Received 28 December 2027, accepted 27 February 2026, date of publication 31 March 2026
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.25.1.4486

An Approach to Ontology-Driven Processing of Scientific Texts and User Natural Language Queries

OLEKSANDR PALAGIN, MYKOLA PETRENKO, DMYTRO VYKHOVANETS, MYKOLA BOYKO
Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine,
Academician Glushkov Avenue, 40, Kyiv, 03187, Ukraine

Corresponding author: Oleksandr Palagin (e-mail: palagin_a@ukr.net).

The research was carried out within the framework of the Institute of Cybernetics research themes: "Develop theoretical foundations and a functional computer model for processing complex information structures" (State Registration No. 0119U002226); and "Develop tools to support virtualization technologies and their use in computer engineering and other applications" (State Registration No. 0124U001826).

ABSTRACT This research presents a method and system architecture for ontology-driven processing of scientific natural-language texts that integrates a linguistic processor, a domain OWL ontology, a reasoning engine, and a knowledge graph. The system constructs a consistent knowledge graph suitable for the automatic interpretation of structurally complex user queries with subsequent transformation into SPARQL queries. The proposed pipeline includes formal mappings of linguistic analysis results to ontology classes, properties, and assertions; consistency checking; fact materialization; and an explanation mechanism that derives minimal sets of axioms and facts to justify the reasoning engine's conclusions or to identify the causes of inconsistency. The output is a set of consistency-compliant subject–predicate–object triples. An ontology-consistency evaluation metric is introduced as the proportion of triples in the knowledge graph that do not violate ontological constraints, and the impact of logical processing on the harmonic mean of precision and recall for triple extraction is evaluated on a representative corpus of scientific texts in the field of knowledge engineering.

KEYWORDS ontology engineering; semantic web technologies; scientific texts; knowledge graph; OWL ontology; RDF triples; logical inference.

I. INTRODUCTION

In recent years, neural-network methods for natural language processing have advanced significantly. Despite high performance on benchmark test suites, both industry and academia face persistent challenges related to explainability, controllability, knowledge consistency, and knowledge reuse. Within the ontology-driven paradigm, a complementary approach is advocated, based on the use of explicit domain models, strict logical-consistency constraints, and semantically grounded reasoning, in order to increase trustworthiness, controllability, and reproducibility of knowledge artifacts. Broader issues concerning the role of ontology engineering in building a "knowledge industry" and supporting the full research life cycle are discussed in detail in [1–4].

This paper presents a system architecture that implements the following processing chain: "linguistic processing of scientific natural language texts → OWL ontology → logical inference engine (reasoner) → knowledge graph (KG) → natural language queries → SPARQL queries." The architecture systematizes the extracted basic linguistic knowledge in the form of elementary senses and the corresponding Subject, Predicate, Object triples (⟨S, P, O⟩ triples), extending the approach proposed in [5]. The solution is targeted at corpora of scientific publications in knowledge engineering; however, the system can also be implemented for other domains due to the consistency of interfaces and the availability of appropriate ontologies. The resulting knowledge graph is treated as a target model for the automatic interpretation of complex structured natural-language user queries.

II. RELATED WORKS

Research most closely related to our work can be broadly grouped into four directions: (1) ontology learning methods for text processing [5–17]; (2) ontology-driven information and relation extraction (information extraction / relation extraction, IE/RE) [18–19]; (3) knowledge-graph models and architectures, including logical reasoning [20–29]; and (4) natural-language querying over knowledge graphs (NLP→SPARQL / QA over Linked Data) [6–14].

Experience with question answering over linked data indicates that the best performance is typically achieved by combining rule- and template-based processing, entity linking, and semantically grounded query construction [6–14]. In addition, controlled neural components are often employed, accompanied by mandatory validation of the generated SPARQL queries [10–12]. Representative systems and datasets in this line of work include SPARKLIS [8], QAnswer KG [9], Neural SPARQL Machines [10], neural NL→SPARQL models [11], SPARQL-QA-v2 [12], as well as the LC-QuAD and LC-QuAD 2.0 benchmarks [13, 14].

In the area of ontology learning from text, the seminal work by A. Maedche and S. Staab [15] formulated a semi-automatic approach to constructing Semantic Web ontologies based on statistical and linguistic methods. Subsequent advances were systematized in the survey by W. Y. Wong, W. Liu, and M. Bennamoun [16], which proposed a taxonomy of methods—from term and concept extraction to semantic relation extraction—and analyzed typical pipelines for ontology construction from texts across different domains. More recent studies, in particular J. Wątróbski [17], extend this line of research by incorporating knowledge-oriented approaches and addressing the quality of the resulting ontologies.

Within ontology-driven information extraction, the results reported by P. Cimiano, U. Reyle, and J. Šarić [18] showed that discourse analysis combined with an ontology improves the identification of relations and roles in text and enables detection of semantic anomalies through logical constraints. Later works, such as the study by V. Pertsas and P. Constantopoulos [19], focus on scientific publications and describe an ontology-driven extraction pipeline for research articles, where the ontology is used both as a domain model and as a framework for quality control and consistency of the extracted structured metadata.

Regarding knowledge graphs, foundational surveys [20–23] provide formal models of knowledge graphs, discuss the role of schema (TBox), identity, and context, and review methods for construction, enrichment, quality assessment, and publication of knowledge graphs in industrial and academic scenarios. The use of OWL ontologies, SWRL rules, and reasoning tools within such an infrastructure builds on Semantic Web standards and tool support [24, 27–29]. This is consistent with the linguistic processing chain proposed in this paper: "linguistic processing of scientific natural-language texts → OWL ontology → reasoning engine → knowledge graph → natural-language queries → SPARQL queries."

In the direction of QA systems over Linked Data and NLP→SPARQL, considerable attention has been devoted to surveys of querying systems for RDF/Linked Data and documents. The study by E. Dimitrakis, K. Sgontzos, and Y. Tzitzikas [7] analyzes the architecture of QA systems over linked data, classifies approaches by types of knowledge sources, question types, and domains, and systematizes widely used datasets and evaluation metrics. These works indicate that successful NLP→SPARQL solutions require consistent ontologies and stable knowledge-graph schemas as the target model for query generation [6–14].

For practical implementation of NLP→SPARQL, it is critically important to maintain a stable schema (classes/properties) and a whitelist of permitted SPARQL patterns. This reduces ambiguity in interpreting natural-language formulations and ensures safe query generation, in particular by preventing unauthorized injection of fragments into a SPARQL query through restricting substitutions to slot values only [21, 22].

The knowledge-oriented system architecture proposed in this paper generalizes and integrates the above directions, treating the ontology as a controlling component of the knowledge life cycle. Previously, the authors investigated ontology-driven processing of domain and transdisciplinary knowledge in the context of computer-based knowledge systems and ontological infrastructure for scientific research [1, 5]. The architecture presented in this study extends these approaches by formalizing mappings from the text level and introducing the OCS and ΔF1_reasoning metrics.

III. REPRESENTATION MODEL AND PROBLEM STATEMENT

The basic unit of knowledge representation in the proposed system is a triple of the form ⟨S, P, O⟩, where S (subject) denotes the subject, P (predicate) denotes the predicate (relation/property), and O (object) denotes the object. In formal terms:

$$S \in E,$$
$$P \in R,$$
$$O \in E \cup L,$$

where E is a set of individuals or classes (named entities), R is a set of binary relations (object properties and data properties), L is a set of literals (numbers, strings, dates, etc.).

The ontological part of the knowledge base comprises two mutually complementary components: TBox (terminological), which defines the intensional specification (intension) of the domain by describing classes, their hierarchies, domain/range restrictions for properties, cardinality axioms, and other logical constraints; and ABox (assertional/factual), which captures extensional information as assertions about individuals represented by ⟨S, P, O⟩ triples. Thus, the system must: (1) transform the textual corpus D into a set of candidate triples C; (2) select a subset C′ that is consistent with the TBox; and (3) materialize C′ in the knowledge graph while simultaneously supporting explanation and justification mechanisms for the derived assertions.

A. PROBLEM STATEMENT

Let R be the set of candidate relations extracted by the linguistic processor; E be the set of named entities and/or ontology classes; $C \subseteq E \times R \times (E \cup L)$ – C be the set of candidate S/P/O triples; T be the set of TBox axioms; and S be the set of SHACL shapes (Shapes Constraint Language). The task is to construct a subset C′ ⊆ C that satisfies the following requirements:

1. Consistency with respect to the TBox and SHACL, i.e., all triples in C′ do not violate domain/range constraints, cardinality restrictions, and additional constraints specified in the TBox and SHACL shapes;
2. Maximization of extraction quality measured by the F1-score relative to a gold-standard annotation;
3. Maximization of the Ontology-Consistency Score defined as the proportion of triples that pass ontological checks;
4. Provision of explanations for decisions using justification mechanisms.

B. ELEMENTARY SENSES OF A KNOWLEDGE DOMAIN

In [5], the notion of an elementary sense was introduced as a minimal semantic unit for representing domain knowledge, close in form to RDF triples. An elementary sense corresponds to a simple statement that can be represented as an ⟨S, P, O⟩ triple and serves as a basic (atomic) element for structured knowledge representation. In the context of scientific publications, complex sentences are decomposed into one or more elementary senses, which yields a compact and uniform representation of article content for subsequent ontology-driven processing.

IV. SYSTEM ARCHITECTURE

The architecture of the proposed ontology-driven system comprises a sequence of blocks and is presented in Fig. 1 using the C4 notation. The system implements the following processing chain: "linguistic processing of scientific natural language texts (PDF) → OWL ontology → reasoning engine → knowledge graph → natural-language queries → SPARQL queries." Each stage is associated with clearly defined interface formats specifying the structure of input and output data (JSON/RDF schemas, triple formats), admissible field values, event log formats, and artifact identification rules. This enables independent operation of components, supports experimental reproducibility, and increases processing transparency.

The linguistic processor includes modules for text normalization (encoding unification, removal of layout artifacts, segmentation into paragraphs and sentences), text chunking and morphosyntactic analysis (tokenization, part-of-speech tagging, lemmatization), syntactic parsing (construction of dependency trees, identification of predicative constructions), semantic annotation (named entity recognition, assignment of semantic roles), extraction of candidate ⟨S, P, O⟩ triples, and coreference resolution. The output is a set of candidate triples with associated weights, references to sentences and text spans, and preliminary hypotheses for mapping to ontological entities.

The linguistic processor is implemented as a pipeline of loosely coupled modules with well-defined input/output data formats, which allows individual components (e.g., the morphological analyzer or named entity recognition modules) to be updated or replaced independently without modifying the overall architecture of the sequence. For each stage, auxiliary metadata are stored (document and sentence identifiers, model versions, run parameters), which are later used to populate provenance metadata (PROV model) for the generated triples [26]. The current implementation supports processing of Ukrainian-language scientific texts, with the possibility of extension to other languages by integrating additional lexical resources and annotation models. A dedicated submodule performs internal validation of the consistency of linguistic data (in particular, alignment between morphological tags, syntactic dependencies, and semantic roles), which reduces the number of false candidate ⟨S, P, O⟩ triples passed to the ontology-mapping stage.

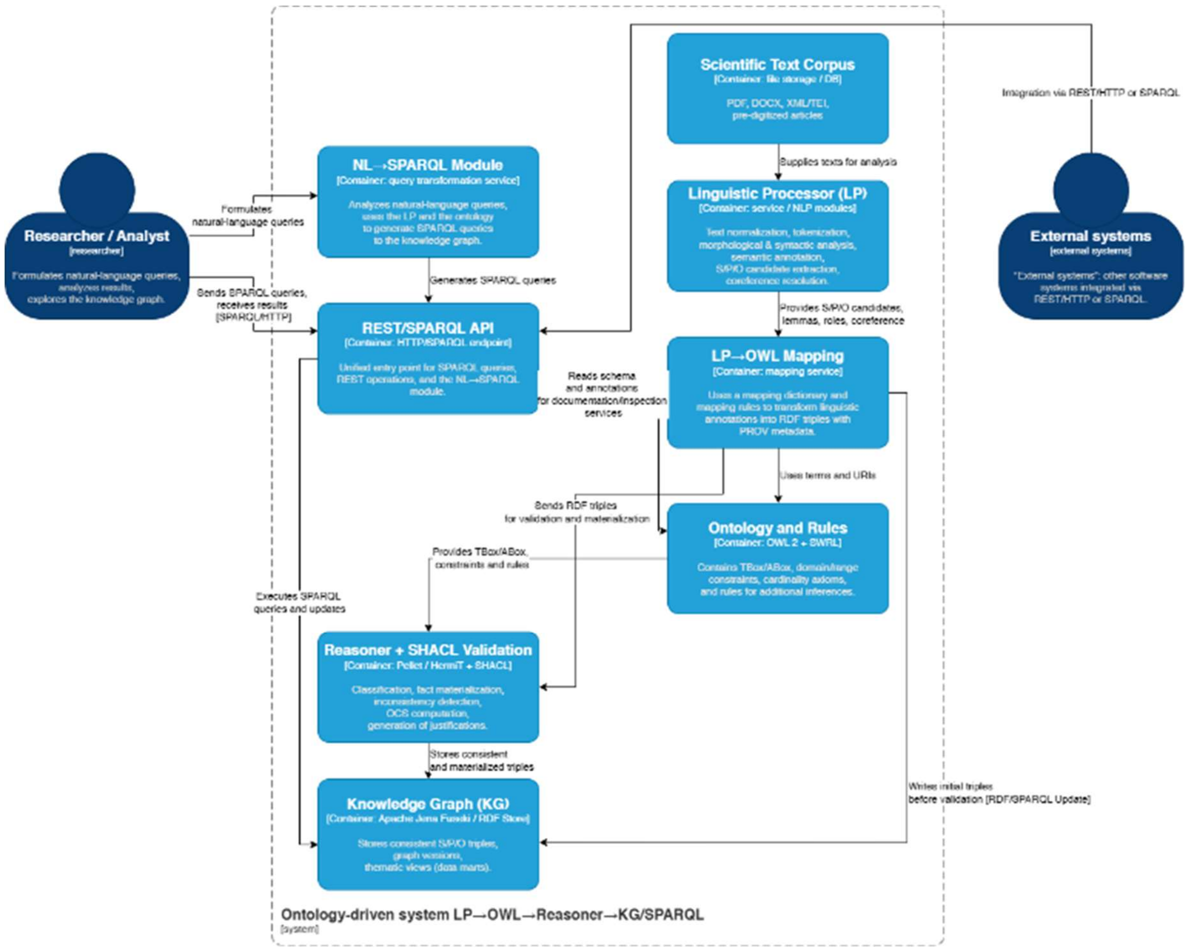
Researcher / Analyst
[researcher]
Formulates natural-language queries, analyzes results, explores the knowledge graph.
Formulates natural-language queries
Sends SPARQL queries, receives results [SPARQL/HTTP]
NL→SPARQL Module
[Container: query transformation service]
Analyzes natural-language queries, uses the LP and the ontology to generate SPARQL queries to the knowledge graph.
Generates SPARQL queries
REST/SPARQL API
[Container: HTTP/SPARQL endpoint]
Unified entry point for SPARQL queries, REST operations, and the NL→SPARQL module.
Reads schema and annotations for documentation/inspection services
Scientific Text Corpus
[Container: file storage / DB]
PDF, DOCX, XML/TEI, pre-digitized articles
Supplies texts for analysis
Linguistic Processor (LP)
[Container: service / NLP modules]
Text normalization, tokenization, morphological & syntactic analysis, semantic annotation, S/P/O candidate extraction, coreference resolution.
Provides S/P/O candidates, lemmas, roles, coreference
LP→OWL Mapping
[Container: mapping service]
Uses a mapping dictionary and mapping rules to transform linguistic annotations into RDF triples with PROV metadata.
Uses terms and URIs
Sends RDF triples for validation and materialization
Provides TBox/ABox, constraints and rules
Ontology and Rules
[Container: OWL 2 + SWRL]
Contains TBox/ABox, domain/range constraints, cardinality axioms, and rules for additional inferences.
Executes SPARQL queries and updates
Reasoner + SHACL Validation
[Container: Pellet / HermiT + SHACL]
Classification, fact materialization, inconsistency detection, OCS computation, generation of justifications.
Stores consistent and materialized triples
Knowledge Graph (KG)
[Container: Apache Jena Fuseki / RDF Store]
Stores consistent S/P/O triples, graph versions, thematic views (data marts).
Writes initial triples before validation [RDF/SPARQL Update]
Integration via REST/HTTP or SPARQL
External systems
[external systems]
"External systems": other software systems integrated via REST/HTTP or SPARQL.
Ontology-driven system LP→OWL→Reasoner→KG/SPARQL
[system]

Figure 1. Architecture of the ontology-driven system

V. MAPPING LINGUISTIC ANALYSIS RESULTS TO OWL STRUCTURES

The module for linguistic processing of scientific natural language texts (PDF) and their mapping to OWL structures (LP→OWL) formalizes the transformation of candidate triples and linguistic analysis outputs into RDF/OWL assertions consistent with a domain OWL ontology. To this end, the module relies on (i) a mapping dictionary between lexemes/terms and ontology URIs and (ii) a set of mapping rules that take into account part of speech, grammatical features, local context, and previous mapping decisions [19–21].

The declarative specification of rules makes it possible to update mappings without modifying the module code and to maintain multiple mapping variants for different domains within a single architecture. In cases where multiple URI candidates are possible for a single linguistic annotation, the module constructs an ordered list of weighted hypotheses and applies a conflict-resolution policy that considers domain/range constraints and existing ABox assertions. The output is a sequence of RDF triples augmented with provenance annotations (PROV metadata) that link assertions to text spans, pipeline run parameters, and the specific rules applied. When ambiguities are detected or constraints are violated, the module generates diagnostic messages that can be used both to refine the mapping dictionary and rules and to reprocess problematic text fragments.

Ontology, Logical Reasoning, and SHACL Validation. The ontology specifies domain classes, object and data properties, domain and range restrictions, cardinality axioms, and Semantic Web Rule Language (SWRL) rules for additional inferences. The terminological level (TBox) defines class and property hierarchies and serves as a global schema for all data, whereas the assertional level (ABox) contains statements about specific individuals obtained from the linguistic pipeline. A reasoning engine (e.g., Pellet or HermiT) performs ontology classification, materialization of entailed facts (deriving subclass relations, individual types, and additional links via SWRL rules), and detection of inconsistencies between the TBox and the ABox. The system supports both full and incremental reasoning modes, enabling a trade-off between computational cost and the freshness of materialized facts across different experimental scenarios.

To verify the structural correctness of the knowledge graph, the system employs SHACL. SHACL shapes define admissible node and edge patterns, including cardinality constraints, required types, and permissible combinations of properties, and the resulting triples are validated against these constraints. Detected violations are recorded as detailed validation reports, which are integrated into the computation of the Ontology-Consistency Score and provide a basis for improving both the ontological model and the LP→OWL mapping rules [24, 25, 27–29].

VI. KNOWLEDGE GRAPH AND THEMATIC DATA CORPORA

The knowledge graph is deployed on an Apache Jena Fuseki server with SPARQL 1.1 support [30–31]. On top of the base graph, thematic data corpora are constructed—specialized subgraphs of the knowledge graph optimized for specific analytical and reporting scenarios. In addition, mechanisms for caching the results of typical queries and for version control of graphs are implemented, which enables tracking the evolution of knowledge over time.

A dedicated component is responsible for knowledge-graph versioning: time-stamped "snapshots" of the ABox state and, when required, the TBox are recorded, which makes it possible to monitor knowledge evolution and reproduce experimental results. Knowledge-graph versions are also used to compare different configurations of the linguistic pipeline and the LP→OWL mapping rules, since each run produces a separate named subgraph with associated metadata.

Access to the knowledge graph is provided via a remote SPARQL endpoint, which is used both by external systems and by NLP→SPARQL modules to execute automatically generated queries.

From an architectural and structural perspective, the module that transforms a natural-language query into a SPARQL query should be implemented as a separate service with a well-defined interface. The service takes as input the natural-language query text and its execution context (in particular, the query language, the domain, and the selected ontology profile), and returns an intermediate structured representation of the query, the generated SPARQL query, and an explanation of which interpretation rules and ontology-element matching procedures were applied. For practical deployment, additional constraints are specified, including execution timeouts, response-size limits (e.g., LIMIT), constraints on the number of generated triples, and event logging, which ensures reproducibility and facilitates error tracing during debugging.

A unified programming interface enables integration of the proposed knowledge graph with other ontological resources, publication repositories, and services for analytical processing of scientific data. The chosen deployment architecture—hosting the knowledge graph on Apache Jena Fuseki and using a remote SPARQL endpoint—conceptually extends solutions proposed in the instrumental suite and ontology-based system for processing scientific publication databases [16]. Unlike that work, in the present paper the knowledge graph is treated not only as a metadata store but also as a target model for an ontology-driven linguistic pipeline and subsequent logical reasoning. This infrastructure organization ensures consistency across the levels "text (PDF) → ontology → reasoning → SPARQL queries" and facilitates transferring the proposed solution to other domains.

VII. METHODS AND ALGORITHMS

The S/P/O triple construction algorithm comprises the following sequence of stages: sentence segmentation and removal of technical artifacts; lemmatization and morphosyntactic analysis; identification of predicative constructions; normalization of predicates to the infinitive; extraction of subject and object with respect to syntactic dependencies and grammatical features; coreference resolution; generation of weighted candidate triples with references to text spans and applied rules; and application of quality filters (length constraints, stop-lexicon removal, and semantic compatibility checks) [18, 19].

Ontology mapping validation relies on a term-to-URI mapping dictionary and the LP→OWL mapping rules. Role conflicts are resolved using a priority policy that favors mappings that do not violate domain and range constraints. For terms with multiple potential matches in the ontology, additional heuristics are applied that take into account local context, the type of syntactic construction, previously accepted mapping decisions within the document, and frequency characteristics in the domain corpus.

SHACL validation is applied to the generated triples; elements that violate any constraints are either removed or re-submitted to the linguistic processor for reprocessing, with the detected violation type explicitly annotated. For a subset of constraints, a soft-constraint mode is used, in which triples are flagged as suspicious but are not automatically removed, enabling analysis of potentially useful–albeit atypical–knowledge structures. [25].

Based on the validation results, a report of typical violations (domain/range errors, cardinality violations, missing required types, etc.) is produced and used to refine the ontology, the mapping dictionary, and the mapping rules. Aggregate success indicators for SHACL checks are incorporated into the computation of the OCS metric, enabling quantitative assessment of how changes in the linguistic pipeline and the ontological model affect the overall consistency of the knowledge graph.

VIII. COMPUTATION OF OCS AND ΔF1_REASONING

The OCS metric is defined as the ratio of the number of triples that successfully pass all ontological and structural checks to the total number of triples in the knowledge graph. In practice, this means that for each processing configuration (B1–B4), a subgraph is constructed to which TBox constraints, SHACL shapes, and, when necessary, additional integrity rules are applied sequentially. The proportion of triples that do not violate any of these constraints is then computed. The OCS value is calculated separately before and after applying logical reasoning, which makes it possible to assess whether reasoning improves (or, in boundary cases, degrades) the overall consistency of the knowledge graph.

ΔF1_reasoning is defined as the difference between the F1-score before and after applying logical reasoning and fact materialization. The F1-score is computed using the standard combination of precision and recall at the level of ⟨S, P, O⟩ triples under an exact-match criterion, i.e., a triple is considered correct only if the subject, predicate, and object are all simultaneously correct. A positive ΔF1_reasoning value is interpreted as evidence that logical reasoning and ontological validation help eliminate false or inconsistent triples and/or infer additional correct facts via materialization. A zero or near-zero ΔF1_reasoning indicates that reasoning has little effect on the quality of extracted triples, whereas a negative value signals overly strict rules or incorrect constraints in the ontology.

In the proposed method, both indicators–OCS and ΔF1_reasoning–are computed for each configuration B1–B4 and are further aggregated using k-fold cross-validation, which reduces the impact of random corpus fluctuations. The obtained values are summarized in Tables 1 and 2, demonstrating that the ontology-driven configuration B4 yields the largest gains both in knowledge-graph consistency and in the post-reasoning F1-score.

IX. JUSTIFICATION AND NLP→SPARQL

Justifications are generated using a justification (explanation) mechanism that identifies minimal sets of axioms sufficient to entail a given assertion or to detect an inconsistency. The resulting minimal sets are stored together with references to the originating triples and applied rules, which makes it possible to reproduce the reasoning chain and localize the source of an error in the ontology or data. Explanations are aligned with the PROV provenance model: for each entailed assertion, the system records which input triples, TBox axioms, SWRL rules, and reasoning steps were involved. This supports both machine-oriented decision logs and human-readable reports for expert analysis [26].

Within the "natural-language processing → SPARQL queries" setting, the system employs query templates linked to ontological classes and properties. Natural-language queries are analyzed by the linguistic processor; the key entities and relations are mapped to ontology URIs; and an appropriate SPARQL query over the knowledge graph is then constructed. To reduce ambiguity, ontological constraints (property domain/range, class hierarchies) are leveraged to discard semantically invalid interpretations of the query. The generated SPARQL queries are additionally annotated with references to elements of the original query text, which facilitates interactive debugging and potential user-side editing. In the longer term, this provides a foundation for building "query–clarification" dialog interfaces, in which the user receives not only an answer from the knowledge graph but also a concise explanation of how the natural-language query was interpreted at the ontology and SPARQL levels [6–14].

In practical natural-language interfaces to knowledge graphs, most user queries can be reduced to a limited set of intents: LIST (return a list of objects) and COUNT (return a count), sometimes with GROUP BY (aggregation by year/journal) and ORDER BY (sorting). Therefore, it is reasonable to start by formalizing these intents and common filters, including: topic (dcterms:subject), author (dcterms:creator → foaf:Person → foaf:name), year/date (dcterms:date or dcterms:issued), identifiers (bibo:doi), and links (bibo:uri / foaf:page).

To decouple natural-language analysis from SPARQL generation, an intermediate representation (IR) is introduced— a structure that accumulates query "slots." A minimal set of slots for bibliographic knowledge graphs includes: Intent (LIST/COUNT), Type (Article/Journal/Person), Subject (topic), AuthorRegex (a regular expression for name matching), YearFrom/YearTo (year range), GroupByYear (group by year), NeedPdf (require a link/PDF), Sort (ASC/DESC), and Limit (Top-K bound). The IR can be easily logged, tested on gold-standard (NL, SPARQL) pairs, and explained to the user ("how the system interpreted the query") [7–9, 11, 12].

A. APPROACHES TO NLP→SPARQL TRANSFORMATION: FROM RULES TO LLMS

The initial and most controllable approach is rule-based processing using rules and templates, for example: "how many" → COUNT, "after 2020" → YearFrom = 2020, "between 2021 and 2024" → YearFrom/YearTo, "by" → AuthorRegex, and "about" → Subject. Next, entity linking is introduced to map mentions to URIs via label indexes (e.g., rdfs:label, skos:altLabel, foaf:name). At the subsequent level, a query graph is constructed and compiled into SPARQL. To increase coverage, retrieval-based methods are used (searching for similar NL↔SPARQL examples), as well as LLMs with controlled decoding followed by validation [8–12].

B. DETERMINISTIC SPARQL COMPILATION FROM IR AND CONSISTENCY CONTROL

A key element of a practical solution is a deterministic SPARQL compiler that constructs queries exclusively from a predefined set of permitted templates. For example, for a LIST query with the slots Subject + Author + YearFrom/YearTo, the compiler generates the core of the WHERE clause with explicit typing and metadata joins, and then adds the corresponding FILTER conditions [24, 25].

Listing (SPARQL fragment)

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX bibo:    <http://purl.org/ontology/bibo/>
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX xsd:     <http://www.w3.org/2001/XMLSchema#>

SELECT ?title ?year ?doi ?pdf WHERE {
  ?paper a bibo:AcademicArticle ;
         dcterms:title ?title ;
         dcterms:date ?date .
  OPTIONAL { ?paper bibo:doi ?doi }
  OPTIONAL { ?paper bibo:uri ?pdf }
  OPTIONAL { ?paper dcterms:subject ?topic }
  OPTIONAL { ?paper dcterms:creator ?a . ?a foaf:name
?name }
  BIND(SUBSTR(STR(?date),1,4) AS ?year)
  # FILTER(...) conditions are instantiated from IR slots
} ORDER BY DESC(?year) LIMIT 50
```

Ontology constraints (domain/range restrictions, cardinality constraints, class hierarchies) and SHACL validation can be used not only for knowledge-graph quality control but also as an additional mechanism for ranking candidate mappings during query generation: semantically invalid interpretations are discarded before the SPARQL query is executed.

For security and reproducibility, it is recommended to: (1) use a whitelist of prefixes and SPARQL constructs; (2) avoid concatenating "raw" user text into SPARQL and instead substitute only slot values; (3) enforce timeouts and limits; and (4) return explanations indicating which rules and mappings filled specific slots and which query fragments served as evidence for the corresponding FILTER conditions.

In the developed Java prototype WebInterface (NetBeans + Tomcat), these principles are implemented via a chat-style interface (input field + button) and a servlet controller that invokes the rules-and-slots module to build the IR, followed by the deterministic SPARQL compiler. This "first pass" provides a controllable baseline: it can be readily extended with entity linking, retrieval-based examples, or an LLM component without changing the IR interface or the safe-compilation rules.

C. EXPERIMENTAL METHODOLOGY

The experimental corpus consisted of two computer science research articles: Article [32] (sentences 1–227, approximately 1,000 ⟨S, P, O⟩ triples) and Article [33] (sentences 1–221, approximately 700 triples), yielding a total of 448 sentences and approximately 1,700 ⟨S, P, O⟩ triples. The stages of preliminary text normalization were specified in detail, and the expert annotation procedure was described, noting that annotation was performed by two independent experts followed by an adjudication step to resolve disputed cases. It is further explained that these reconciled annotations were used to construct a gold standard for evaluating triple-extraction quality and subsequent logical reasoning.

D. BASELINE CONFIGURATIONS

To evaluate the proposed approach, three conditionally empirical baseline configurations (B1, B2, B3) and the proposed system (B4) were considered. In computer science, the term conditionally empirical studies is used to denote works in which computer simulation methods or synthetic data are combined with empirical approaches:

B1 – a statistical information extraction approach that does not use an ontology or logical reasoning;

B2 – a neural relation-extraction approach based on modern deep learning models;

B3 – a hybrid approach combining statistical and neural components, but without ontological validation;

B4 – full ontology-driven processing implementing the pipeline LP → OWL → reasoning engine → knowledge graph / SPARQL.

E. EVALUATION METRICS AND RESULTS

The evaluation is conducted using the following indicators:

- Precision, recall, and F1-score for ⟨S, P, O⟩ triples;
- An exact-match criterion for triples, which requires the subject, predicate, and object to be simultaneously correct;
- OCS, defined as the proportion of facts (triples) that pass consistency checks at the TBox and SHACL levels;
- ΔF1_reasoning, i.e., the gain in F1-score after applying logical reasoning and fact materialization;
- Latency of SPARQL query execution and reasoning time across different configurations.

This combination of metrics makes it possible to assess both the quality of initial triple extraction and the contribution of ontological validation and logical reasoning to improving the consistency and practical usefulness of the knowledge graph.

Table 1 presents a comparison of ⟨S, P, O⟩ triple extraction quality before applying logical reasoning.

Table 1. Triple extraction quality before logical reasoning

Configuration	Precision	Recall	F1-score	Exact match
B1 — statistical IE	0.74	0.69	0.71	0.68
B2 — neural RE	0.70	0.74	0.72	0.69
B3 — hybrid (no ontology)	0.78	0.76	0.77	0.74
B4 — ontology-driven pipeline	0.84	0.85	0.84	0.80

The table shows that configuration B4 achieves the highest values of both the F1-score and exact-match accuracy, indicating improved quality and stability of the extracted triples due to the use of ontological validation and stricter consistency control.

The further contribution of logical reasoning to improving the knowledge-graph consistency and triple quality is presented in Table 2.

Table 2. Impact of logical reasoning on consistency and extraction quality

Configuration	OCS (before)	OCS (after)	ΔOCS	F1 (before)	F1 (after)	ΔF1 (reasoning)
B1	0.65	0.65	0.00	0.71	0.71	0.00
B2	0.70	0.74	0.04	0.72	0.74	0.02
B3	0.78	0.83	0.05	0.77	0.80	0.03
B4	0.88	0.94	0.06	0.84	0.89	0.05

For configuration B4, the largest gains are observed in both OCS and the F1-score after logical reasoning, which reflects the effect of structural validation and the materialization of entailed facts based on the ontology.

Table 3 separately evaluates the performance of logical reasoning and knowledge-graph query execution.

Table 3. Reasoning and query performance

Configuration	Reasoning time per 10,000 triples (s)	Average SPARQL query latency (ms)
B1	0.0	95
B2	0.0	100
B3	1.8	120
B4	2.4	140

As shown in Table 3, configuration B4 exhibits a moderate increase in both reasoning time and SPARQL query latency compared to B1–B3; however, this overhead is compensated by a substantial improvement in the quality and consistency of the knowledge graph.

F. Statistical Analysis

To assess the statistical significance of differences between configurations, k-fold cross-validation was employed. For pairwise comparison of F1-scores, the McNemar test was applied to paired classification errors for triple extraction.

In addition, an error analysis by type was conducted, including:

- incorrect role assignment (subject/object);
- lemmatization and lexeme-normalization errors;
- conflicts with ontology property domain/range constraints;
- errors caused by unresolved or incorrectly resolved anaphora (coreference).

The obtained results make it possible not only to compare averaged performance metrics but also to identify typical sources of errors both at the level of the linguistic processor and at the level of ontological validation.

X. DISCUSSION

The obtained results demonstrate that the ontology-driven processing pipeline—"linguistic processor → OWL ontology → reasoning engine → knowledge graph → natural-language queries → SPARQL queries"—provides a substantial increase in both the quality and consistency of knowledge compared to baseline configurations without ontological validation. Configuration B4 consistently outperforms B1–B3 in terms of the F1-score and OCS, which confirms the practical value of integrating linguistic analysis with ontology-based logical reasoning.

The main advantages of the proposed approach include: measurable consistency enabled by the introduced OCS metric; explainability supported by the justification mechanism and decision logs that make it possible to trace which axioms and facts led to a specific entailment or to an inconsistency detection; reproducibility ensured by well-defined interface specifications, ontology and knowledge-graph version control, and a formalized evaluation protocol; and readiness for NLP→SPARQL integration, since the knowledge graph and the ontological model can serve as a target schema for natural-language query generation.

At the same time, the approach has several limitations. The quality and stability of the results depend strongly on the completeness and correctness of the domain ontology and on mapping dictionaries that link linguistic annotations to ontological entities. The initial cost of building and maintaining the TBox can be substantial for complex domains. Moreover, applying logical reasoning over large knowledge graphs increases computational cost and query latency. Possible ways to mitigate these limitations include incremental reasoning, partitioning the knowledge graph into fragments with local reasoning and controlled integration of results, and caching typical SPARQL query results and pre-materialized subgraphs.

Overall, the results support treating the ontology as a controlling component of the knowledge life cycle and demonstrate the practical effectiveness of the ontology-driven approach to processing scientific texts in computer science.

A practical application scenario for the proposed method is digital scientific libraries and publication repositories. In this setting, an ontology-consistent knowledge graph constructed from full-text articles serves as a semantic layer over existing bibliographic metadata and enables NLP→SPARQL querying over the publication collection: from simple queries for works describing a particular method or problem to complex composite queries that incorporate task types, domains, employed models, experimental corpora, and achieved metric values. Representing such characteristics as S/P/O triples within a unified knowledge graph supports faceted navigation, analytical querying of experimental results, and transparent explanations of how a natural-language query was interpreted by the ontology model during answer generation.

XI. CONCLUSIONS

Summarizing the results of the proposed approach to ontology-driven processing of scientific texts, the following key improvements can be highlighted:

A consistent knowledge graph was constructed with high values of the Ontology-Consistency Score (OCS) and F1, and the feasibility of using these indicators as quality-control instruments for the knowledge graph and processing configurations was demonstrated, as supported by the results in Tables 1–2.

PROV metadata were integrated into the knowledge graph, enabling the generation of explanations both for an ontology expert (a domain knowledge engineer) and for an end user. The explanations are based on minimal sets of axioms and facts and explicitly reflect the steps and outcomes of logical reasoning.

Semantic search and navigation over publications were implemented on top of the ontology-consistent knowledge graph, making it possible to move from keyword-based search to queries oriented toward elementary senses, task types, methods, datasets, and metric values.

The possibility of integration with modern LLM-based systems was demonstrated, where the developed ontology-driven pipeline serves as a structured ontological scaffold. An LLM can be employed at the linguistic-analysis stages (annotation and ⟨S, P, O⟩ triple extraction), while the ontology, logical reasoning, SHACL validation, and the OCS/ΔF1_reasoning metrics provide quality control, consistency enforcement, and filtering of erroneous triples.

The transferability of the method to other domains was demonstrated, provided that appropriate domain OWL ontologies and mapping dictionaries are available. The pipeline architecture "LP → OWL → reasoning engine → knowledge graph → SPARQL" is domain-agnostic and can be adapted, in particular, for scientific digital libraries, domain-specific publication repositories, and other knowledge-oriented information systems.

References

[1] O. Palagin, M. Petrenko, K. Malakhov, "Challenges and role of ontology engineering in creating the knowledge industry: A research-related design perspective," *Cybernetics and Systems Analysis*, vol. 60, issue 4, pp. 633–645, 2024. https://doi.org/10.1007/s10559-024-00702-6.

[2] J.V. Rogushina, A.Y. Gladun, O.V. Anishchenko, S.M. Pryima, "Semantic technologies as a tool of information support for professionalization of andragogues," *Problems in Programming*, no. 2–3, pp. 441–448, 2024. https://doi.org/10.15407/pp2024.02-03.441.

[3] N. Guarino, (Ed.). Formal Ontology in Information Systems: Proceedings of the 1st International Conference FOIS'98, Trento, Italy, June 6–8, 1998. Amsterdam: IOS Press, 1998.

[4] A. Gómez-Pérez, M. Fernández-López, O. Corcho, *Ontological Engineering*, London: Springer, 2004. https://doi.org/10.1007/b97353.

[5] M. Petrenko, E. Cohn, O. Shchurov, K. Malakhov, "Ontology–driven computer systems: elementary senses in domain knowledge processing," *South African Computer Journal*, vol. 35, issue 2, pp. 127–144, 2023. https://doi.org/10.18489/sacj.v35i2.17445.

[6] V. López, C. Unger, P. Cimiano, E. Motta, "Evaluating question answering over linked data," *Journal of Web Semantics*, vol. 21, pp. 3–13, 2013. https://doi.org/10.1016/j.websem.2013.05.006.

[7] E. Dimitrakis, K. Sgontzos, Y. Tzitzikas, "A survey on question answering systems over linked data and documents," *Journal of Intelligent Information Systems*, vol. 55, issue 2, pp. 233–259, 2020. https://doi.org/10.1007/s10844-019-00584-7.

[8] S. Ferré, "Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language," *Semantic Web*, vol. 8, issue 3, pp. 405–418, 2017. https://doi.org/10.3233/SW-150208.

[9] D. Diefenbach, J. Giménez-García, A. Both, K. Singh, P. Maret, "QAnswer KG: Designing a portable question answering system over RDF data," In: *The Semantic Web – ESWC 2020. LNCS 12123*. Springer, 2020, pp. 429–445. https://doi.org/10.1007/978-3-030-49461-2_25.

[10] T. Soru, E. Marx, D. Moussallem, A. Valdestilhas, D. Esteves, et al., "SPARQL as a Foreign Language," arXiv preprint, 2017. https://arxiv.org/abs/1708.07624.

[11] X. Yin, D. Gromann, S. Rudolph, "Neural machine translating from natural language to SPARQL," *Future Generation Computer Systems*, vol. 117, pp. 510–519, 2021. https://doi.org/10.1016/j.future.2020.12.013

[12] M.A. Borroto, F. Ricca, "SPARQL-QA-v2 system for knowledge base question answering," *Expert Systems with Applications*, vol. 229(A), Art. 120383, 2023. https://doi.org/10.1016/j.eswa.2023.120383.

[13] P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann, "LC-QuAD: A corpus for complex question answering over knowledge graphs," In: *The Semantic Web – ISWC 2017*, Springer, 2017, pp. 210–218. https://doi.org/10.1007/978-3-319-68204-4_22.

[14] M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, "LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia," In: *The Semantic Web – ISWC 2019, LNCS 11779*. Springer, 2019, pp. 69–78. https://doi.org/10.1007/978-3-030-30796-7_5.

[15] A. Maedche, S. Staab, "Ontology learning for the Semantic Web," *IEEE Intelligent Systems*, vol. 16, issue 2, pp. 72–79, 2001. https://doi.org/10.1109/5254.920602.

[16] W.Y. Wong, W. Liu, M. Bennamoun, "Ontology learning from text: A look back and into the future," *ACM Computing Surveys*, vol. 44, issue 4, Art. 20, 2012. https://doi.org/10.1145/2333112.2333115.

[17] J. Wątróbski, "Ontology learning methods from text – An extensive knowledge-based approach," *Procedia Computer Science*, vol. 176, pp. 3356–3368, 2020. https://doi.org/10.1016/j.procs.2020.09.061.

[18] P. Cimiano, U. Reyle, J. Šarić, "Ontology-driven discourse analysis for information extraction," *Data & Knowledge Engineering*, vol. 55, issue 1, pp. 59–83, 2005. https://doi.org/10.1016/j.datak.2004.11.009.

[19] V. Pertsas, P. Constantopoulos, "Ontology-driven information extraction from research publications," In: *Digital Libraries for Open Knowledge (TPDL 2018). LNCS 11057*. Springer, 2018, pp. 241–253. https://doi.org/10.1007/978-3-030-00066-0_21.

[20] A. Hogan, E. Blomqvist, M. Cochez, et al., "Knowledge graphs," *ACM Computing Surveys*, vol. 54, issue 4, Art. 71, 2021. https://doi.org/10.1145/3447772.

[21] A. Hogan, "Knowledge graphs: Research directions," In: *Reasoning Web 2020. LNCS 12258*. Springer, 2020, pp. 223–253. https://doi.org/10.1007/978-3-030-60067-9_8.

[22] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, J. Taylor, "Industry-scale knowledge graphs: Lessons and challenges," *Communications of the ACM*, vol. 62, issue 8, pp. 36–43, 2019. https://doi.org/10.1145/3331166.

[23] J.F. Sowa, "Conceptual graphs as a universal knowledge representation," *Computers & Mathematics with Applications*, vol. 23, issues 2–5, pp. 75–95, 1992. https://doi.org/10.1016/0898-1221(92)90137-7.

[24] W3C OWL Working Group. OWL 2 Web Ontology Language: Document Overview. W3C Recommendation, 2012. [Online]. Available at: https://www.w3.org/TR/owl2-overview/.

[25] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL). W3C Recommendation, 2017. [Online]. Available at: https://www.w3.org/TR/shacl/.

[26] T. Lebo, S. Sahoo, D. McGuinness, PROV-O: The PROV Ontology. W3C Recommendation, 2013. [Online]. Available at: https://www.w3.org/TR/prov-o/.

[27] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, 2004. [Online]. Available at: https://www.w3.org/Submission/SWRL/.

[28] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, Y. Katz, "Pellet: A practical OWL-DL reasoner," *Web Semantics*, vol. 5, issue 2, pp. 51–53, 2007. https://doi.org/10.1016/j.websem.2007.03.004.

[29] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, "HermiT: An OWL 2 Reasoner," *Journal of Automated Reasoning*, vol. 53, issue 3, pp. 245–269, 2014. https://doi.org/10.1007/s10817-014-9305-1

[30] Apache Software Foundation. Apache Jena Fuseki Documentation. 2024. [Online]. Available at: https://jena.apache.org/documentation/fuseki2/.

[31] B. DuCharme, *Learning SPARQL: Querying and Updating with SPARQL 1.1*, 2nd ed. O'Reilly Media, 2013. https://doi.org/10.1089/big.2012.0004.

[32] A.V. Palagin, N.G. Petrenko, "Methodological foundations for development, formation and IT-support of transdisciplinary researches," *Problems of Control and Informatics*, vol. 50, issue 10, pp. 1-17, 2018. https://doi.org/10.1615/JAutomatInfScien.v50.i10.10.

[33] A.V. Palagin, N.G. Petrenko, "Towards the design of an ontology-driven information system with natural language processing," Mathematical Machines and Systems, no. 2, pp. 14–23, 2008. (In Russian). [Online]. Available at: https://nasplib.isofts.kiev.ua/handle/123456789/2402.

OLEKSANDR PALAGIN, *Glushkov Institute of Cybernetics of the NAS of Ukraine. Major Fields of Scientific Research: Evolutionary cybernetics*
e-mail: palagin_a@ukr.net
http://orcid.org/0000-0003-3223-1391

DMYTRO VYKHOVANETS, *Glushkov Institute of Cybernetics of the NAS of Ukraine.*
e-mail: vykhovanets.dmitriy@gmail.com,
http://orcid.org/0009-0005-7564-1552

MYKOLA PETRENKO, *Glushkov Institute of Cybernetics of the NAS of Ukraine. Major Fields of Scientific Research: AI, Semantic Web; Ontology engineering; Transdisciplinary research.*
e-mail: petrng@ukr.net,
http://orcid.org/0000-0001-6440-0706

MYKOLA BOYKO, *Glushkov Institute of Cybernetics of the NAS of Ukraine. Major Fields of Scientific Research: Semantic Web.*
e-mail:xeldag@ukr.net
http://orcid.org/0000-0003-1723-5765

• • •