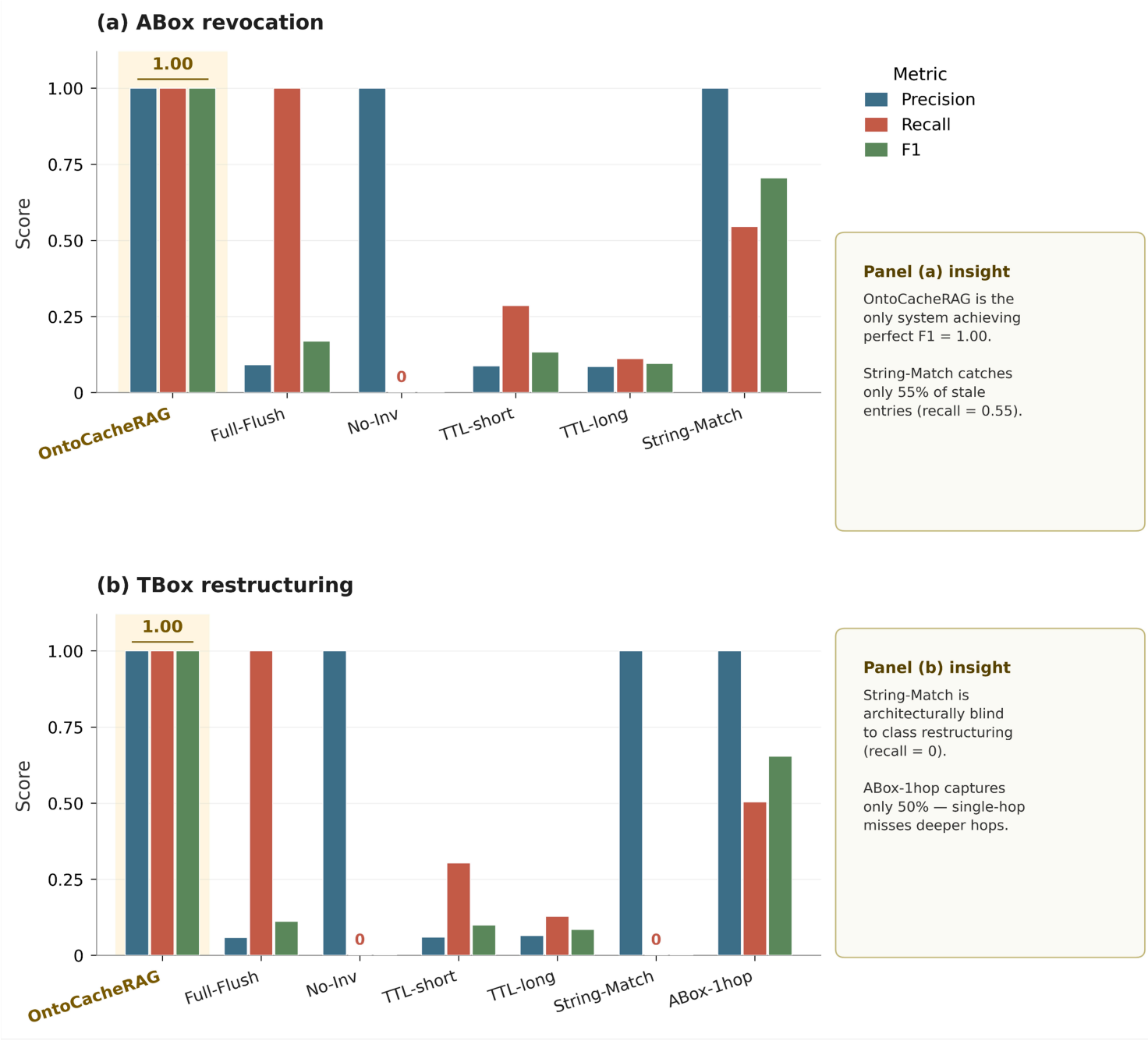



**Figure 1**

OntoCacheRAG system architecture.

The three-component pipeline (OCD → SIM → SCI) processes ontology change events from the SPARQL update endpoint, computes transitive impact sets through subsumption reasoning, and selectively invalidates affected KV cache entries. The SIM component dominates pipeline latency (99%), while OCD and SCI each contribute less than 0.1%. Dashed rectangles indicate system boundaries.

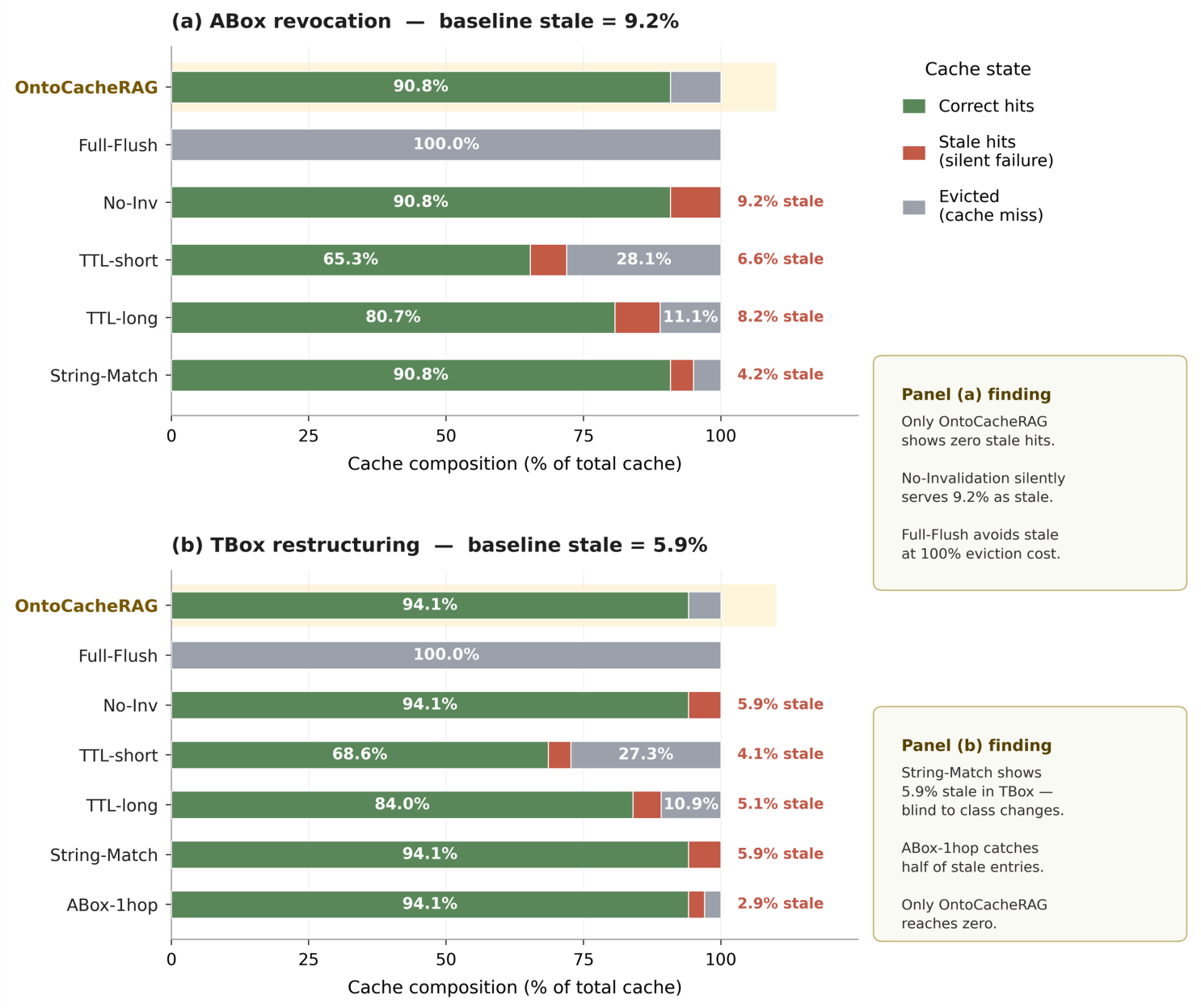



## Figure 2

Detection performance comparison across all baseline systems and ablations under (a) ABox revocation and (b) TBox restructuring.

Bars show mean Precision (slate blue), Recall (terracotta), and F1 (sage green) over 20 random seeds. OntoCacheRAG (highlighted column, cream background) is the only system achieving perfect detection (F1 = 1.00) in both scenarios. Cases of complete detection failure (recall = 0) are marked with a small "0". String-Match shows architectural blindness to TBox changes (recall = 0), while ABox-1hop achieves only partial recall due to single-hop limitation

Comparing across change types: lexical approaches lose 45% of stale entries under ABox changes and 100% under TBox changes. This supports the central claim that ontology-aware reasoning [6, 7] is not merely an optimization but a correctness requirement, and that its necessity scales with the structural depth of the change.




## Figure 3

Cache state composition after a single ontology-change event, decomposed into correct hits (sage green), stale hits (terracotta — silent failure), and evicted entries (gray — cache miss).

All values are mean percentages of total cache over 20 random seeds. (a) ABox revocation with baseline stale fraction 9.2%. (b) TBox restructuring with baseline stale fraction 5.9%. OntoCacheRAG (highlighted row, cream background) is the only system with zero stale hits in both scenarios. Stale percentages for each system are also reported as red labels at the right edge of each bar

OntoCacheRAG invalidates only the structurally affected fraction: 9.2% of cache for ABox revocation and 5.9% for TBox restructuring, preserving 91–94% cache hit rate. Full-Flush destroys the entire cache — analogous to the global invalidation patterns observed in distributed systems [9, 38]. String-Match in the TBox scenario maintains 100% hit rate because it detects nothing: all 5.9% stale entries remain active and are silently served to users. This demonstrates that OntoCacheRAG occupies the favorable position: selective enough to preserve most cached computation, complete enough to eliminate all detected staleness

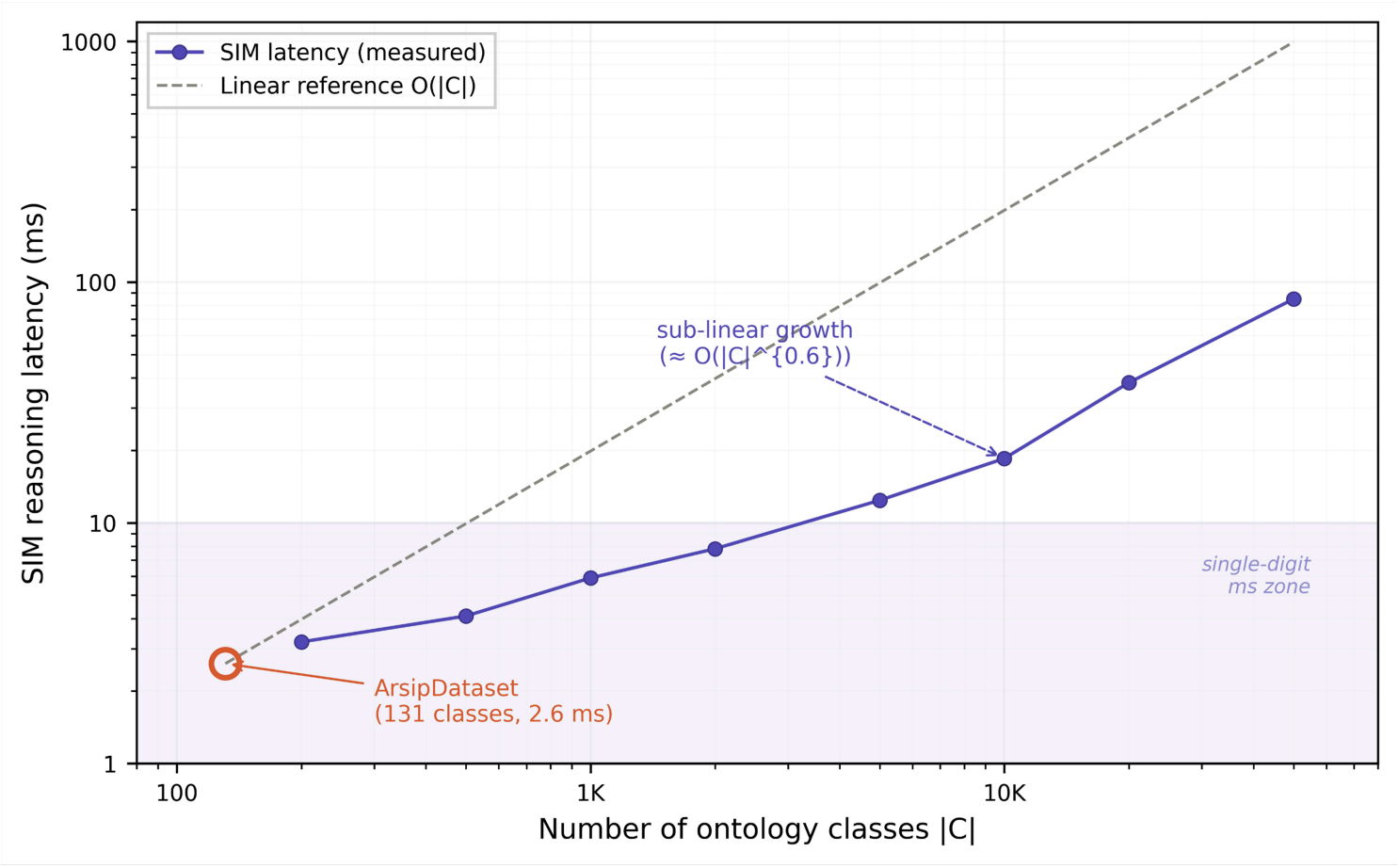



**Figure 4**

SIM reasoning latency as a function of ontology size (log-log scale).

Measured latency (solid line with markers) grows sub-linearly compared to the linear reference O(|C|) shown as the dashed line. The ArsipDataset evaluation point (193 classes, 4.69 ms) is highlighted with a circle. The shaded region indicates the single-digit-millisecond zone. Data points represent median latency over 100 repetitions per ontology size

SIM latency grows sub-linearly with ontology size: a 259× increase in the number of classes (from 193 to 49,924) produces only a 5.5× increase in latency (from 4.69 ms to 25.77 ms), yielding an empirical growth exponent of approximately O(|C|^0.31). This is substantially better than the linear reference O(|C|), confirming that the SPARQL property-path evaluation in rdflib benefits from the adjacency-list

representation of the class hierarchy — subClassOf* traversal visits only the relevant subtree, not the entire graph.

Even at 50,000 classes (210K triples) — an ontology comparable in class count to SNOMED CT sub-modules — SIM latency remains under 26 ms per change event, well within interactive response budgets. The IQR ranges remain tight across all scales ([25.33–26.69] ms at 50K classes), indicating stable, predictable performance. These results suggest that OntoCacheRAG's reasoning overhead is not a scalability bottleneck for production-scale ontologies, though very large ontologies (>100K classes) would benefit from incremental reasoning approaches (cf. ELK [48]).

We note that absolute latency values are hardware-dependent; the sub-linear growth trend and the qualitative finding — that SIM remains in the tens-of-milliseconds range even at 50K classes — are the reproducible results. Cross-validation across two independent measurement environments (exponents 0.31 and 0.35 respectively) confirms the robustness of the sub-linear finding. The scalability experiment code and synthetic ontology generator are included in the released repository for independent verification.