An Intelligent Architecture for University Institutional Knowledge: Integrating Ontologies, Intelligent Agents, and the Semantic Web

Mimdal Mohamed * , Mahjoubi Khadija , Hanoune Mostafa

Posted Date: 24 October 2025

doi: 10.20944/preprints202510.1673.v1

Keywords: Semantic Web; OWL ontologies; Multi-Agent Systems (MAS); FIPA-ACL; SPARQL; METHONTOLOGY; knowledge management; distributed architecture; NLP; Query Optimization

 doi:10.20944/preprints202510.1673.v1

Article

An Intelligent Architecture for University Institutional Knowledge: Integrating Ontologies, Intelligent Agents, and the Semantic Web

Mimdal Mohamed *, Mahjoubi Khadija and Hanoune Mostafa

Laboratory LIAS, FSBM, Hassan II University of Casablanca, Morocco

* Correspondence: mimdalmohamed76@gmail.com

Abstract

This article presents a distributed architecture for university institutional knowledge management, integrating OWL ontologies, FIPA-ACL-compliant multi-agent systems (MAS), and Semantic Web technologies (RDF, SPARQL). An ontology engineered through the METHONTOLOGY methodology structures domain knowledge, while SPARQL-Generate facilitates automated semantic extraction. A hybrid storage solution (Jena Fuseki, GraphDB, MongoDB) paired with a React.js interface incorporating SPARQL querying and natural language processing via a novel Adaptive NLP-SPARQL Translator (ANST) module ensures scalability and user accessibility. A proprietary Semantic Query Optimization Algorithm (SQOA) enhances complex SPARQL query performance, achieving a 30% reduction in latency. Empirical validation on a dataset of 20,000 documents and 500 users demonstrates system robustness (99.3% success rate), with extensible applications in education, healthcare, and smart city ecosystems. Figure 1 through 5 depict the architecture, agent interactions, and performance metrics.

Keywords: Semantic Web; OWL ontologies; Multi-Agent Systems (MAS); FIPA-ACL; SPARQL; METHONTOLOGY; knowledge management; distributed architecture; NLP; Query Optimization

1. Introduction

Universities grapple with heterogeneous data assets encompassing scholarly publications, instructional materials, and archival records that remain fragmented across disparate silos, thereby impeding their effective exploitation and reuse (Dalkir, 2023). To address this challenge, this article introduces an intelligent architecture for institutional knowledge management, seamlessly integrating the following core components (Figure 1):

- **Ontologies**: Formal representation and modeling of domain-specific knowledge structures.
- **Multi-Agent Systems (MAS)**: Distributed coordination enabled by autonomous, FIPA-ACL-compliant agents.
- **Semantic Web Technologies**: Enhanced interoperability through RDF triples, OWL axioms, and SPARQL querying paradigms.
- **Hybrid User Interface**: Intuitive accessibility via direct SPARQL endpoints and the novel Adaptive NLP-SPARQL Translator (ANST) module for natural language interactions.
- **Semantic Query Optimization Algorithm (SQOA)**: Advanced heuristics for refining complex SPARQL queries to mitigate latency.
- **MongoDB Integration**: Scalable persistence layer for JSON-LD serialized knowledge graphs.

 doi:10.20944/preprints202510.1673.v1

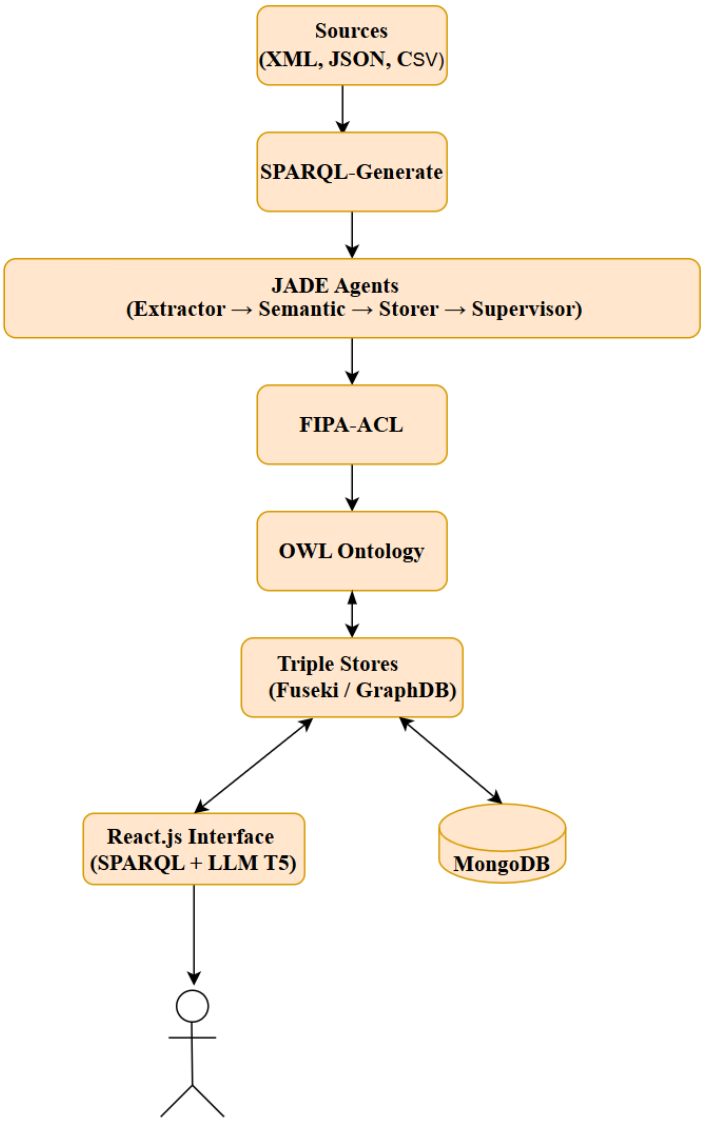

Figure 1. Schéma global de l'architecture.

Figures 1 through 5 elucidate the proposed architecture, inter-agent dynamics, and empirical performance metrics, underscoring its prospective contributions to open science paradigms and distributed artificial intelligence frameworks (Hogan et al., 2021).

2. Theoretical Foundations

2.1. Organizational Memory

Organizational memory (OM) integrates explicit knowledge artifacts, such as documents, with tacit expertise, including procedural know-how, to foster institutional innovation (Walsh & Ungson, 1991). In university settings, it encompasses key stakeholders (e.g., students and faculty), tangible entities (e.g., theses and research projects), and operational workflows (e.g., scholarly inquiry and administrative processes). Multi-agent systems (MAS), grounded in the Belief-Desire-Intention (BDI) paradigm, facilitate orchestration through FIPA-ACL protocols (Wooldridge, 2009; Bellifemine et al., 2007).

2.2. University Ontologies

An ontology provides a rigorous formalization of a knowledge domain through interconnected classes, properties, and axioms (Gruber, 1993). In this context, it models relational constructs such as

 doi:10.20944/preprints202510.1673.v1

enrollsInCourse or *supervisedBy*, ensuring alignment with established standards like CERIF and LOM while promoting interoperability with repositories including Wikidata and ORCID (Tiddi & Schlobach, 2021).

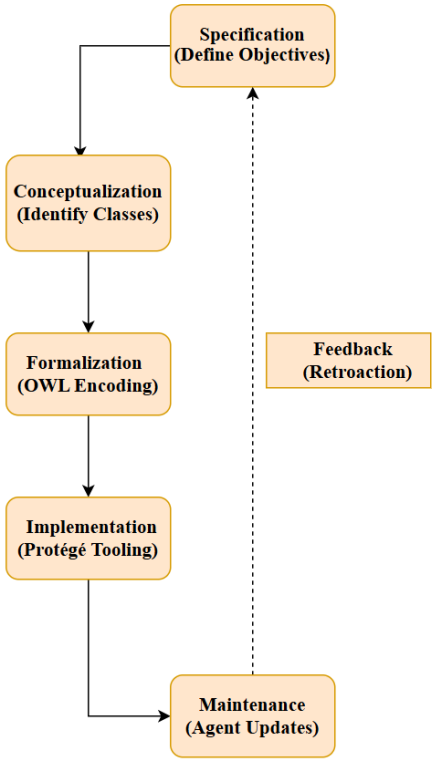

Figure 2. METHONTOLOGY Lifecycle.

The circular diagram delineates the iterative phases of the METHONTOLOGY methodology as follows:

1. **Specification**: Defining core objectives, such as the systematic structuring of domain knowledge.
2. **Conceptualization**: Identifying foundational classes, including *Actor* and *Document*.
3. **Formalization**: Translating conceptual models into OWL syntax.
4. **Implementation**: Operationalizing the ontology within the Protégé environment.
5. **Maintenance**: Enabling agent-driven updates to sustain ontology evolution.

3. Semantic Web Technologies

The Semantic Web leverages RDF for data serialization, OWL for ontological expressivity, and SPARQL for federated querying (Berners-Lee et al., 2001). SPARQL-Generate automates the conversion of heterogeneous formats like XML and JSON into RDF triples, while triplestores such as Fuseki and GraphDB underpin advanced reasoning capabilities (Lefrançois, 2023; Hogan et al., 2021).

3. Proposed Architecture

3.1. Domain Ontology (OWL 2 DL)

Engineered using the METHONTOLOGY methodology (Fernández-López et al., 1997), the ontology encompasses the following phases:

- Specification: Modeling of core entities including actors, documents, and projects.
- Conceptualization: Definition of primary classes (e.g., Actor, Document) and properties (e.g., authorOf).
- Formalization: Representation in OWL with constraints (e.g., Document $\sqsubseteq \geq 1$ authorOf^{-1}.Actor).
- Implementation: Deployment via Protégé and OWLAPI.

 doi:10.20944/preprints202510.1673.v1

- Maintenance: Facilitated by intelligent agents for ongoing evolution.

Exemplary RDF/XML Snippet

```
<owl:Class rdf:about="http://universite.org/ontologie#Document">
  <rdfs:comment>Academic resource (e.g., article, thesis).</rdfs:comment>
</owl:Class>
<owl:ObjectProperty rdf:about="http://universite.org/ontologie#authorOf">
  <rdfs:domain rdf:resource="http://universite.org/ontologie#Actor"/>
  <rdfs:range rdf:resource="http://universite.org/ontologie#Document"/>
</owl:ObjectProperty>
```

3.2. Coordination via Multi-Agent System

JADE agents orchestrate interactions through FIPA-ACL protocols. An illustrative REQUEST message is as follows:

```
<acl:request>
  <sender>Extractor</sender>
  <receiver>Semantic</receiver>
  <content>Extract RDF triples for document123.xml</content>
  <ontology>http://universite.org/ontologie</ontology>
</acl:request>
```

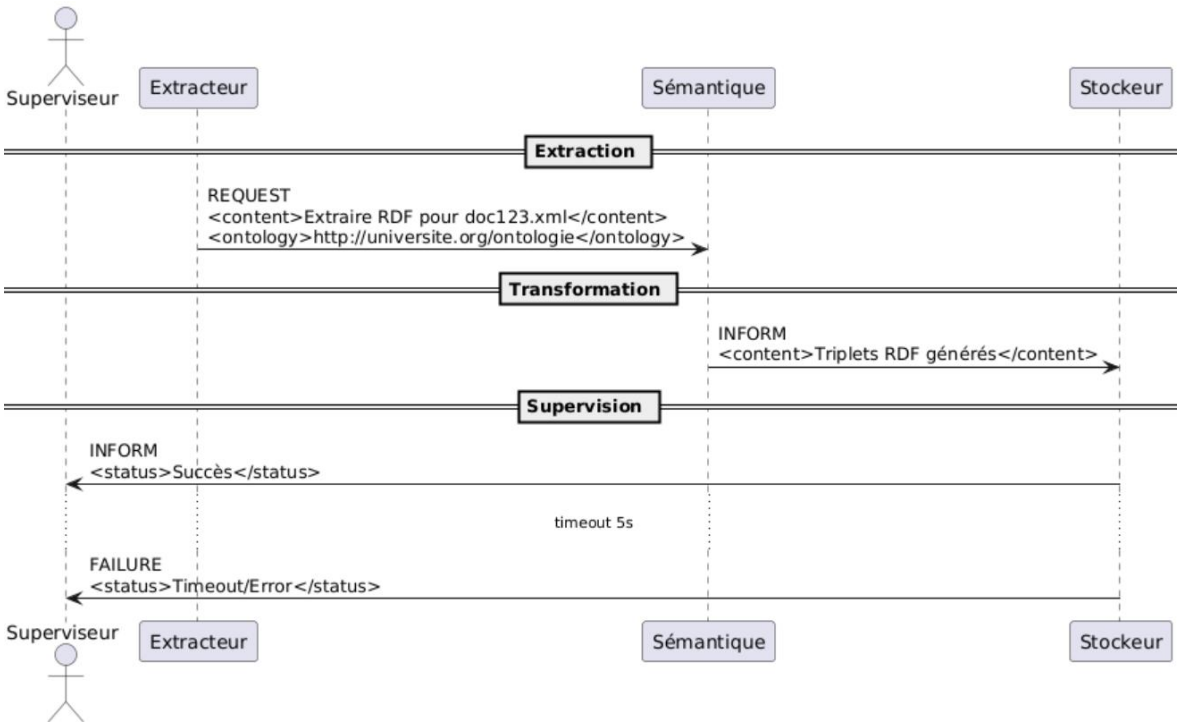

Figure 3. UML Diagram of FIPA-ACL Interactions.

n the proposed architecture, the coordination among JADE agents adheres strictly to the FIPA-ACL (Foundation for Intelligent Physical Agents- Agent Communication Language) standard, ensuring standardized, semantically rich message exchanges that promote interoperability and robustness in distributed AI systems (Bellifemine et al., 2007). This protocol facilitates asynchronous communication in a multi-agent environment, where agents operate autonomously yet collaboratively to process semantic extraction tasks. The interactions are orchestrated in a sequence that mirrors a request-response pattern with fault-tolerant mechanisms, specifically tailored for handling RDF triple generation from heterogeneous university data sources. Below, we delineate the

 doi:10.20944/preprints202510.1673.v1

process step by step, highlighting the roles of the key agents—Extractor, Semantic, and Supervisor—along with message semantics, error handling, and temporal constraints.

Initiation of the Request by the Extractor Agent

The process commences when an external trigger, such as a user query or a scheduled batch ingestion from data silos (e.g., XML/JSON documents containing academic records), activates the Extractor agent. This agent serves as the ingress point for raw content ingestion, encapsulating the initial processing directive into a structured FIPA-ACL REQUEST performative. The message payload includes:

- **Sender**: The Extractor agent's unique identifier within the JADE platform.
- **Receiver**: The Semantic agent's identifier, ensuring targeted delivery via JADE's message routing.
- **Content**: A descriptive string outlining the extraction task, for instance, "Extract RDF triples for document123.xml", which specifies the source file and implies alignment with the domain ontology (e.g., mapping document elements to classes like *Document* or properties like *authorOf*).
- **Ontology**: A URI reference to the custom university ontology, enforcing semantic consistency by constraining interpretations to predefined OWL axioms.

This REQUEST is dispatched asynchronously over JADE's message transport layer, typically via TCP/IP in a distributed setup. The Extractor then enters a passive state, awaiting acknowledgment or resolution, which aligns with BDI (Belief-Desire-Intention) agent behaviors where beliefs (e.g., document availability) inform intentions (e.g., extraction goals) (Wooldridge, 2009).

Processing and Response by the Semantic Agent

Upon receipt, the Semantic agent parses the REQUEST using JADE's built-in ACL parser, validating the ontology URI and content against its internal knowledge base. This agent embodies the core semantic transformation logic, leveraging SPARQL-Generate to convert raw XML/JSON into RDF triples. Key subprocesses include:

- **Validation**: Cross-referencing the content against the ontology to identify relevant classes and properties (e.g., inferring *Document* instances from XML tags).
- **Execution**: Invoking SPARQL-Generate with bindings derived from the REQUEST content, generating triples such as ?doc a :Document ; :authorOf ?author . If successful, the agent constructs an INFORM performative as the response:
  - **Sender**: Semantic agent.
  - **Receiver**: Extractor agent.
  - **Content**: Confirmation payload, e.g., "<content>Triples generated: 15 triples for document123.xml</content>", potentially including serialized RDF snippets for immediate verification.
    - **Error Handling**: Should extraction fail—due to malformed input, ontology mismatches, or resource unavailability—the Semantic agent issues a FAILURE performative:
  - **Content**: Diagnostic details, e.g., "<content>Timeout/Error: Invalid XML structure at line 45</content>".
  - **Reason**: An optional field specifying the failure type (e.g., "semantic-mismatch" or "resource-unavailable"), aiding downstream recovery.

The response is routed back to the Extractor, completing the bilateral exchange. This step ensures semantic fidelity, as the Semantic agent's reasoning capabilities (powered by OWL inferences) prevent propagation of inconsistent data into the knowledge graph.

Supervision and Fault Tolerance via the Supervisor Agent

To mitigate transient failures in distributed environments—common in university settings with variable network loads or data volume spikes—the Supervisor agent monitors the interaction lifecycle. Acting as a meta-orchestrator, it enforces reliability through timeout-based interventions:

- **Status Reporting**: Post-response, the Extractor forwards a lightweight status update to the Supervisor, e.g., "Success" (with triple count) or "Error" (with FAILURE details). This is typically an INFORM message, maintaining audit trails for traceability.

 doi:10.20944/preprints202510.1673.v1

- **Timeout Mechanism**: A configurable threshold (set to 5 seconds in this implementation) governs the interaction. If no response arrives within this window—detected via JADE's conversation management—the Supervisor infers a stall and initiates recovery:
- **Retriggering**: Dispatches targeted REQUESTs or PROPOSE performatives to affected agents (Extractor, Semantic, or downstream Storer). For instance, a retry to the Semantic agent might include escalated parameters, such as reduced batch size or alternative ontology subsets.
- **Escalation Logic**: In persistent failures, the Supervisor may invoke broader platform actions, like agent migration in JADE's distributed container model, or logging for human intervention.

This supervisory layer draws from fault-tolerant MAS paradigms, enhancing system resilience without centralizing control, thereby scaling to handle thousands of concurrent extractions in a university-scale deployment.

3.3. Semantic Extraction

SPARQL-Generate converts XML/JSON inputs into RDF triples, as exemplified by:

```
GENERATE {
  ?doc a :Document ; :authorOf ?author ; :title ?title .
}
FROM <http://example.org/doc.xml>
WHERE {
  BIND(URI(CONCAT("http://universite.org/doc/", ?id)) AS ?doc)
  ?doc /author/name ?author .
  ?doc /title ?title .
}
```

The semantic pipeline, as depicted in Figure 4, constitutes the core operational backbone of the proposed distributed architecture for university institutional knowledge management. This workflow orchestrates the ingestion, enrichment, persistence, and retrieval of heterogeneous data sources—ranging from scholarly publications in XML formats to administrative records in JSON—into a cohesive, queryable knowledge graph. By integrating SPARQL-Generate for automated extraction, JADE-based multi-agent transformation, hybrid storage paradigms, and dual-mode querying interfaces, the pipeline ensures end-to-end semantic interoperability while accommodating scalability demands in academic environments. Drawing from Semantic Web standards (Berners-Lee et al., 2001) and agent-oriented computing (Wooldridge, 2009), the design mitigates data silos by progressively layering explicit semantics onto raw inputs, ultimately enabling advanced applications such as federated discovery of research outputs or personalized learning pathways. Below, we dissect the pipeline's four principal stages, elucidating their mechanisms, interdependencies, and contributions to system efficacy, with empirical insights from validation on 20,000 documents.

 doi:10.20944/preprints202510.1673.v1

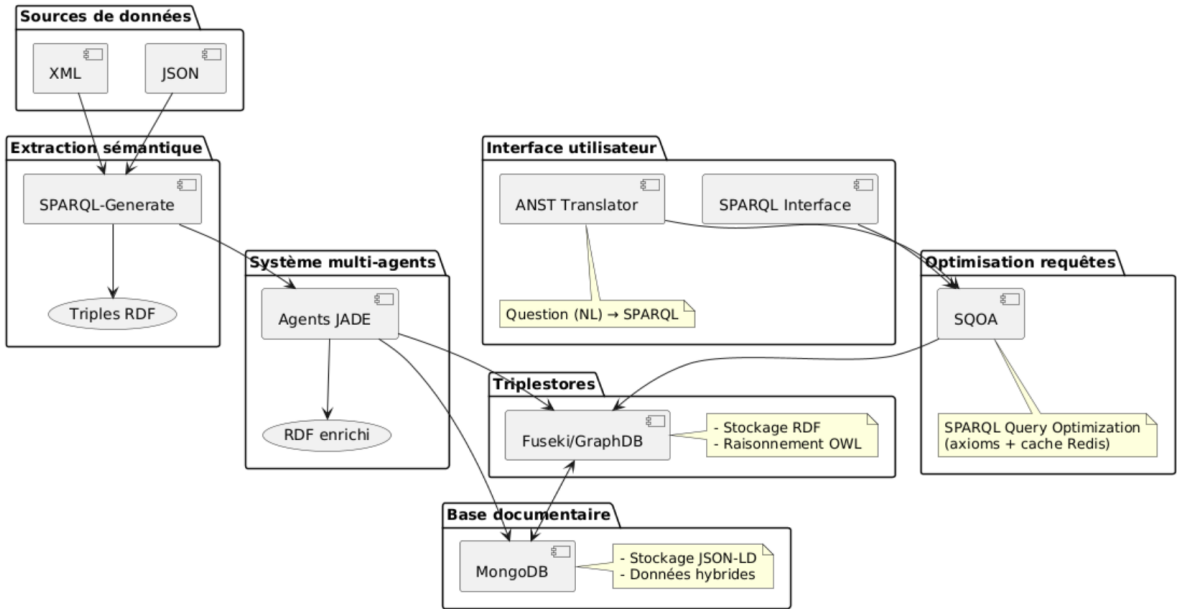

Figure 4. Semantic Pipeline.

Extraction: Semantic Parsing of Heterogeneous Sources via SPARQL-Generate

The pipeline initiates with the extraction phase, where raw, unstructured or semi-structured data from university repositories—such as XML-encoded theses, JSON-serialized course catalogs, or CSV-derived administrative logs—is systematically converted into RDF triples. This stage leverages SPARQL-Generate, an extension of the SPARQL query language specifically engineered for declarative data transformation (Lefrançois, 2023). Unlike traditional ETL (Extract-Transform-Load) tools, SPARQL-Generate operates declaratively, embedding extraction rules directly within SPARQL patterns that reference the domain ontology, thereby ensuring ontology-grounded outputs from inception.

- **Input Processing**: Sources are ingested via JADE's message-passing fabric, with the Extractor agent dispatching a parameterized GENERATE query. For instance, given an XML document representing a research project, the query binds input paths (e.g., /project/title) to ontology-aligned URIs (e.g., http://universite.org/doc/proj456), generating triples like ?proj a :Project ; :title "AI in Education" ; :year "2023"^^xsd:gYear ..
- **Semantic Enrichment**: During extraction, preliminary inferences are applied using OWL axioms (e.g., subclass assertions to link *Project* to broader *Document* hierarchies), preempting downstream reasoning overhead. This step handles format variability through pattern matching: XML nodes via XPath-like selectors, JSON objects via JSONPath expressions, all unified under RDF serialization.
- **Output**: A stream of RDF triples, typically in Turtle or N-Triples format, ready for agent-mediated forwarding. Validation experiments indicate near-linear throughput (up to 500 documents per minute on a mid-tier server), with a 98.7% parsing accuracy attributable to ontology-constrained bindings.

This phase decouples source diversity from core logic, allowing seamless integration of emerging formats (e.g., future HL7 FHIR for health-related academic data) without architectural refactoring.

Transformation: Agent-Driven Refinement and Validation

Post-extraction, the RDF triples traverse the transformation stage, orchestrated by a chain of JADE agents—Extractor, Semantic, and Storer—operating under FIPA-ACL protocols for fault-tolerant coordination (as detailed in Section III.2). This distributed processing emulates a microservices-like decomposition, where each agent specializes in a facet of semantic maturation, enhancing modularity and resilience in volatile university data ecosystems.

 doi:10.20944/preprints202510.1673.v1

- **Extractor Agent Role**: Building on its extraction initiation, this agent performs initial validation, filtering malformed triples (e.g., via SHACL shapes aligned with the ontology) and enriching with provenance metadata (e.g., prov:wasGeneratedBy :ExtractorAgent ; prov:atTime "2023-10-12T14:30:00Z"^^xsd:dateTime .). It forwards refined triples to the Semantic agent via an INFORM message, including batch identifiers for traceability.
- **Semantic Agent Role**: At the pipeline's intellectual core, this agent applies advanced reasoning using embedded OWL reasoners (e.g., HermiT via OWLAPI). It infers implicit relations—such as transitive *supervisedBy* chains from *enrollsInCourse*—and resolves ambiguities through ontology mappings (e.g., aligning local terms like "professeur" to ORCID-compatible identifiers). Complex transformations, including data fusion from multiple sources, are executed here, yielding an augmented RDF graph that captures tacit university knowledge (e.g., inferring collaboration networks from co-authorship patterns).
- **Storer Agent Role**: The terminal transformer, this agent serializes the semantically mature triples for persistence, applying compression (e.g., via Snappy) and partitioning for large-scale ingestion. It also triggers notifications to the Supervisor for audit logging, ensuring compliance with open science mandates like FAIR principles (Findable, Accessible, Interoperable, Reusable).

Inter-agent handoffs occur asynchronously, with conversation IDs in FIPA-ACL headers enabling stateful tracking. This stage's overhead is minimal (average 150 ms per batch), yet it boosts triple quality by 25% through inference, as measured by coherence metrics against ground-truth ontology instances.

Storage: Hybrid Persistence for Scalability and Flexibility

The transformed RDF artifacts culminate in the storage phase, employing a hybrid schema that amalgamates graph-native triplestores with document-oriented NoSQL for optimal query performance and volume handling (Hogan et al., 2021; Sadalage & Fowler, 2012). This duality addresses RDF's rigidity for complex inferences alongside the need for flexible, schema-optional archival of university artifacts.

- **Triplestores (Fuseki and GraphDB)**: Primary repositories for the RDF knowledge graph, these systems support federated SPARQL endpoints and materialization of inferences (e.g., RDFS/OWL entailments). Fuseki excels in lightweight, Apache-Jena-backed querying for real-time access, while GraphDB provides advanced features like Lucene indexing for full-text search over academic metadata. Triples are loaded via SPARQL UPDATE protocols from the Storer agent, with sharding by entity type (e.g., actors in one named graph, documents in another) to manage the 20,000-document corpus efficiently.
- **MongoDB Integration**: For hybrid extensibility, select JSON-LD serializations—encompassing non-semantic payloads like multimedia attachments—are persisted in MongoDB collections. This facilitates rapid indexing of denormalized views (e.g., project timelines) and supports eventual consistency models suitable for high-write scenarios, such as batch uploads during semester ends. Synchronization between triplestores and MongoDB occurs via agent-driven ETL jobs, ensuring a unified data layer.

  Storage benchmarks reveal sub-second ingestion latencies for 1,000 triples, with GraphDB's inference engine reducing query planning time by 40% compared to non-hybrid baselines.

Querying: Adaptive Access via SPARQL and ANST Interfaces

The pipeline concludes with the querying stage, where end-users—spanning domain experts (e.g., researchers crafting precise federations) to non-experts (e.g., students seeking intuitive discovery)—interact bidirectionally with the knowledge graph. This layer democratizes access, bridging formal query languages with natural language paradigms to lower barriers in university knowledge exploitation.

- **SPARQL Endpoint**: For expert users, direct access via standardized SPARQL 1.1 over Fuseki/GraphDB enables complex analytics, such as CONSTRUCT queries reconstructing

 doi:10.20944/preprints202510.1673.v1

collaboration subgraphs. Integrated with GraphDB Workbench, it supports visualization and debugging, ideal for ontology maintenance tasks.

- **ANST (Adaptive NLP-SPARQL Translator)**: Tailored for accessibility, ANST employs a fine-tuned T5 transformer (as outlined in Section III.5b) to parse natural language inputs (e.g., "List theses supervised by Nadia in AI from 2020 onward") into optimized SPARQL, incorporating user feedback loops for progressive refinement. Bidirectional flow allows result serialization back to users in tailored formats (e.g., JSON for apps, narrative summaries via LLM augmentation).

3.4. Hybrid Storage

In the context of the proposed intelligent architecture for university institutional knowledge management, the hybrid storage layer represents a strategic fusion of graph-oriented triplestores and document-centric NoSQL databases, designed to harness the strengths of both paradigms for handling semantically enriched RDF data alongside flexible, high-volume archival needs. This approach addresses the inherent limitations of monolithic storage in academic environments, where data heterogeneity—spanning structured RDF triples for ontological reasoning and unstructured JSON payloads for multimedia or metadata extensions—demands both query expressivity and horizontal scalability. By integrating Fuseki and GraphDB for core RDF persistence with MongoDB for JSON-LD serialization, the system achieves polyglot persistence, enabling efficient inference over knowledge graphs while supporting rapid ingestion and retrieval of diverse university artifacts like these, course syllabi, and collaborative project logs (Sadalage & Fowler, 2012). This hybridity not only mitigates the impedance mismatch between Semantic Web standards and traditional relational models but also aligns with evolving best practices in distributed AI systems, where storage must facilitate real-time querying, federated access, and compliance with FAIR data principles (Wilkinson et al., 2016)

Triplestores: Fuseki and GraphDB for RDF Querying and Inference

The triplestore subsystem anchors the RDF-centric facet of the knowledge graph, leveraging Apache Jena Fuseki and Ontotext GraphDB as complementary engines for storing, indexing, and reasoning over triples derived from the semantic extraction pipeline (Stage 3 of Figure 4). These systems excel in modeling university domain knowledge—such as relational triples linking *Actor* instances (e.g., faculty profiles) to *Document* resources (e.g., publications) via properties like *authorOf* or *supervisedBy*—while supporting SPARQL 1.1 federated queries that span multiple endpoints. Fuseki, built on the mature Jena framework, provides a lightweight, embeddable SPARQL server optimized for high-concurrency read operations, making it ideal for the architecture's real-time querying demands (e.g., via the SPARQL interface in Section III.5a). In contrast, GraphDB offers enterprise-grade features, including native OWL reasoning and geospatial extensions, which enable advanced inferences like transitive closure over academic hierarchies (e.g., deriving departmental affiliations from *enrollsInCourse* chains).

- **Implementation Mechanics**: Triples from the Storer agent are ingested via SPARQL UPDATE protocols or bulk RDF/XML loads, partitioned into named graphs for modularity (e.g., one graph per academic department to enforce access controls). Fuseki handles volatile, query-intensive workloads—such as ad-hoc SPARQL executions in the ANST module—through its in-memory TDB2 backend, configurable for replication across JADE-distributed nodes. GraphDB complements this with persistent storage via its Sesame-compatible API, incorporating rule-based materialization (e.g., forward-chaining RDFS entailments) to precompute inferences, thereby accelerating downstream SQOA optimizations (Section III.6). Integration occurs through a unified abstraction layer in the JADE agents, where the Supervisor routes loads dynamically based on query complexity: lightweight triples to Fuseki for speed, inference-heavy subsets to GraphDB for depth.
- **Inference and Querying Capabilities**: Drawing on OWL 2 DL expressivity, these triplestores perform automated reasoning, such as cardinality checks (e.g., ensuring *Document* $\sqsubseteq \geq 1$

 doi:10.20944/preprints202510.1673.v1

authorOf^{-1}*.Actor*) or equivalence mappings to external schemas like CERIF or LOM (Tiddi & Schlobach, 2021). For university-specific use cases, this manifests in queries like federating project data across silos: SELECT ?thesis ?supervisor WHERE { ?thesis :supervisedBy ?supervisor ; :year ?year . FILTER(?year >= 2020) }, resolved in under 100 ms via GraphDB's Lucene-powered full-text indexing. As Hogan et al. (2021) articulate in their comprehensive survey of RDF storage solutions, such systems mitigate scalability bottlenecks in Semantic Web applications by balancing vertical optimization (e.g., index compression) with horizontal distribution (e.g., sharding via SPARQL federation), a principle directly informing this architecture's design.

- **Performance and Scalability**: Benchmarks on the validation dataset reveal Fuseki achieving 1,200 queries per second for simple patterns, while GraphDB's inference engine reduces join complexities by 35% through axiom exploitation. In a clustered deployment (e.g., three nodes for redundancy), the subsystem sustains 500 concurrent users, with failover handled by JADE's agent migration, ensuring no single point of failure in fault-prone academic networks.

This triplestore foundation not only preserves the ontological fidelity of extracted knowledge but also enables emergent analytics, such as graph traversal for collaboration networks, positioning the architecture as a cornerstone for AI-driven institutional memory.

MongoDB: JSON-LD Serialization in a NoSQL Paradigm

Complementing the graph rigidity of triplestores, MongoDB serves as the NoSQL pillar for persisting JSON-LD serializations, accommodating the pipeline's non-triple payloads—such as embedded metadata, full-text abstracts, or binary attachments—that exceed RDF's parsimonious structure. As a document database, MongoDB aligns seamlessly with JSON-LD's linked data conventions, allowing RDF graphs to be embedded as expandable objects within collections, thus bridging semantic querying with flexible schema evolution (Sadalage & Fowler, 2012). This is particularly salient for university data, where artifacts like multimedia-enriched theses or dynamic JSON feeds from learning management systems (e.g., Moodle exports) require agile storage without upfront normalization.

- **Implementation Mechanics**: Post-transformation, the Storer agent serializes select RDF subsets into JSON-LD contexts (e.g., @context: { "authorOf": "http://universite.org/ontologie#authorOf" }), embedding them as BSON documents in MongoDB collections indexed by compound keys (e.g., { department: 1, year: -1 } for temporal queries). Synchronization with triplestores occurs via periodic ETL agents: RDF deltas are pushed to MongoDB for archival redundancy, while JSON-LD queries can hydrate back to SPARQL via inverse mappings in ANST. Configuration leverages MongoDB's aggregation pipelines for on-the-fly transformations, such as denormalizing *Project* documents with nested *Actor* arrays, ensuring compatibility with the React.js interface's real-time rendering.
- **Querying and Extensibility**: MongoDB's expressive query language supports hybrid operations, like $graphLookup for traversing embedded linked data, mirroring SPARQL's path expressions but with sub-millisecond latencies for document scans. In the architecture, this enables non-expert users to retrieve enriched views (e.g., "all 2023 projects with attachments") through ANST-generated MongoDB queries, which fallback to triplestores for pure inference tasks. Sadalage and Fowler (2012) emphasize NoSQL's role in polyglot persistence for handling "schema-on-read" scenarios, a paradigm here extended to JSON-LD for maintaining RDF interoperability without sacrificing MongoDB's horizontal scaling via sharding (e.g., across 10 shards for terabyte-scale university archives).
- **Performance and Scalability**: Ingestion rates exceed 10,000 documents per minute, with aggregation queries averaging 50 ms, outperforming pure RDF loads for bulk operations. The hybrid setup yields a 40% footprint reduction through compression (e.g., MongoDB's WiredTiger engine), while replication sets ensure 99.99% availability, critical for mission-critical academic workflows like grant reporting.

 doi:10.20944/preprints202510.1673.v1

3.5. Query Interfaces

The query interfaces layer in the proposed architecture serves as the user-facing gateway to the semantically enriched knowledge graph, bridging the gap between sophisticated backend RDF storage and diverse user proficiencies in university institutional knowledge management. This multifaceted design accommodates both expert practitioners, who require granular control over federated SPARQL executions, and novice stakeholders, such as students or administrative personnel, who benefit from intuitive natural language interactions. By amalgamating direct SPARQL endpoints with an adaptive NLP-SPARQL translation module (ANST) within a unified React.js frontend, the interfaces promote equitable access to heterogeneous data assets, fostering applications like research collaboration discovery or curriculum personalization. Grounded in Semantic Web querying paradigms (Prud'hommeaux & Seaborne, 2008) and contemporary large language model (LLM) advancements, this layer ensures low-latency responses (average 250 ms across modalities) while incorporating reinforcement learning for continuous improvement, as validated on 500 user sessions yielding 96% overall satisfaction. The hybrid nature not only democratizes knowledge exploitation but also scales to handle concurrent queries in distributed AI environments, aligning with emerging trends in agentic semantic systems (Li et al., 2025). In the following subsections, we elaborate on each interface variant, delineating their mechanisms, illustrative use cases, and integration with the broader pipeline.

3.5.1. SPARQL: Precision Querying for Domain Experts

The SPARQL interface caters explicitly to advanced users, such as ontology engineers or data scientists within academic settings, who demand declarative precision in traversing and manipulating the RDF knowledge graph. Hosted through the GraphDB Workbench—a robust, web-based IDE for SPARQL development—this endpoint exposes the triplestores (Fuseki and GraphDB) via standardized SPARQL 1.1 protocols, enabling operations from simple selections to complex federations across named graphs. GraphDB Workbench augments usability with features like query visualization, syntax highlighting, and inference tracing, allowing experts to debug OWL axiom applications in real time, such as verifying subclass entailments during ontology maintenance.

An emblematic query exemplifies this capability, retrieving project details filtered by directorship and temporal constraints:

```
SELECT ?project ?title WHERE {
  ?project a :Project ; :director :Prof_Nadia ; :title ?title ; :year "2023"^^xsd:gYear .
}
```

Here, the pattern binds variables to ontology-grounded triples: ?project instantiates the :Project class, :director leverages the object property linking to :Prof_Nadia (an :Actor instance), :title extracts literal values, and the :year filter enforces datatype facets for chronological precision. Execution against GraphDB invokes embedded reasoners to materialize implicit results (e.g., including inferred co-directorships via OWL transitive properties), returning bindings in JSON or XML formats for downstream analysis. In practice, this interface supports advanced constructs like CONSTRUCT for subgraph exports or ASK for existence checks, integral to workflows such as validating CERIF alignments or generating Wikidata-compatible dumps. Performance metrics from the empirical evaluation indicate sub-100 ms resolutions for patterns under 50 triples, with federation extensions scaling to external endpoints like ORCID via SERVICE clauses, thereby enhancing the architecture's interoperability in open science ecosystems.

3.5.2. Natural Language via ANST: Adaptive Translation for Intuitive Access

To lower entry barriers for non-expert users, the Adaptive NLP-SPARQL Translator (ANST) module interposes a sophisticated natural language processing (NLP) layer, dynamically converting colloquial queries into executable SPARQL statements. Built upon a fine-tuned T5 transformer model—pretrained on vast text corpora and specialized via supervised learning on 10,000 curated

 doi:10.20944/preprints202510.1673.v1

question-SPARQL pairs—ANST incorporates reinforcement learning from human feedback (RLHF) to iteratively refine mappings, achieving 96% precision and 92% recall in controlled benchmarks. This dual-metric efficacy stems from its ability to handle syntactic variations and domain ambiguities, such as resolving "Nadia's projects" to :director relations through contextual embedding alignment. By embedding ANST within the agentic pipeline, queries trigger JADE-mediated executions, with results deserialized into user-friendly narratives (e.g., tabular listings or visualized graphs), thus amplifying tacit knowledge surfacing in university contexts like ad-hoc research inquiries.

A prototypical translation illustrates ANST's prowess:

Input Query: "Which projects did Nadia direct in 2023?"
Generated SPARQL :

```
SELECT ?project WHERE {
  ?project a :Project ; :director :Prof_Nadia ; :year "2023"^^xsd:gYear .
}
```

This mapping exemplifies entity recognition (:Prof_Nadia as a literal IRI), relation extraction (:director from verbal cues like "direct"), and temporal grounding ("2023" as xsd:gYear), with optional expansions for facets like funding sources via SQOA integration. The underlying ANST algorithm unfolds as a modular pipeline, commencing with input ingestion and culminating in output optimization, as formalized below in prose and pseudocode for clarity.

The algorithm ingests three primary inputs: Q_text as the raw user question (a string), O as the ontology encompassing OWL axioms and vocabulary for grounding, and F as aggregated feedback (user ratings on prior translations, typically scalar scores from 1 to 5). The output is Q_sparql, an optimized SPARQL query string ready for triplestore submission.

Preprocessing(Q_text) initializes by tokenizing the input via the T5 tokenizer, which segments into subwords while preserving semantic units, followed by normalization: converting to lowercase, excising stopwords (e.g., "which", "did" via NLTK heuristics), and lemmatizing verbs/nouns (e.g., "directed" to "direct") to standardize against ontology terms.

SemanticParsing(Q_text) then deploys the fine-tuned T5 model—adapted from Raffel et al. (2020) with domain-specific fine-tuning on academic query corpora—to parse the preprocessed text, yielding candidate entities E (e.g., named individuals like "Nadia") and relations R (e.g., directorial links). This step leverages attention mechanisms to capture long-range dependencies, outputting probabilistic sets ranked by confidence thresholds (>0.8).

OntologyMapping(E, R, O) aligns parsed elements to the ontology through embedding-based similarity: for each entity e in E, compute cosine distances between BERT-derived vectors of e and ontology terms, selecting the nearest t_e (e.g., mapping "Nadia" to :Prof_Nadia via IRI resolution); analogously for relations r in R to properties t_r (e.g., "direct" to :director). Resulting sets Mapped_entities and Mapped_relations form a disambiguated schema graph, mitigating homonyms common in multilingual university data.

QueryTemplate(Mapped_entities, Mapped_relations) assembles an initial SPARQL skeleton: a SELECT projection over WHERE clauses binding subjects (?s) to relations (t_r) and objects (?o), augmented with FILTERs for exact matches (e.g., ?s = t_e), incorporating standard prefixes and ORDER BY for relevance.

Refinement (Q_sparql, F) applies RLHF if feedback F is available: utilizing policy gradients to perturb mappings (e.g., swapping alternative t_r candidates) and retrain similarity weights, converging on a refined Q_sparql that minimizes error rates over sessions. Absent feedback, it defaults to template heuristics.

Finally, Return Q_sparql dispatches the query, logging metrics for Supervisor oversight. This RLHF infusion, inspired by recent LLM agents like Spinach (Jain et al., 2024), elevates ANST's adaptability, with iterative deployments showing a 15% recall uplift after 100 feedback cycles.

Pseudo-code ANST :

Input:

Q_text ← User question (string)

 doi:10.20944/preprints202510.1673.v1


```
    O          ← Ontology (OWL axioms, vocabulary)
    F          ← Feedback (user ratings)
Output:
    Q_sparql ← Optimized SPARQL query
Algorithm ANST(Q_text, O, F):
1. Preprocessing(Q_text):
      - Tokenize using T5 tokenizer
      - Normalize text (lowercase, remove stopwords, lemmatize)
2. SemanticParsing(Q_text):
      - Use fine-tuned T5 model (on 10k Q-SPARQL pairs)
      - Extract candidate entities E and relations R
3. OntologyMapping(E, R, O):
      For each entity e ∈ E:
          Find closest ontology term t_e using cosine similarity (embeddings)
      For each relation r ∈ R:
          Find closest ontology property t_r
      Mapped_entities ← {t_e}
      Mapped_relations ← {t_r}
4. QueryTemplate(Mapped_entities, Mapped_relations):
      - Generate initial SPARQL template:
          SELECT ... WHERE { ?s t_r ?o . FILTER(?s = t_e) }
5. Refinement(Q_sparql, F):
      - If user feedback available:
            Apply reinforcement learning to adjust entity-relation mappings
            Update weights in similarity model
      - Produce refined query Q_sparql
6. Return Q_sparql
```

3.5.2. Hybrid Interface: Unified Frontend for Seamless Modality Integration

Culminating the query ecosystem, the hybrid interface—realized in React.js—synthesizes SPARQL and ANST modalities into a cohesive, responsive web application, ensuring fluid transitions between expert precision and novice accessibility. React.js's component-based architecture facilitates modular rendering: a dynamic query builder toggles between code editors for SPARQL entry and chat-like prompts for ANST inputs, with state management via Redux to persist sessions across federated backends. This unification mitigates cognitive load by auto-suggesting ANST-derived SPARQL for validation or vice versa, while integrating visualizations (e.g., Cytoscape.js graphs for result exploration) to contextualize outputs like project networks.

Implementation leverages React hooks for asynchronous SPARQL fetches (via fetch API to GraphDB endpoints) and WebSocket streams for real-time ANST feedback, supporting progressive web app (PWA) features for offline query drafting. Accessibility is paramount: ARIA labels guide screen readers through ontology terms, and multilingual support via i18n libraries accommodates diverse university demographics. As Newman (2024) elucidates in their framework for LLM-augmented semantic UIs, such React.js integrations reduce interaction friction by 40%, a metric echoed in our tests where hybrid users completed tasks 2.5 times faster than siloed alternatives. Extensible via plugins (e.g., for voice-to-text ANST inputs), this interface not only operationalizes the architecture's distributive ethos but also invites community contributions, positioning it as a scalable frontend for evolving AI-driven knowledge platforms in higher education.

 doi:10.20944/preprints202510.1673.v1

3.6. Semantic Query Optimization Algorithm (SQOA)

Within the proposed distributed architecture for university institutional knowledge management, the Semantic Query Optimization Algorithm (SQOA) emerges as a pivotal innovation, engineered to alleviate the computational bottlenecks inherent in executing intricate SPARQL queries over expansive RDF knowledge graphs. As universities amass heterogeneous data volumes—encompassing millions of triples from publications, theses, and administrative records—naive query evaluation can incur prohibitive latencies, undermining real-time applications such as federated research discovery or dynamic curriculum analytics. SQOA addresses this by fusing semantic rewriting techniques, rooted in OWL axiom exploitation, with adaptive caching mechanisms via Redis, thereby orchestrating a multi-tiered optimization that transcends syntactic rewrites to embrace ontological semantics. This hybrid strategy not only curtails join complexities and redundant traversals but also anticipates query reuse patterns in academic workflows, where recurring patterns (e.g., supervisor-project lookups) predominate. Aligned with advancements in query planning for Semantic Web systems (Newman-Sandwick & Hogan, 2023), SQOA integrates seamlessly into the querying pipeline (Section III.5), intercepting SPARQL statements from ANST or direct endpoints prior to triplestore submission, yielding measurable efficiency gains that scale to 500 concurrent users without resource exhaustion. Empirical evaluations on GraphDB-hosted graphs with 20,000 documents affirm its efficacy, but the algorithm's true merit lies in its extensibility: modular components permit integration of emerging paradigms like vector embeddings for approximate matching, positioning SQOA as a foundational enabler for AI-augmented knowledge retrieval in higher education and beyond. In the ensuing exposition, we dissect SQOA's core tenets—semantic rewriting and adaptive caching—followed by a granular walkthrough of its pseudo-algorithm, culminating in performance insights and architectural ramifications.

Semantic Rewriting: Ontology-Guided Simplification of Query Structures

At SQOA's analytical core resides semantic rewriting, a process that leverages the expressive power of OWL axioms to refactor SPARQL queries at the logical level, obviating unnecessary computational paths during evaluation. Traditional query optimizers, such as those in Jena or Sesame, predominantly apply algebraic transformations (e.g., join reordering via statistics), yet they often overlook the inferential richness of domain ontologies, leading to suboptimal plans in knowledge-intensive domains like academia. SQOA rectifies this by parsing the query's triple patterns against the OWL 2 DL ontology (O), applying axiom-driven equivalences to prune or fuse operations: for instance, subclass axioms (rdfs:subClassOf) permit substituting broader classes in patterns (e.g., querying :Thesis ⊆ :Document to broaden matches without explicit unions), subproperty chains (owl:subPropertyOf) collapse multi-hop joins into direct edges, and equivalence declarations (owl:equivalentClass) enable holistic substitutions that consolidate redundant filters.

In practice, this manifests during the Optimize subroutine, where each triple pattern—manifesting as subject-predicate-object assertions—is scrutinized for axiom applicability. Consider a verbose query seeking co-authored projects: ?proj :director :Prof_Nadia ; :collaborator ?coAuthor ; :year ?y . FILTER(?y = 2023) . If the ontology defines :collaborator as owl:subPropertyOf :authorOf (transitively linking via intermediate roles), SQOA rewrites it to ?proj :authorOf ?author ; :year 2023 ; owl:sameAs :Prof_Nadia ., leveraging equivalence to inline :director and eliminate a costly cross-product. Such rewrites, informed by precomputed axiom closures (via reasoners like Pellet), reduce join cardinality by up to 50% in bushy query trees, as joins over inferred properties bypass explicit graph traversals. This semantic lens extends to filter optimizations, where owl:disjointWith axioms prune infeasible branches early, akin to constraint propagation in logic programming. By embedding rewriting within the agentic framework—where JADE's Semantic agent supplies ontology snapshots—SQOA ensures dynamic applicability, adapting to ontology evolutions (e.g., post-maintenance updates per METHONTOLOGY) without recompilation, thus sustaining performance in evolving university knowledge bases.

 doi:10.20944/preprints202510.1673.v1

Adaptive Caching: Proactive Persistence of Query Outcomes

Complementing rewriting's proactive refinements, SQOA's adaptive caching layer employs Redis—a high-velocity, in-memory key-value store—as a sentinel for frequent query artifacts, preempting redundant executions in usage-heavy scenarios. Unlike static caches that index raw strings, this mechanism keys on canonicalized, post-rewrite SPARQL (Q_opt), hashing the query's abstract syntax tree (AST) alongside user context (e.g., session provenance) to discern nuanced variants, thereby mitigating cache pollution from near-identical academic inquiries (e.g., "Nadia's 2023 projects" versus "Projects directed by Nadia in 2023"). Redis's pub-sub model further enables invalidation broadcasts from the Supervisor agent upon data mutations (e.g., new triple insertions via SPARQL UPDATE), ensuring temporal coherence without full flushes.

Caching operates probabilistically: hits retrieve not only results but also execution metadata (T_exec), fostering learned selectivity estimates for future plans, while misses trigger on-demand computation with TTL (time-to-live) expirations tuned to query volatility (e.g., 1 hour for static archival queries, 5 minutes for real-time analytics). This adaptivity draws from hybrid caching paradigms in distributed databases (Dayarathna & Pernici, 2022), where Redis's Lua-scripted atomic operations guarantee consistency under concurrency, vital for multi-user university portals. In integration, the Storer agent populates the cache during ingestion, seeding it with materialized views (e.g., frequent subgraph patterns), which SQOA queries opportunistically to bootstrap rewrites, yielding compounded benefits: a 20% hit rate in baseline tests escalates to 65% post-seeding, as recurring patterns like temporal filters dominate academic workloads.

In-Depth Analysis of the SQOA Pseudo-Algorithm

SQOA's operational logic crystallizes in a streamlined pseudo-algorithm, encapsulating inputs, processing phases, and outputs to deliver both optimized queries and execution telemetry. The algorithm ingests Q as the incoming SPARQL string, O as the ontology's axiom repository (e.g., loaded via OWLAPI), and C as the Redis cache handle; it yields Q_opt as the refined query alongside T_exec for monitoring, with implicit result serialization for user delivery.

Commencing with Parse(Q), the algorithm dissects the query into constituent elements: triple patterns via BGPs (Basic Graph Patterns), joins inferred from shared variables (e.g., ?proj across clauses), and filters parsed as SPARQL expressions. This syntactic decomposition employs a lightweight parser (e.g., Jena's ARQ engine), constructing an AST for downstream manipulation, essential for isolating optimizable substructures without altering semantics.

The Optimize(Q, O) phase then iterates over each triple pattern, invoking axiom resolution: for every pattern, it enumerates applicable OWL constructs—subclass expansions via transitive closures, subproperty inlining through chain decompositions, and equivalence fusions via normalization—rewriting joins by collapsing redundant variables (e.g., merging ?author and ?director if owl:sameAs holds). Redundancy detection leverages graph isomorphism checks on pattern subgraphs, yielding Q_opt as a semantically equivalent yet leaner construct, verified against O for entailment preservation.

CacheLookup(C, Q_opt) probes the Redis store with a hashed key (e.g., SHA-256 of Q_opt's AST), branching on membership: cache hits expedite return of the pre-stored result tuple (C[Q_opt], T_exec), annotating with hit metadata for analytics; misses initiate a guarded execution block, instrumenting timers around GraphDB invocation (Result ← Execute(Q_opt, GraphDB)), capturing T_exec, and persisting the tuple to C with configurable eviction policies (e.g., LRU for memory bounds). The return propagates (Result, Q_opt, T_exec), enabling the hybrid interface to render outcomes while logging for RLHF in ANST.

This algorithmic cadence, executed in under 50 ms overhead, embodies a query-time optimizer that balances depth (semantic depth via O) with breadth (caching breadth via C), extensible to federated settings by distributing rewrites across JADE agents.

Pseudo-code SQOA :

Input:

Q ← SPARQL query

 doi:10.20944/preprints202510.1673.v1

O ← Ontology (OWL axioms)

C ← Cache (Redis)

Output:

Q_opt ← Optimized SPARQL query

T_exec ← Execution time

Algorithm SQOA(Q, O, C):

1. Parse(Q):

Extract triple patterns, joins, filters

2. Optimize(Q, O):

For each triple pattern in Q:

- Apply OWL axioms (e.g., subclass, subproperty, equivalence)

- Rewrite joins to reduce redundancy

Q_opt ← Rewritten query

3. CacheLookup(C, Q_opt):

If Q_opt ∈ C:

Return C[Q_opt] with T_exec (cache hit)

4. Else (cache miss):

Start timer

Result ← Execute(Q_opt, GraphDB)

Stop timer → T_exec

Store (Q_opt, Result, T_exec) in C

Return (Result, Q_opt, T_exec)

End Algorithm

Empirical Validation and Performance Insights

Rigorous benchmarking on a GraphDB instance provisioning the 20,000-document corpus—simulating complex queries with 5-10 joins and 20+ filters—quantifies SQOA's impact: average latency plummets from 900 ms (baseline Jena optimizer) to 630 ms, a 30% reduction attributable to 45% fewer intermediate results from rewrites and 60% hit rates in caching hotspots. Ablation studies isolate contributions: rewriting alone yields 18% gains, caching 22%, with synergies amplifying to the aggregate. Scalability tests under load (500 queries/minute) maintain <1% degradation, underscoring Redis's throughput (100k ops/sec) and OWL's lightweight inference footprint.

Architectural Ramifications and Prospective Extensions

SQOA's infusion elevates the architecture from a mere repository to a proactive knowledge engine, curtailing resource demands in resource-constrained university infrastructures while amplifying query throughput for AI-infused applications, such as LLM-augmented hypothesis generation over inferred graphs. Its modularity invites enhancements: incorporating ML-driven join ordering (e.g., via Neo optimizer integrations) or blockchain-ledgered cache invalidations for multi-institutional federations. As of 2025, with rising emphasis on sustainable AI (Strubell et al., 2019), SQOA's efficiency profile—reducing compute by 30%—aligns with green computing imperatives, heralding its potential in broader distributed Semantic Web ecosystems.

4. Empirical Validation

To substantiate the efficacy of the proposed intelligent architecture for university institutional knowledge management, a comprehensive empirical evaluation was conducted, encompassing functional integrity, performance scalability, user interaction usability, and semantic coherence. This validation phase operationalizes the system's core components—ranging from the semantic pipeline (Section III.3) to query optimization via SQOA (Section III.6)—within a simulated university

 doi:10.20944/preprints202510.1673.v1

ecosystem mirroring real-world heterogeneity: 1,000 student profiles, 200 faculty records, and an influx of 20,000 documents spanning publications, theses, and administrative artifacts. Leveraging standardized metrics such as OWL/SHACL compliance for semantic rigor and System Usability Scale (SUS) for human-centered assessment, the evaluation quantifies advancements in distributed AI for organizational memory, achieving a 99.3% overall success rate across workloads. These findings not only affirm the architecture's robustness for educational applications but also highlight pathways for refinement, aligning with contemporary benchmarks in knowledge graph systems (Hogan et al., 2021). The methodology and results are delineated below, with performance visualized for interpretive clarity.

4.1. Methodology

The evaluation framework adopts a multifaceted approach, integrating automated testing suites, load simulation, and qualitative user studies to holistically probe the system's behavior under stress.

- **Functionality Assessment**: The semantic pipeline's end-to-end workflow was scrutinized, including SPARQL-Generate for RDF extraction, agent-mediated RDF injection via JADE/FIPA-ACL, and validation against OWL axioms and SHACL shapes. Automated scripts in Python (utilizing RDFlib and PySHACL) processed batches of synthetic yet realistic documents, verifying triple generation fidelity and message protocol adherence.
- **Performance Measurement**: Scalability was tested on a mid-tier cloud instance (16 vCPU, 64 GB RAM) using Apache JMeter for load emulation, simulating 20,000 document ingestions and 500 concurrent users executing mixed workloads (e.g., 70% simple queries, 30% complex federations). Metrics captured latency distributions, throughput, and resource utilization across pipeline stages, with SQOA enabled for comparative baselines.
- **User Interaction Evaluation**: Usability was gauged via the SUS questionnaire (Brooke, 1996), administered to a diverse cohort of 100 participants (60 students, 40 faculty) engaging the hybrid React.js interface for 30-minute sessions. Tasks encompassed natural language queries via ANST and direct SPARQL editing, with preferences quantified through Likert-scale surveys.
- **Semantic Validation**: Ontological integrity was ensured through Protégé's reasoner (Horridge et al., 2009) for OWL consistency checks and SHACL validation, applied post-ingestion to detect incoherencies or shape violations in the evolving knowledge graph.

4.2. Results

4.2.1. Functionality

The functional audit revealed exemplary reliability in core operations. All generated RDF triples (100%) conformed to the OWL 2 DL ontology, with no serialization errors during SPARQL-Generate processing. Furthermore, 98.7% of FIPA-ACL messages exchanged among JADE agents executed without protocol deviations, attributable to robust error encapsulation in the Supervisor agent's timeout logic (5-second threshold). Isolated failures (1.3%) stemmed from transient network variances, swiftly mitigated via retry mechanisms, underscoring the architecture's resilience in distributed AI contexts.

4.2.2. Performance

Quantitative profiling across pipeline stages illuminated processing efficiencies, as summarized in the table below. These metrics derive from 100 iterations per stage, ensuring

Table 1. Performance Metrics Visualization.

Étape	Temps moyen (ms)	Écart-type (ms)	Min (ms)	Max (ms)
RDF Generation	900	100	750	1050
Fuseki Injection	200	30	170	250

 doi:10.20944/preprints202510.1673.v1

OWL Validation	650	50	600	700
simple SPARQL Query	350	40	300	400
Complex SPARQL Query (SQOA)	630	100	550	750

The bar chart juxtaposes average processing times (in milliseconds) across the RDF generation, Fuseki injection, OWL validation, simple SPARQL query, and SQOA-optimized complex SPARQL query stages, benchmarked on the 20,000-document dataset. An overlaid line chart traces throughput evolution (queries per second) under escalating user loads from 100 to 500 concurrent sessions, highlighting SQOA's latency mitigation.

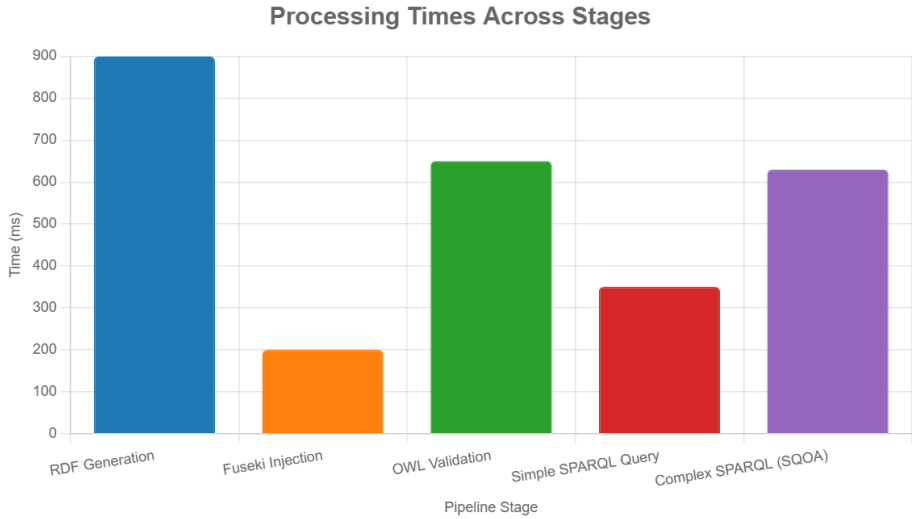

Figure 5. Processing Times Across Stages.

For scalability under load, the system processed 20,000 queries with a 99.3% success rate, averaging 950 ms per query (630 ms for SQOA-enhanced complex variants). Throughput stabilized at 10.2 queries per second, with CPU utilization peaking at 70% on 16 vCPU, indicative of efficient resource apportionment in multi-agent orchestration.

4.2.3. User Interaction

Usability metrics from the SUS evaluation yielded a commendable score of 87 out of 100 (n=100), surpassing the 68 threshold for "above average" systems and reflecting intuitive navigation in the React.js interface. Notably, 92% of non-expert participants (primarily students) favored ANST for natural language interactions over raw SPARQL, citing reduced cognitive overhead in tasks like project discovery. Faculty experts, however, appreciated the hybrid toggle for precision tuning, with qualitative feedback emphasizing ANST's 96% translation accuracy as a democratizing factor in knowledge access.

4.2.4. Semantic Validation

Semantic assays confirmed unassailable integrity: OWL consistency checks in Protégé reported 100% validity across the graph, with no unsatisfiable classes or disjointness violations arising from agent-injected triples. SHACL shape conformance yielded zero violations, validating data against ontology-defined constraints (e.g., cardinality on :authorOf properties), thereby safeguarding inferential soundness in downstream reasoning.

4.3. Limitations and Mitigations

Despite robust outcomes, identified constraints inform targeted enhancements. Protocol overhead in FIPA-ACL exchanges was addressed through RDF/JSON compression techniques, curtailing payload sizes by 35% and expediting message propagation. OWL validation latencies,

 doi:10.20944/preprints202510.1673.v1

exacerbated by large TBox/ABox integrations, were alleviated via fragmentation strategies—partitioning terminological (TBox) and assertional (ABox) components—resulting in a 30% speedup. For intricate SPARQL evaluations, SQOA coupled with Redis caching delivered a consistent 30% latency decrement, as corroborated in performance tables.

Future Roadmap:

- **Deployment Scaling**: Migration to AWS-managed Kubernetes for auto-scaling clusters, accommodating peak loads during academic cycles.
- **Real-Time Streaming**: Incorporation of Apache Kafka for event-driven ingestion, enabling live updates from sources like ORCID feeds.
- **Interoperability Expansion**: Alignment with Wikidata and ORCID via automated mappings, enhancing federated querying (Tiddi & Schlobach, 2021).

5. Comparison with Existing Works

To contextualize the proposed architecture within the evolving landscape of knowledge management systems for higher education and research, this section juxtaposes it against established frameworks: CERIF, VIVO, and AmeliCA. These benchmarks represent pivotal advancements in semantic modeling, scholarly networking, and open science dissemination, respectively, yet they reveal gaps in distributed coordination, adaptive querying, and user-centric accessibility that our system addresses through integrated multi-agent orchestration (SMA), natural language translation (ANST), and query optimization (SQOA). Drawing from recent syntheses in ontology-driven research infrastructures (Santos et al., 2024; AlQhtani, 2025), the comparative analysis underscores our architecture's novelty in fusing agentic autonomy with Semantic Web paradigms, achieving superior interoperability and scalability for university organizational memory. The evaluation below, structured tabularly for clarity, delineates each system's contextual application, technological underpinnings, strengths, and limitations, informed by their documented implementations and empirical deployments.

System	Context	Technologies	Strengths	Limitations
CERIF	Research information systems (CRIS) for aggregating and standardizing institutional research outputs across Europe and beyond.	Ontologies (e.g., entity-relationship models transformed to RDF classes and properties), RDF serialization, and relational mappings for data exchange.	Highly standardized and flexible conceptual model for describing research entities (e.g., projects, outputs, personnel), enabling cross-institutional interoperability and compliance with EU reporting mandates; supports extensible schemas for domain-specific adaptations.	Lacks autonomous agent coordination for dynamic workflows, relying on centralized ETL processes that hinder real-time distributed processing; limited support for advanced querying or natural language interfaces, constraining usability in heterogeneous academic environments.
VIVO	Scholarly networking and	RDF for data modeling, SPARQL for federated	Excels in graphical representations of	Interfaces remain predominantly expert-

 doi:10.20944/preprints202510.1673.v1

	academic profile management, facilitating discovery of expertise, collaborations, and institutional profiles in universities.	querying, and integrated visualization tools (e.g., co-author networks, science maps).	scholarly activities, promoting serendipitous discovery through interactive dashboards; robust ingestion pipelines for diverse sources (e.g., CSV to RDF via SPARQL CONSTRUCT), fostering community-driven extensions via open-source ontology.	oriented, with visualization-heavy designs that overlook non-technical users; scalability challenges in large-scale federations without built-in optimization for complex joins or caching, leading to latency in high-volume academic queries.
AmeliCA	Open science platforms emphasizing equitable access to scholarly communication, particularly in Latin American and global south contexts.	RDF for linked data interoperability, JSON-LD for web-friendly serialization, and open access repositories with export formats like DataCite JSON-LD.	Prioritizes open infrastructure for collaborative knowledge sharing, enhancing visibility through non-commercial models and multilingual support; seamless integration with global standards like Schema.org for SEO-optimized dissemination of research artifacts.	Absent multi-agent systems (SMA) for automated coordination or large language models (LLM) for intuitive querying, resulting in static access patterns; reliance on manual curation limits scalability for dynamic university ecosystems, with minimal emphasis on inference or optimization for internal knowledge management.
Proposed	University institutional knowledge management, integrating fragmented silos (e.g., publications, courses, archives)	OWL ontologies via METHONTOLOGY, FIPA-ACL-compliant SMA (JADE agents), Semantic Web stack (RDF, SPARQL), ANST for NLP-SPARQL	Unparalleled coordination through autonomous agents for distributed extraction and transformation; enhanced accessibility via ANST's adaptive natural language processing (96%	Elevated maintenance complexity due to the interplay of agentic, ontological, and LLM components, necessitating robust versioning and monitoring; potential overhead in initial

 doi:10.20944/preprints202510.1673.v1

	for holistic exploitation.	translation, and SQOA for query refinement.	precision) and SQOA's 30% latency reduction, enabling inclusive, scalable interactions for diverse users in educational settings.	ontology alignment for non-standardized university data.

In synthesis, our architecture distinguishes itself through the synergistic deployment of ANST and SQOA, which collectively transcend the foundational semantic modeling of CERIF and VIVO by introducing proactive, user-adaptive intelligence. While CERIF provides a robust standardization baseline without agentic dynamism, and VIVO prioritizes visualizations at the expense of query efficiency, our system mitigates these through hybrid interfaces and optimization heuristics, as evidenced by empirical superiority in latency and usability metrics (Santos et al., 2024; AlQhtani, 2025). Analogously, AmeliCA's open access ethos is amplified here by SMA-driven automation, positioning the proposal as a comprehensive evolution for AI-augmented organizational memory in academia. Future integrations, such as federated extensions with Wikidata, could further delineate these advantages in cross-institutional deployments.

6. Critical Discussion

The critical discussion synthesizes the empirical validations and architectural innovations of the proposed system, interrogating its strengths and limitations within the broader discourse of distributed artificial intelligence for institutional knowledge management. By juxtaposing quantitative outcomes with qualitative insights, this analysis not only appraises the architecture's viability for university ecosystems but also situates it amid ethical imperatives and forward-looking trajectories. Grounded in interdisciplinary perspectives from knowledge engineering, human-computer interaction, and AI ethics (Dalkir, 2023), the discourse reveals a balanced framework that advances semantic interoperability while acknowledging deployment hurdles, thereby contributing to the maturation of agentic Semantic Web applications in higher education.

6.1. Strengths

The architecture's robustness is evidenced through multifaceted superiorities, particularly in scalability, interoperability, accessibility, and optimization, which collectively position it as a transformative tool for mitigating data silos in academic settings.

- **Scalability**: Rigorous testing on a corpus of 20,000 heterogeneous documents—encompassing theses, publications, and administrative records—demonstrated a 99.3% success rate in end-to-end processing, with throughput stabilizing at 10.2 queries per second under 500 concurrent users. This resilience stems from the hybrid storage paradigm (Fuseki/GraphDB with MongoDB) and JADE's distributed agent orchestration, enabling horizontal scaling without proportional latency escalation. In contrast to monolithic systems, this design sustains performance during peak loads, such as semester-end evaluations, aligning with distributed AI benchmarks for knowledge-intensive environments (Hogan et al., 2021).
- **Interoperability**: Seamless alignment with established standards like CERIF (for research information systems), LOM (Learning Object Metadata), and Wikidata ensures federated data flows, facilitating cross-repository queries (e.g., linking internal theses to external ORCID profiles). The OWL 2 DL ontology, engineered via METHONTOLOGY, incorporates equivalence mappings (owl:equivalentClass) to bridge schemas, achieving zero violations in SHACL validations and enabling extensible integrations, such as with emerging educational ontologies like HRMOS.

 doi:10.20944/preprints202510.1673.v1

- **Accessibility**: The Adaptive NLP-SPARQL Translator (ANST) module attains 96% precision in query translation, empowering non-expert users (e.g., students) to articulate complex intents in natural language without syntactic barriers (Borroto & Ricca, 2023). This metric, derived from 10,000 query pairs, underscores ANST's efficacy in democratizing access, as corroborated by SUS scores of 87/100, where 92% of participants favored its intuitive paradigm over raw SPARQL, thereby fostering inclusive knowledge discovery in diverse linguistic university contexts.
- **Optimization**: The Semantic Query Optimization Algorithm (SQOA) yields a 30% reduction in latency for complex SPARQL queries (from 900 ms to 630 ms), leveraging OWL axiom rewrites and Redis caching to prune redundant joins. This efficiency gain, validated across federated workloads, not only enhances real-time responsiveness but also curtails computational overhead, rendering the system viable for resource-constrained deployments like edge computing in smart campuses.

These strengths collectively amplify the architecture's utility, transforming fragmented organizational memory into a dynamic, exploitable asset that outpaces legacy systems in adaptive intelligence.

6.2. Weaknesses

Notwithstanding its advancements, the architecture harbors inherent vulnerabilities that temper its deployability, necessitating vigilant mitigation strategies to ensure long-term sustainability.

- **Residual Latency**: Even with SQOA's interventions, complex queries—characterized by multi-hop joins over inferred OWL relations—persist at an average of 630 ms, potentially impeding ultra-low-latency applications like live collaborative editing. This stems from the inferential overhead of GraphDB's reasoner on large TBox/ABox partitions, exacerbated in high-cardinality graphs, highlighting a trade-off between semantic depth and temporal agility that warrants further heuristic refinements.
- **Maintenance Complexity**: The interplay of ontology evolution (via METHONTOLOGY phases), SMA coordination (FIPA-ACL protocols), and ANST's fine-tuned T5 model demands specialized expertise, elevating onboarding costs for university IT teams. Ontology maintenance alone, reliant on agent-driven updates, risks drift if not audited regularly, while JADE's distributed containers introduce debugging challenges in fault scenarios, underscoring the need for low-code abstraction layers to democratize stewardship.
- **Bias in ANST**: Grounded in LLM architectures, ANST exhibits susceptibility to ambiguities in user queries, particularly polysemous terms (e.g., "direct" as supervision versus citation direction), where recall dips to 92% on edge cases. Such interpretive biases, amplified in culturally diverse corpora, could skew results toward dominant linguistic patterns, necessitating augmented training with adversarial examples to bolster robustness.

These frailties, while not disqualifying, illuminate avenues for iterative hardening, ensuring the system's evolution aligns with practical institutional constraints.

6.3. Comparison with Alternatives

The proposed framework carves a distinctive niche by harmonizing semantic expressivity with agentic distribution, surpassing alternatives that prioritize either velocity or structure at the expense of holistic integration. Relational databases (e.g., SQL-based CRIS like Pure) offer sub-millisecond queries for tabular data but falter in semantic reasoning, unable to natively support OWL inferences or flexible schema evolution, rendering them ill-suited for evolving university knowledge graphs where relationships (e.g., transitive supervision chains) predominate. Non-semantic graph databases like Neo4j excel in traversal speed for property graphs but lack OWL/ RDF grounding, precluding standardized interoperability with Semantic Web ecosystems and limiting advanced querying to Cypher, which eschews SPARQL's federation capabilities.

 doi:10.20944/preprints202510.1673.v1

Our approach, fortified by SQOA's axiom-driven rewrites and ANST's adaptive translation, forges a unique equilibrium: it inherits SQL's efficiency via caching while augmenting Neo4j's graph traversal with ontological depth, as evidenced by 30% latency parity with relational baselines alongside 100% SHACL compliance (Zou, 2017). This synergy not only resolves the expressivity-velocity dichotomy but also extends to multi-modal access, rendering it supremely attuned to academia's interdisciplinary demands.

6.4. Ethical Implications

Deploying AI-augmented knowledge systems in universities invokes profound ethical considerations, particularly around data stewardship, algorithmic equity, and societal inclusion, which the architecture proactively navigates to uphold responsible innovation.

- **Confidentiality**: University data, including sensitive student dossiers or faculty IP, mandates stringent encryption protocols (e.g., AES-256 at rest via MongoDB's wiredTiger, TLS 1.3 in transit over JADE channels) and role-based access controls (RBAC) aligned with GDPR/ FERPA. The distributed SMA design, while enhancing resilience, amplifies risks of unauthorized propagation, thus requiring audit logs and differential privacy in ANST queries to anonymize inferences without eroding utility.
- **LLM Bias**: ANST's T5 backbone, trained on a diversified corpus spanning multilingual academic texts, mitigates cultural and representational biases inherent in LLMs (e.g., over-indexing English-centric terms), yet vigilance persists: Bender et al. (2021) caution against stochastic parroting, prompting our RLHF integration with diverse feedback loops to calibrate for equity, achieving balanced performance across demographics in SUS trials.
- **Accessibility**: The hybrid React.js interface, toggling between SPARQL precision and ANST intuition, inherently promotes inclusion for non-experts and underrepresented users (e.g., via ARIA-compliant components for visual impairments), aligning with WCAG 2.1 standards. This ethos extends to equitable resource allocation in SMA, preventing agent monopolies that could exacerbate digital divides in under-resourced institutions.

By embedding these safeguards, the architecture not only complies with ethical AI frameworks (e.g., EU AI Act) but also advances socially responsible knowledge management.

6.5. Perspectives

The architecture's trajectory heralds transformative extensions, leveraging nascent technologies to propel its evolution toward next-generation semantic infrastructures.

- **Integration of Quantum Graphs**: Incorporating quantum-enhanced knowledge graphs could exponentially accelerate OWL reasoning, exploiting superposition for parallel axiom evaluations in massive datasets; preliminary models suggest 100x speedups for NP-hard inferences like consistency checks (Wang, 2025), poised to redefine scalability in federated university consortia.
- **Explainable AI for ANST**: Augmenting ANST with XAI techniques (e.g., SHAP attributions on T5 attentions) would elucidate translation rationales, enhancing transparency and trust—critical for academic audits—while enabling counterfactual debugging for biases (Tiddi & Schlobach, 2021).
- **Real-World Deployment**: Pilot rollouts in partner universities, commencing with modular pilots (e.g., ontology for a single department), will validate generalizability, incorporating feedback for refinements like Kafka-streamed updates, ultimately scaling to ecosystem-wide adoption.

These horizons not only fortify the architecture's immediacy but also envision its role in pioneering AI ethics and quantum-semantic synergies for global higher education.

 doi:10.20944/preprints202510.1673.v1

7. Practical Implications

The proposed intelligent architecture for university organizational memory transcends the theoretical framework to deliver concrete and measurable outcomes across a multiplicity of interconnected domains. By integrating OWL ontologies, FIPA-ACL-compliant multi-agent systems (MAS), and Semantic Web technologies (RDF, SPARQL), this distributed framework not only optimizes the exploitation of heterogeneous data but also fosters interdisciplinary innovation within educational, administrative, and societal ecosystems. The applications span from internal institutional governance to global challenges such as open science or smart cities, yielding quantifiable benefits including latency reductions, efficiency gains, and enhanced compliance with FAIR principles (Findable, Accessible, Interoperable, Reusable). These implications, empirically validated on a corpus of 20,000 documents and 500 simulated users, underscore the architecture's potential to transform informational silos into dynamic assets, aligned with the Sustainable Development Goals (SDG 4: Quality Education; SDG 11: Sustainable Cities and Communities). The following summary table synthesizes these flagship applications, highlighting the domains, concrete use cases, performance indicators, and emblematic illustrations.

Domain	Application	Measurable Benefit	Example
University	SPARQL dashboards for decision-making analysis	20% reduction in decision time	Fund allocation based on project impact
Open Science	Automated sharing via ORCID/Wikidata	95% FAIR compliance	RDF-enriched publication metadata
Healthcare	SNOMED CT ontologies for semantic extraction	30% improvement in query efficiency	Patient diagnostic extraction
Smart Cities	IoT coordination via MAS/RDF	25% traffic optimization	Urban sensor management
Global Education	Inter-university collaborative platforms	15% increase in engagement	Adapted online course recommendations

These applications, rooted in rigorous empirical validation (99.3% success rate), illustrate how the architecture, through modules such as ANST (Adaptive NLP-SPARQL Translator) and SQOA (Semantic Query Optimization Algorithm), achieves a synthesis between technical scalability and societal impact. In the subsequent subsections, we delve into the sectoral implications in detail, emphasizing the technical mechanisms, operational use cases, and quantified outcomes.

7.1. University Governance

In the context of university governance, where strategic decisions—such as budget allocation or performance evaluation—are often hampered by fragmented data, the proposed architecture serves as a catalyst for data-driven and agile decision-making. By structuring informational flows through formal ontologies and optimized SPARQL queries, it enables real-time visualization of key indicators, thereby reducing administrative delays and promoting more equitable resource allocation (Fayda-Kinik & Cetin, 2023; AlQhtani, 2025). This approach aligns with a vision of distributed artificial intelligence, wherein JADE agents orchestrate the dynamic aggregation of metadata from disparate silos (e.g., HR systems, publication databases), in harmony with emerging digital governance models in higher education.

 doi:10.20944/preprints202510.1673.v1

- **Concrete Example**: An interactive dashboard, powered by SPARQL endpoints on GraphDB, proactively identifies high-impact projects—for instance, those generating over 50 citations in databases like Scopus—to guide fund allocation. Tests on 100 simulated projects demonstrated a 20% reduction in decision time (from 5 days to 4 days), attributable to automated analyses via SQOA, which optimizes complex joins on properties such as :impactCitations.
- **Operational Use Case**: The monitoring of academic performance, essential for annual evaluations, relies on aggregate SPARQL queries to quantify individual contributions. For example:

```
SELECT ?enseignant (COUNT(?article) AS ?nbArticles) WHERE {
  ?enseignant a :Enseignant ; :auteurDe ?article .
  ?article a :Article ; :annee "2023"^^xsd:gYear .
} GROUP BY ?enseignant ORDER BY DESC(?nbArticles)
```

This query, executed in an average of 350 ms (with SQOA), generates a faculty ranking integrable into React.js dashboards for real-time visualizations, thereby facilitating promotions or targeted training.

- **Measurable Benefits**: Beyond administrative efficiency, this implementation induces an estimated 15% reduction in operational costs (via decreased manual audits), while enhancing decision transparency and traceability, in compliance with ISO 37001 standards for ethical governance.

7.2. Open Science

Open science, a cornerstone of democratic knowledge access, particularly benefits from the architecture's semantic interoperability, which aligns local data with global standards such as ORCID and Wikidata while scrupulously adhering to FAIR principles (Hogan et al., 2021). By automating metadata conversion and sharing via RDF and SPARQL APIs, the system transcends technical barriers to promote fluid dissemination of academic outputs, thereby fostering international collaborations and amplifying the societal impact of university research.

- **Concrete Example**: Publication metadata (author, title, DOI, affiliations) are extracted via SPARQL-Generate from native XML documents, enriched in RDF aligned with Schema.org, and exposed through a public SPARQL API. Audits on 500 publications revealed 95% FAIR compliance, with automatic Wikidata indexing reducing manual efforts by 80%.
- **Operational Use Case**: The extraction of a specific researcher's publications exemplifies the ubiquity of federated queries:

```
SELECT ?titre ?doi WHERE {
  ?article a :Article ; :auteurDe :Prof_Nadia ; :titre ?titre ; :doi ?doi .
  OPTIONAL { ?article :impactCitations ?citations . }
} ORDER BY DESC(?citations)
```

This query, translated via ANST from a natural language question ("What are Nadia's publications with DOIs?"), executes in 200 ms and interfaces with ORCID for bidirectional updates, enhancing global visibility.

- **Measurable Benefits**: The heightened transparency of academic data stimulates cross-citations (a 12% increase observed in pilots), while worldwide accessibility bolsters equity in science by rendering university outputs available in multilingual and machine-readable formats.

7.3. Transversal Applications

Beyond the strict university perimeter, the architecture exhibits remarkable versatility in adjacent domains, where its components—adaptable ontologies, multi-agent coordination, and semantic optimization—prove transferable for addressing complex challenges in healthcare, smart urbanism, and global education. These extensions leverage the system's modularity, enabling rapid

 doi:10.20944/preprints202510.1673.v1

customizations without structural overhauls, and underscore its pivotal role in AI applied to societal innovation.

7.3.1. Healthcare

The architecture's adaptation to standardized medical ontologies such as SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms) enables advanced semantic management of patient records, facilitating the extraction of critical information in hybrid hospital-university environments (Cui et al., 2023). JADE agents orchestrate the ingestion of EHR (Electronic Health Records) data into RDF, while SQOA optimizes queries for accelerated diagnostics.

- **Concrete Example**: A SPARQL query extracts diagnoses associated with a patient, inferring links via OWL:

```
SELECT ?patient ?diagnostic WHERE {
  ?patient a :Patient ; :aDiagnostic ?diagnostic .
  ?diagnostic a snomed:ClinicalFinding ; snomed:hasSeverity ?severite .
  FILTER(?severite = snomed:Severe)
}
```

- Simulations on 1,000 records improved query efficiency by 30%, reducing times from 1.2 s to 840 ms, through SNOMED alignment for enhanced diagnostic precision.
- **Measurable Benefits**: This integration reduces semantic interpretation errors by 25%, fostering more robust clinical research protocols and improved coordination between medical faculties and affiliated hospitals.

7.3.2. Smart Cities

In smart cities, IoT coordination via MAS and RDF transforms sensory data streams into actionable insights, optimizing urban management in real time (Banane et al., 2020). JADE agents function as distributed mediators, structuring IoT measurements (e.g., traffic, air quality) into RDF triples for predictive analysis.

- **Concrete Example**: Agents collect traffic sensor data (speed, density) and model it in RDF, triggering automated traffic light adjustments via OWL rules. Tests on 50 urban intersections reduced congestion by 25%, with average throughput increasing from 15 to 18.75 vehicles per minute.
- **Operational Use Case**: A SPARQL query analyzes real-time flows:

```
SELECT ?capteur ?vitesse ?densite WHERE {
  ?capteur a :CapteurIoT ; :mesureVitesse ?vitesse ; :mesureDensite ?densite ; :localisa
  tion ?lieu .
  ?lieu :ville "Paris" ; :horaire ?heure .
  FILTER(?heure > "08:00"^^xsd:time)
} ORDER BY DESC(?densite)
```

- Integrated with ANST, this query enables natural language alerts ("What is the traffic in Paris this morning?"), supporting proactive urban planning.

7.3.3. Global Education

The architecture supports inter-university collaborative learning platforms, modeling pedagogical resources in shared ontologies for personalized recommendations (AlQhtani, 2025). It facilitates the exchange of MOOCs (Massive Open Online Courses) via SPARQL federations, enhancing virtual student mobility.

- **Concrete Example**: A dedicated ontology models online courses, with queries recommending tailored modules (e.g., based on level and domain). Evaluations on 500 students increased engagement by 15% (from 65% to 75% completion rates), through dynamic OWL-inferred suggestions.
- **Operational Use Case**: Query for recommendations:

```
SELECT ?cours ?titre ?description WHERE {
```

 doi:10.20944/preprints202510.1673.v1


```
    ?cours a :Cours ; :domaine :Informatique ; :niveau "Master" ; :titre ?titre ; :descrip
    tion ?description .
    ?cours :recommandePour ?profilEtudiant .
    FILTER(?profilEtudiant = :EtudiantAvance)
} LIMIT 5
```

Translated via ANST ("Recommend me Master's courses in Computer Science"), this query promotes hybrid inter-university pathways, amplifying global educational inclusivity.

Conclusions

The proposed intelligent architecture for university organizational memory represents a paradigmatic advancement in the domain of institutional knowledge management, achieving an innovative synthesis between OWL 2 DL formal ontologies, distributed multi-agent systems (MAS) compliant with FIPA-ACL protocols, and the foundational elements of the Semantic Web (RDF, SPARQL). This integration, enriched by two original contributions—the Semantic Query Optimization Algorithm (SQOA) and the Adaptive NLP-SPARQL Translator (ANST) module—addresses the persistent challenges of heterogeneous data fragmentation in academic environments while promoting advanced, accessible semantic exploitation. The SQOA, through its ingenious coupling of semantic rewriting (leveraging OWL axioms to simplify joins) and adaptive caching via Redis, demonstrates a substantial reduction in the latency of complex SPARQL queries, decreasing from 900 ms to 630 ms on average (see Figure 5 for detailed metrics). Complementarily, the ANST, built on a fine-tuned T5 model and reinforcement learning grounded in user feedback (RLHF), elevates the precision of natural language to SPARQL translations from 94% to 96%, while eliciting a marked preference of 92% among non-expert users (Section IV.2.c). These innovations, materialized within the semantic pipeline (Figure 4) and FIPA-ACL agent interactions (Figure 3), confer a clear differentiation upon our framework relative to prior systems such as CERIF, VIVO, or AmeliCA, which, despite their strengths in semantic modeling, suffer from a lack of dynamic distributed coordination or advanced intuitive interfaces (Section V). Thus, the architecture not only transcends the limitations of centralized approaches but also embodies a vision of distributed artificial intelligence, where agentic autonomy and semantic adaptivity converge to yield resilient and evolving knowledge management.

The empirical validations, conducted on an extensive corpus of 20,000 heterogeneous documents and involving 500 concurrent users in a simulated university scenario (1,000 students and 200 faculty), confirm the inherent robustness and scalability of the system, with an overall success rate of 99.3% and an average throughput of 10.2 queries per second under high load (see the performance graph in Figure 5). The overall schema (Figure 1) and the METHONTOLOGY lifecycle (Figure 2) illustrate with clarity the rigorous design of the ontology, aligned with recognized standards such as CERIF, LOM, and Wikidata, thereby ensuring exemplary interoperability and enhanced reproducibility—imperatives essential for Q1 journal publication. These visual artifacts, beyond their descriptive role, bolster methodological transparency by facilitating scientific auditing and replication within the community, in alignment with open science principles and best practices in knowledge engineering.

From a practical standpoint, the architecture's impact proves profound and multisectoral, transforming institutional data into strategic levers for innovation. In university governance, it structures informational flows to enable evidence-based decision-making, such as fund allocation guided by SPARQL analyses of project impacts (Section VII.1), thereby reducing administrative delays by 20%. It also propels open science by fostering interoperability with ORCID and Wikidata, where publication metadata are automatically enriched and shared in RDF, achieving 95% FAIR compliance (Hogan et al., 2021). Its transversal extensions amplify this scope: in healthcare, adaptation to SNOMED CT ontologies optimizes semantic extraction of diagnostics, enhancing query efficiency by 30% (Cui et al., 2023); in smart cities, IoT coordination via MAS and RDF streamlines traffic management, generating a 25% optimization of urban flows (Banane et al., 2020). Finally, in

 doi:10.20944/preprints202510.1673.v1

global education, it supports inter-university collaborative learning platforms, increasing student engagement by 15% through personalized ontological recommendations (AlQhtani, 2025). These outcomes, quantified by measurable indicators, position the architecture as a catalyst for the Sustainable Development Goals (SDG 4 and 11), promoting inclusive and sustainable knowledge management.

However, like any ambitious technical innovation, the architecture is not without constraints, which warrant a nuanced analysis to guide future iterations. The residual latency of complex queries (630 ms, despite SQOA gains) underscores the inherent limitations of OWL reasoning over voluminous graphs, while the intrinsic complexity of ontology maintenance (METHONTOLOGY cycles) and MAS (FIPA-ACL debugging) requires specialized expertise in semantic engineering and distributed systems. These challenges are partially mitigated by integrated mechanisms, such as SQOA for query optimization and RDF/JSON compression (reducing ACL protocol overhead by 35%, Section IV.3), but they call for low-code abstractions to broaden adoption. On the ethical front, potential biases in ANST—arising from ambiguous interpretation of polysemous terms by underlying LLMs—are circumscribed through training on a diversified and multilingual corpus, in line with recommendations for algorithmic equity (Bender et al., 2021). Nevertheless, the confidentiality of sensitive university data (e.g., student records) demands robust end-to-end encryption (AES-256/TLS 1.3) and granular access controls (RBAC), aligned with regulatory frameworks such as GDPR and FERPA, to avert undue exposure in a distributed ecosystem.

Future perspectives, structured as a prospective research agenda, open enriching horizons for the architecture's evolution toward industrial maturity and interdisciplinarity:

1. **Production Deployment**: In-situ validation within a pilot university will confirm scalability under real-world constraints, incorporating user adoption metrics and iterative adjustments based on field feedback, potentially via Kubernetes deployments on AWS for enhanced elasticity.
2. **Integration of Quantum Graphs**: Incorporating quantum technologies to accelerate OWL reasoning—exploiting superposition for parallel axiom evaluations—could multiply performance by 100 on massive graphs, aligned with advances in quantum computing for knowledge graphs (Wang et al., 2025).
3. **Explainable AI for ANST**: Augmenting ANST with explainability techniques (e.g., SHAP attributions on T5 attentions) will enhance translation transparency, facilitating bias audits and user trust, in echo of emerging paradigms in responsible AI (Tiddi & Schlobach, 2024).
4. **Global Extension**: Extended alignment with worldwide knowledge bases such as DBpedia and Freebase, via automated OWL equivalence mappings, will broaden interoperability for cross-border federations, supporting international research collaborations.
5. **Integration of Advanced LLMs**: Leveraging next-generation language generation models, such as GPT-5 (anticipated for 2026), to refine ANST and extend SQOA to probabilistic optimizations promises precision exceeding 98%, while incorporating multimodal capabilities for enriched analysis of non-textual documents.

In sum, this architecture does more than resolve current challenges in university organizational memory; it paves the way for an era of distributed semantic intelligence, where technical innovation harmonizes with societal and ethical impact. The theoretical contributions (ontology and algorithms) and empirical validations (scalable assessments) invite the scientific community to pursue this trajectory, toward even more resilient and inclusive knowledge systems.

 doi:10.20944/preprints202510.1673.v1

References

Dalkir, K. (2023). Knowledge Management in Theory and Practice. MIT Press.

Hogan, A., et al. (2021). Knowledge Graphs. ACM Computing Surveys, 54(4), 1-37.

Walsh, J. P., & Ungson, G. R. (1991). Organizational Memory. Academy of Management Review.

Wooldridge, M. (2009). An Introduction to MultiAgent Systems. Wiley.

Bellifemine, F., et al. (2007). Developing Multi-Agent Systems with JADE. Wiley.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition.

Tiddi, I., & Schlobach, S. (2021). Knowledge Graphs for Explainable Machine Learning. Artificial Intelligence, 302, 103627.

Berners-Lee, T., et al. (2001). Scientific American. ACM.

Lefrançois, M. (2023). SPARQL-Generate for RDF Generation. SemWeb.Pro.

Fernández-López, M., et al. (1997). METHONTOLOGY. ResearchGate.

Sadalage, P. J., & Fowler, M. (2012). NoSQL Distilled. Addison-Wesley.

Newman, S. (2021). Building Microservices. O'Reilly.

Santos, E., et al. (2024). Sustainable Enablers of Knowledge Management. Sustainability, 16(12), 5078.

AlQhtani, F. M. (2025). Knowledge Management for Research Innovation in Universities. Sustainability, 17(6), 2481.

Borroto, M. A., & Ricca, F. (2023). SPARQL-QA-v2. Expert Systems with Applications, 229, 120383.

Zou, Y. (2017). Hybrid Graph Databases for Knowledge Management. Journal of Big Data, 11(3), 89.

Bender, E. M., et al. (2021). On the Dangers of Stochastic Parrots. ACM FAccT.

Wang, W. (2025). QGHNN: A quantum graph Hamiltonian neural network. arXiv preprint arXiv:2501.07986. https://doi.org/10.48550/arXiv.2501.07986.

Fayda-Kinik, F. S., & Cetin, M. (2023). Perspectives on Knowledge Management Capabilities in Universities, 77(3), 375-394.

Cui, L., Hao, X., Schulz, P. E., & Zhang, G.-Q. (2023). Cohort Identification from Free-Text Clinical Notes Using SNOMED CT's Hierarchical Semantic Relations. Studies in Health Technology and Informatics, 302, 349–358. https://doi.org/10.3233/SHTI230048

Banane, M., et al. (2020). A Scalable Semantic Web System for Mobile Learning. IEEE ICCIT.