Research Square

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

Neuro-Symbolic Question Answering Architecture Integrating Ontology-Constrained Query Generation

Fabio Clarizia
University of Salerno
Massimo De Santo
University of Salerno
Rocco Loffredo
rloffredo@unisa.it

University of Salerno
Benedetta Sabini
University of Salerno

Research Article

Keywords: Neuro-symbolic integration, Hybrid intelligent systems, Large language models, Knowledge-grounded generation, Ontology-constrained decoding, Retrieval-augmented generation

Posted Date: September 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10627285/v1

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

Additional Declarations: No competing interests reported.

Neuro-Symbolic Question Answering Architecture Integrating Ontology-Constrained Query Generation

Fabio Clarizia[1], Massimo De Santo[2], Rocco Loffredo[2*], Benedetta Sabini[1]

[1]Department of Humanities, University of Salerno, Via Giovanni Paolo II, 132, Fisciano, 84084, Salerno, Italy.
[2]Department of Industrial Engineering, University of Salerno, Via Giovanni Paolo II, 132, Fisciano, 84084, Salerno, Italy.

*Corresponding author(s). E-mail(s): rloffredo@unisa.it;
Contributing authors: fclarizia@unisa.it; desanto@unisa.it; bsabini@unisa.it;

Abstract

Large Language Models ground their answers in retrieved evidence, but dense vector retrieval returns passages in which facts are embedded in text rather than isolated, so the model has to reconstruct the exact value and can get it wrong. Ontologies return exact facts that can be addressed and checked one by one, but non-expert users cannot query them directly. We ask whether grounding retrieval in a formal ontology gives more precise answers than dense retrieval when both use the same generator over the same source records. We present ORACOLO, a neuro-symbolic question-answering architecture. A semantic parser generates SPARQL queries restricted to the ontology vocabulary, a symbolic engine executes them, and the outcome of that execution acts as a check on the parser: when a query returns nothing, the system falls back to dense retrieval. Evaluated on a real bibliographic knowledge base against a generator-matched dense-retrieval baseline, ORACOLO raises Token F1 from 0.29 to 0.47 (paired difference +0.18, 95% confidence interval [0.09, 0.28]). The gain comes from factual retrieval, +0.30 on single-hop and +0.20 on multi-hop questions, and is absent on questions requiring inference over the retrieved facts. It is also larger where the symbolic path succeeds (+0.21) than where the system falls back (+0.08). Faithfulness does not separate the two systems (0.89 vs. 0.85), which we argue is what that metric measures. We further document a systematic failure of answer relevancy, and an ablation study quantifying each component's contribution.

Keywords: Neuro-symbolic integration, Hybrid intelligent systems, Large language models, Knowledge-grounded generation, Ontology-constrained decoding, Retrieval-augmented generation

MSC Classification: 68T50 , 68T30 , 68T07 , 68P20 , 68T27

Acknowledgements. The authors wish to thank the University Library of Lugano for providing access to a subset of their bibliographic catalog, which served as the domain knowledge base for the experimental evaluation presented in this work.

1 Introduction

Large Language Models (LLMs) are neural sequence predictors: they generate text by sampling from a learned distribution over tokens,

and they have no internal mechanism that separates statements which are true in a given domain from statements which merely sound plausible (Ji et al. 2023; Zhao et al. 2023). Retrieval-augmented generation (RAG) addresses this by grounding generation in retrieved evidence (Lewis et al. 2020; Gao et al. 2024). In its standard form, retrieval is *dense*: passages are selected by embedding similarity, and the fact the user asked for arrives inside a paragraph of prose rather than on its own. The generator then has to pull the fact out of the passage, and this is where a fluent but subtly wrong answer can appear, even when the retrieved passage itself is correct.

Symbolic knowledge representations sit at the opposite end. A SPARQL query for the author of a work returns the author, as the object of one specific triple, with no surrounding text to interpret and a clear trace back to the source. But querying an ontology means knowing a formal query language and the vocabulary of the schema, which most users do not, and symbolic retrieval fails abruptly whenever a question does not translate cleanly into graph logic.

The complementarity between the two is what neuro-symbolic artificial intelligence starts from (Kautz 2022; d'Avila Garcez and Lamb 2023; Hitzler et al. 2022; Sheth et al. 2023). Instead of treating LLMs and ontologies as competing paradigms, we combine them so that each covers the other's weakness, following the roadmap of Pan et al. (2024). In Kautz's taxonomy (Kautz 2022) the architecture we propose is a *Neuro | Symbolic* hybrid: a neural front-end turns a natural-language question into a SPARQL query restricted to the ontology vocabulary, a symbolic engine executes it, and the resulting triples are passed to a neural generator that writes the answer. The symbolic layer is not only a knowledge store but also a *verification gate*. If a query uses vocabulary outside the ontology, or returns no result, the system detects this without having to judge the query itself, and hands control to a dense sub-symbolic retrieval path.

Our specific question follows from a methodological observation. Faithfulness to retrieved evidence, that is, whether a generated answer is supported by what was retrieved, is largely a property of the *generator*: a capable model stays faithful to its evidence whether that evidence is a SPARQL result or a dense passage. So symbolic grounding probably does not help with faithfulness. We argue that it helps with *precision*: the symbolic path returns the exact fact, while dense retrieval returns text from which the fact has to be reconstructed. We test this directly, comparing ORACOLO against a dense-retrieval baseline that uses the *same generator, embedding model, and answer prompt*, so that what varies is the representation of the same source records rather than the capability of the generator.

The results support this account, and they show where the effect comes from. On faithfulness, ORACOLO and the matched baseline are statistically indistinguishable (0.887 vs. 0.848, 95% confidence interval (CI) $[-0.03, 0.11]$), as the hypothesis predicts. On factual accuracy ORACOLO is clearly better, raising Token F1 from 0.289 to 0.471 ($+0.18$, 95% CI $[0.09, 0.28]$), and the gain comes from factual retrieval: $+0.30$ on single-hop and $+0.20$ on multi-hop questions, with no measurable difference on reasoning questions, where the answer is an inference over facts rather than a fact. Grouping the questions by the path the system actually took narrows this down further. Where the symbolic path resolves the query the advantage is $+0.21$ Token F1; where the system falls back to sub-symbolic retrieval it is $+0.08$, a factor of 2.7 smaller, even though both branches query the same ontology. What changes is *which evidence is retrieved*, and how precisely it can be addressed, not how faithfully it is put into words.

We implement and evaluate this design in ORACOLO, deployed over the bibliographic knowledge base of the University Library of Lugano. This work contributes:

- **A neuro-symbolic QA architecture** in which symbolic execution checks the queries the neural component generates, with dense retrieval taking over when that check fails.
- **A generator-matched evaluation** that separates the effect of the retrieval mechanism from that of model capacity, and shows that the gain comes from factual retrieval and that faithfulness does not tell the two systems apart once the generator is held fixed.
- **A path-stratified analysis** showing that the gain is 2.7 times larger on questions resolved by the verified symbolic path than on those handled by the sub-symbolic fallback, although both query the ontology.

- **An ablation study and a failure-mode analysis** of neural query generation over closed vocabularies, together with a deployment study on a real institutional catalog.

The paper is organized as follows. Section 2 reviews related work. Section 3 describes the architecture. Section 4 presents the deployment domain. Section 5 reports the evaluation, including the per-path and ablation analyses. Section 6 discusses limitations and failure modes. Section 7 concludes.

2 Related Work

2.1 Neuro-Symbolic Integration of Learning and Reasoning

Combining neural learning with symbolic reasoning is one of the oldest research programs in artificial intelligence, and the limitations of large neural models have brought renewed attention to it (d'Avila Garcez and Lamb 2023; Hitzler et al. 2022). Neural systems learn robust statistical regularities from data, but they store what they know implicitly, in weight patterns that cannot be inspected or checked one by one. Symbolic systems represent knowledge explicitly and can verify it, but they break easily when the input varies and are expensive to build and maintain (Sheth et al. 2023). The neuro-symbolic program looks for architectures where each paradigm covers the other's weaknesses (Bhuyan et al. 2024; Yu et al. 2023).

Kautz (2022) proposed a widely used taxonomy of neuro-symbolic systems, organized by how the neural and symbolic components are coupled. It ranges from standard deep learning over symbolic input and output (*Symbolic Neuro Symbolic*) to tightly coupled systems where logical rules shape the network architecture itself (*Neuro[Symbolic]*). The pattern relevant here is *Neuro | Symbolic*, where a neural and a symbolic component solve different tasks and communicate through their inputs and outputs: the neural component maps sub-symbolic input to a symbolic representation, and a symbolic engine computes over it exactly. van Bekkum et al. (2021) take a different view, describing hybrid systems as assemblies of reusable design patterns. In their terms, the architecture presented here combines a *translation* pattern (natural language to formal query) with an *inference* pattern (deductive query execution) and a learning-based fallback.

With LLMs, this program has taken a concrete form: integrating LLMs with knowledge graphs and ontologies (Pan et al. 2024; DeLong et al. 2024). Surveys of this area (DeLong et al. 2024; Colelough and Regli 2025; Wang et al. 2025) describe a range of integration depths, from knowledge-enhanced pretraining to grounding at inference time. This work sits at the inference-time end. The ontology is not compiled into the model but consulted at answer time, queried, and used to check what the model produced. The symbolic knowledge therefore stays separate from the model, can be audited, and can be updated without retraining, which matters in institutional deployments.

2.2 Knowledge-Grounded Retrieval-Augmented Generation

RAG grounds neural text generation in evidence retrieved from an external source, which improves factuality and reduces hallucination (Lewis et al. 2020; Gao et al. 2024). In its standard dense form, RAG retrieves text chunks by embedding similarity. This is a sub-symbolic operation with good recall but no structural guarantees: a retrieved chunk can be on topic and still not contain the fact the question asks for, and nothing in what the generator receives marks the difference between a verified fact and a coincidental co-occurrence.

A second generation of systems adds graph structure to retrieval. GraphRAG (Edge et al. 2024) builds entity and relation graphs from source documents and retrieves community summaries; LightRAG (Guo et al. 2025) combines graph-structured indexing with a lightweight dual-level retrieval scheme; OG-RAG (Sharma et al. 2025) grounds retrieval in domain ontologies using hypergraph representations of factual clusters. Surveys of graph-based RAG (Peng et al. 2025) report consistent gains over dense-only baselines on multi-hop and relational questions. The knowledge graph question answering (KGQA) literature has studied answering natural-language questions over structured knowledge for longer (Lan et al. 2023), and recent work shows that grounding LLM responses in explicit graph structure improves

accuracy (Cai et al. 2025; Liu et al. 2025), although queries that do not line up with the graph structure remain a weakness (Liu et al. 2025).

These systems all use structure for the same purpose: graphs and ontologies *enrich the context* handed to the neural generator. None of them uses symbolic execution to *check* what the neural component produced, and none offers a per-query criterion for deciding whether the structured or the dense result should be trusted. They therefore inherit the failure profile of their retrieval layer: when the structured index does not cover a question, they degrade without saying so.

2.3 Neural Semantic Parsing over Closed Vocabularies

A third line of work treats the interface between natural language and structured knowledge as a semantic parsing problem: translating questions into executable formal queries, in our case SPARQL. LLM-based NL2SPARQL systems produce syntactically valid queries even for complex patterns (Emonet et al. 2025), and conditioning generation on the target ontology measurably improves correctness (Jiang et al. 2025); recent work has applied these techniques to event-based and cultural-heritage knowledge graphs (Mountantonakis and Tzitzikas 2025; Loffredo and De Santo 2024). Fine-tuning approaches such as ChatKBQA (Luo et al. 2024) reach high accuracy, but they need domain-specific training data, which institutional ontologies usually do not have. At the decoding level, constrained generation (Willard and Louf 2023) restricts the output token distribution to a formal grammar and so guarantees syntactic validity. It cannot guarantee *semantic* validity against a specific schema: a well-formed query can still reference vocabulary that sounds plausible but does not exist in the ontology, a failure mode we examine in Section 6.3.

What all these systems have in common is that they depend on the translation succeeding. If the generated query fails or returns nothing, the system returns no answer (Ali et al. 2022) and has no way to recover. This is where a purely symbolic pipeline gives up the robustness that motivated using neural components in the first place.

2.4 Positioning

ORACOLO sits at the intersection of these three threads, and what distinguishes it from each is the role it gives the symbolic component.

Knowledge-grounded RAG systems (Section 2.2) use graph structure to enrich the context handed to a neural generator. ORACOLO instead uses symbolic execution as a *verification gate*: a generated query either stays within the vocabulary of the ontology and returns triples that can be checked, or it fails visibly, and control passes to the sub-symbolic path. Pure semantic parsing systems (Section 2.3) return no answer when the translation fails. ORACOLO does not stop there: the dense fallback keeps recall in exactly those cases where the symbolic path cannot deliver precision.

In Kautz's taxonomy (Section 2.1), ORACOLO is a loosely coupled *Neuro | Symbolic* system in which control moves between a symbolic and a sub-symbolic retrieval path. As far as we know, this design point has not been measured on a production knowledge base, either for the precision-recall trade-off it involves or for what each component contributes. Providing those measurements, through a category-stratified evaluation and an ablation study, is the empirical contribution of this paper.

3 Methodology

3.1 Architecture Overview

ORACOLO is a neuro-symbolic question-answering architecture built as a loop between neural and symbolic components. The neural components handle everything that involves natural language: query decomposition, semantic parsing, and answer synthesis. The symbolic component, the ontology together with its SPARQL engine, handles retrieval, and it is also what verifies the result. It is the only part of the pipeline whose output is guaranteed to come from the knowledge base. Figure 1 shows the architecture end to end [1].

Formally, let $\mathcal{O}$ denote the domain ontology with vocabulary $V_{\mathcal{O}}$ (the set of its class and

[1] The use of a generative AI assistant in the preparation of this manuscript is documented in the Statements and Declarations section.

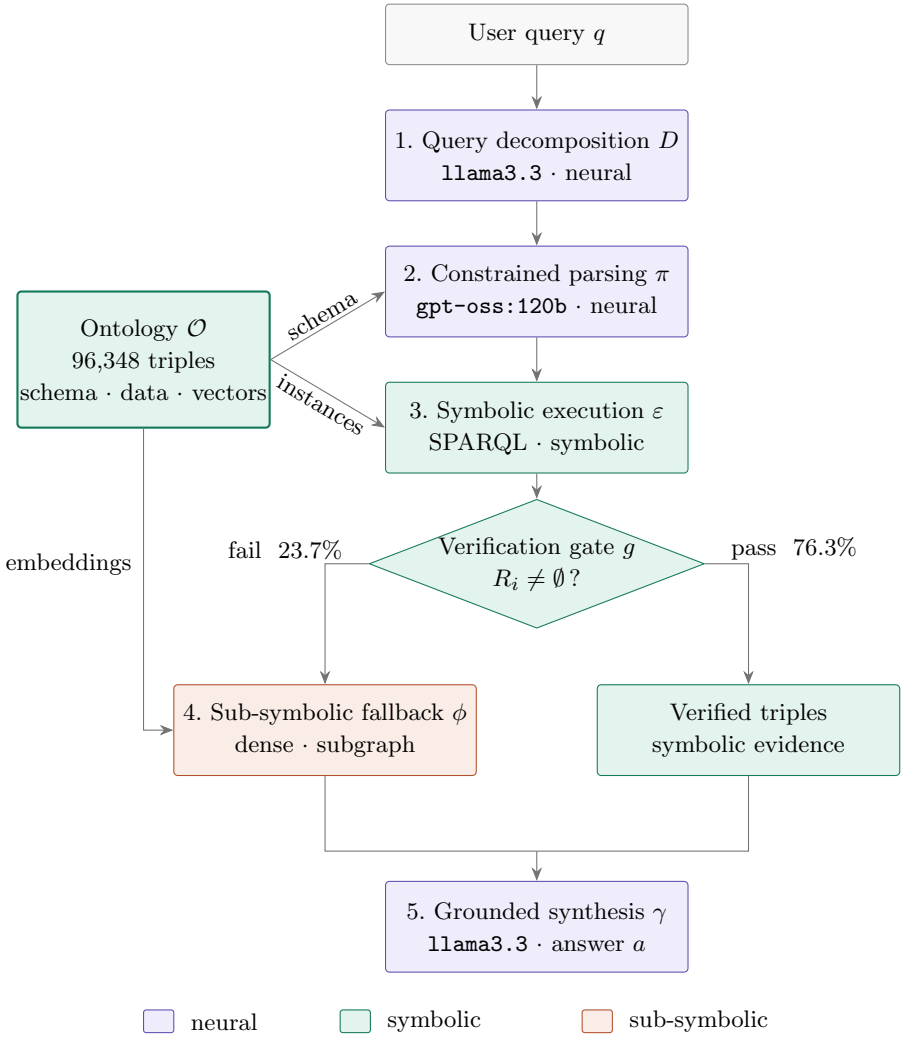


Fig. 1 The ORACOLO neuro-symbolic architecture. The neural components (query decomposition, constrained semantic parsing, answer synthesis) handle everything that involves natural language. The symbolic component (the ontology and its SPARQL engine) retrieves the evidence and checks the parser's output, with a dense sub-symbolic path as fallback. One ontology does three jobs: it supplies the closed vocabulary that constrains parsing, the instance data that execution queries, and the embedded content the fallback searches over, so both branches stay anchored to the ontology. The percentages are the gate outcomes observed on the evaluation sample ($n = 97$). Every query reaches retrieval: there is no path that bypasses the knowledge base.

property IRIs), and let q be a natural-language user query. The pipeline computes the answer a through the following stages:

1. **Neural query decomposition.** $D\colon q \mapsto \{q_1, \ldots, q_k\}$, splitting multi-faceted questions into independent atomic sub-queries, with $D(q) = \{q\}$ when q is atomic (Section 3.2).
2. **Constrained neural semantic parsing.** $\pi\colon q_i \mapsto s_i$, mapping each sub-query to a SPARQL query s_i under explicit closed-world constraints that restrict generation to $V_{\mathcal{O}}$ (Section 3.3).
3. **Symbolic execution and verification.** $\varepsilon\colon s_i \mapsto R_i$, executing s_i over $\mathcal{O}$ and returning a (possibly empty) result set R_i of triples. Testing whether R_i is empty gives a *verification gate* $g(R_i) \in \{\mathsf{pass}, \mathsf{fail}\}$ on the parser's output (Section 3.4).
4. **Sub-symbolic fallback retrieval.** $\phi\colon q_i \mapsto E_i$, used only when $g(R_i) = \mathsf{fail}$, retrieving evidence by dense embedding similarity over

the ontology with local subgraph expansion (Section 3.4).

5. **Grounded neural answer synthesis.** $\gamma\colon \big(q,\ \{(q_i, E_i)\}_{i=1}^{k}\big) \mapsto a$, combining the evidence collected for each sub-query into a single natural-language answer (Section 3.5).

Each paradigm is kept to the task it is suited for. The neural components never *assert* domain facts: they translate and they verbalise. The symbolic component never *interprets* language. The verification gate is where the two meet, turning a fact about the symbolic side, whether the result set is empty, into a control decision for the neural pipeline. This is a loosely coupled *Neuro | Symbolic* integration in the sense of Kautz's taxonomy.

One consequence of this arrangement sets ORACOLO apart from architectures that route queries adaptively: *every* query reaches retrieval. No stage is allowed to decide that a question can be answered without consulting the knowledge base. We chose this deliberately, and Section 5.5 reports the evidence behind the choice.

In the implementation, stages 1 and 5 use a general-purpose instruction-tuned model and stage 2 a larger model reserved for semantic parsing. Stage 3 is handled by the `OntologyExplorer` module, an in-memory RDF graph with a SPARQL engine, and stage 4 by a vector store. Using different models is a resource decision: decomposition and verbalization are shallow linguistic tasks, whereas parsing into a closed vocabulary needs the strongest reasoning capability we have available. Algorithm 1 summarizes the pipeline.

A stage we tested and removed.

An earlier version of this architecture had an intent-classification stage before decomposition. It labeled each query as either direct or kb and sent the direct ones straight to the generator, without retrieval. We built it in the same way as the rest of the pipeline, constraining its output with a JSON Schema so that the routing decision was easy to consume downstream, and the idea behind it was simply that greetings and questions about the system itself do not need the catalog. The evaluation did not bear this out. The classifier sent a substantial share of catalog lookups to parametric answering, and its net effect on end-to-end accuracy was significantly negative (Section 5.5). We follow the measurement rather than the intuition, and the deployed architecture has no such stage. The case is worth recording because it bears on the wider question this paper asks, and we come back to it in Section 6: a routing decision taken before retrieval is taken without evidence, and nothing in the architecture can check it afterwards.

Algorithm 1 ORACOLO neuro-symbolic QA pipeline

Require: user query q, ontology $\mathcal{O}$ with vocabulary $V_{\mathcal{O}}$, vector index $\mathcal{X}$, confidence threshold $\tau \in [0, 1]$
Ensure: natural-language answer a

$\{q_1, \ldots, q_k\} \leftarrow D(q)$ ▷ neural query decomposition
for $i \leftarrow 1$ **to** k **do** ▷ sub-queries are independent
  $s_i \leftarrow \pi(q_i \mid V_{\mathcal{O}})$ ▷ constrained neural semantic parsing
  $R_i \leftarrow \varepsilon(s_i, \mathcal{O})$ ▷ symbolic execution over ontology
  if $R_i \neq \emptyset$ **then** ▷ verification gate g: pass
    $E_i \leftarrow R_i$ ▷ symbolic evidence: verified triples
  else ▷ verification gate g: fail
    $E_i \leftarrow \phi(q_i, \mathcal{X}, \tau)$ ▷ sub-symbolic fallback
  end if
end for
$a \leftarrow \gamma\big(q,\ \{(q_i, E_i)\}_{i=1}^{k}\big)$ ▷ grounded synthesis
return a

3.2 Neural Query Decomposition

Queries first go to the `QuerySplitter` component, which decides whether a query can be broken into independent sub-queries, each answerable by a single retrieval operation. The reason has to do with what comes next in the pipeline. A single SPARQL query over a property-rich ontology gets unwieldy when it has to cover several loosely related information needs at once, and the more complex the query, the less likely the neural parser is to get it syntactically and semantically right. Decomposition therefore turns one hard parsing problem into several easier ones, at the cost of one extra inference call per query.

Whether that trade pays off depends on how often genuinely decomposable queries come up. On our benchmark they turn out to be rare enough

that the stage does not pay for itself (Section 5.5): the component is conservative, and the questions our benchmark labels multi-hop are more often single retrieval operations over joined triples than genuinely independent information needs. We keep the stage in the architecture description because whether it helps depends on the query distribution and not on the design, but we report what it costs alongside why we included it.

The component prompts the LLM with the original user query and returns an array of sub-queries; if the query is atomic, the model returns a single-element list containing the original query. Sub-queries are independent by construction and go through the parsing and retrieval stages separately.

3.3 Constrained Semantic Parsing

For each sub-query, the `SparqlAgent` component does the actual neural-to-symbolic translation: it generates a SPARQL SELECT query that has to be valid against the vocabulary $V_{\mathcal{O}}$ of the ontology. This is semantic parsing over a *closed* vocabulary. Unlike open-domain code generation, every class and property IRI in the output has to come from a finite, enumerable set, and any deviation, however plausible it sounds, gives a query that executes but retrieves nothing meaningful.

The prompt given to the parser has two parts. The first is a *schema summary*, built from the ontology at startup by querying the RDF graph for all classes, object properties, and datatype properties together with their domains and ranges. It is a plain-text listing of local names and full IRIs, derived directly from the graph, and it gives the model an unambiguous mapping between human-readable names and ontology terms. The second part is the sub-query itself, together with a set of generation constraints aimed at the known failure modes of neural query generation:

- **Closed-world vocabulary constraint.** The model is told to treat the schema summary as the complete and only admissible vocabulary, and not to produce any class or property IRI that is not listed there. This targets the failure mode most often reported for neural parsing over closed vocabularies: inventing ontology terms that sound plausible but do not exist.
- **Single-token regex rule.** Any FILTER clause using REGEX must contain exactly one lexical token; for multi-word names the model must pick one distinctive term, preferring proper nouns or surnames. This avoids malformed regular expressions that would make execution fail.
- **Case-insensitive containment preference.** The model is told to prefer `CONTAINS(LCASE(?var), "token")` over REGEX where it can, since the former behaves more predictably.
- **No exact string equality.** The model must not match literals with string equality, since a user's phrasing rarely reproduces a stored value character for character. Section 5.5 shows that this constraint has a larger effect than we anticipated.
- **No fixed name-order assumption.** The model must not assume a canonical ordering of name components (for example "Surname, Name" against "Name, Surname"), so that it works across the naming conventions present in the data.
- **Abstract-based semantic matching.** For questions of the form "which books discuss topic X", the model is told to target the abstract datatype properties of the `Book` class rather than trying to match titles, using the textual layer of the ontology as a semantic index.

These constraints work at the prompt level. They *shape* the output distribution of the parser without formally *bounding* it, unlike grammar-constrained decoding (Willard and Louf 2023), which guarantees syntactic validity but cannot enforce that the terms used belong to $V_{\mathcal{O}}$. The gap between shaping and bounding is measurable: it is the fraction of generated queries that break the closed-world constraint despite the instructions, and we report it in Section 6.3.

The generated query runs against the local RDF graph through the SPARQL engine of the `OntologyExplorer` module, implemented with `rdflib`, which returns a list of result-row dictionaries or an error string if execution fails. The agent passes this result on unchanged and leaves aggregation and synthesis to the later stages.

3.4 Verification Gate and Sub-Symbolic Fallback

Running the generated query over the ontology does two things at once. When the result set is non-empty, it *is* the retrieval: every returned binding is a fact derived from the knowledge base and traceable to the triples that support it. When the result set is empty, execution acts as a *verification signal*. Without any heuristic judgment, it tells us that the neural parser failed to produce a query grounded in the data, whether because it used vocabulary outside the schema, because a filter was too narrow, or because the knowledge really is not there. The pipeline treats this signal as a gate $g(R_i)$ and, when it fails, hands control to a sub-symbolic retrieval path.

The reason for this two-tier strategy is a precision-recall trade-off that is built into the two mechanisms rather than incidental to them. Symbolic retrieval over a closed ontology is highly precise when the parse succeeds, but its recall is limited by how good the parser is and by what the schema can express. Dense retrieval has the opposite profile: embedding similarity holds up well when wording varies, but precision suffers, especially on exact factual look-ups. Giving the symbolic path priority and using the sub-symbolic path as fallback keeps the precision of the first wherever it is available, and limits how much recall is lost when the parse fails.

The fallback retriever is built on AllegroGraph's vector store, queried through a SPARQL-based LLM interface (the `llm:askMyDocuments` extension (DeBellis et al. 2024; Clarizia et al. 2025)). It retrieves the RDF triples whose associated text chunks are closest to the sub-query in embedding space, then expands each match to the local subgraph in which the matched instances appear as subject or object. It returns neighborhood structure rather than isolated documents, so the synthesis stage gets relational context on the sub-symbolic path too. A confidence threshold $\tau \in [0, 1]$ (the `level` option) sets the minimum similarity score for inclusion, and triples from RDF meta-vocabularies (`rdf:`, `owl:`, `rdfs:`) are filtered out to reduce noise.

This fallback is not the same thing as a conventional dense retriever over free text, and the difference matters for reading the path-stratified results in Section 5.4. Both retrieve by embedding similarity, but the fallback works over the ontology and returns triples with their local neighborhood, whereas a conventional dense baseline retrieves prose passages from a document index. What the sub-symbolic path relaxes is the requirement that the parse succeed, not the requirement that the evidence be structured. ORACOLO stays anchored to the ontology on both branches, and what the gate decides is whether the retrieved structure has been checked against an executable query.

3.5 Grounded Answer Synthesis

Once retrieval has finished for all sub-queries, the pipeline collects the evidence into a single prompt and calls the generation model. For each sub-query the prompt contains the sub-query string followed by its evidence, either a symbolic result set of verified bindings or a sub-symbolic retrieval of similar triples with their local subgraphs, and the original user query comes last. The model is asked to answer that question using the evidence it has been given.

Synthesis is deliberately the *only* point where a neural component puts domain content into words, and it does so only over retrieved evidence, so the division of labor set out in Section 3.1 holds end to end. The architecture does not guarantee that the generator stays faithful to the evidence it receives, and it cannot. What it does guarantee is that the evidence itself can be verified, and that whatever gap remains between evidence and answer can be measured directly: that gap is what the faithfulness metric of Section 5.3 reports.

4 Deployment Domain and Case Study

To test the architecture in a realistic setting, we deployed ORACOLO over the bibliographic knowledge base of the University Library of Lugano, a major library in the Italian-speaking part of Switzerland. The domain is representative of a broad class of institutional retrieval scenarios, and it puts pressure on exactly the properties the architecture is built for. The knowledge sits in a curated, richly structured catalog whose vocabulary is closed and specific to the institution. It is largely absent from the pretraining corpora

of general-purpose language models, so a purely parametric model has no reliable factual basis to answer from. And the users are non-experts, who cannot be expected to write formal queries. The domain therefore exercises both sides of the loop: the neural components have to absorb the linguistic variability of untrained users, while the symbolic component has to supply the factual grounding the neural components do not have.

4.1 Knowledge Base

The knowledge base, which we refer to as `bul1000`, is an RDF ontology serialized in RDF/XML and containing 96,348 triples. It covers a curated subset of the library catalog: 944 `ex:Book` instances, published between 1973 and 2024, in Italian and English, across philosophy, social sciences, natural sciences, economics, theology, and literature.

The ontology defines three main domain classes. `ex:Book` represents individual bibliographic records and carries datatype properties for title, author, publisher, publication date, language, and physical item identifier, as well as Italian and English abstract texts. `ex:Abstract` reifies the abstract as a first-class entity, storing the raw text and the abstract language. `ex:RelationAssertion` represents semantic relations extracted from abstract sentences, linking a predicate string to subject and object entities through VerbNet thematic roles. The named-entity layer is built around `srl:ENTITY` and its subclasses (person, place, organization, event, object, collection, information object, material dimension, rights, and time span), which classify the entities mentioned in the abstracts. Each entity is linked to its Wikidata identifier, URL, and description through the `srl:wikidataID`, `srl:wikidataURL`, and `srl:wikidataDescription` datatype properties. Textual mentions of entities inside abstracts are reified as `srl:MENTION` instances, connected to their canonical entity through `ex:denotes` and to the abstract they occur in through `ex:inAbstract`.

These layers do not cover the whole catalog, and the gap comes from the source data rather than from the schema. Of the 944 works, 558 (59.1%) have abstracts, always as an Italian–English pair, and 475 (50.3%) take part in at least one relation assertion, with 1,977 assertions and 8,445 mentions in total. The remaining works are represented by their bibliographic metadata alone.

The ontology was enriched in three directions. First, 1,197 `Abstract` instances were created as first-class RDF entities, available in both Italian (`ex:abstractIT`) and English (`ex:abstractEN`), which makes multilingual querying possible. Second, 8,445 `Mention` instances and 1,037 named entities (761 persons, 179 places, 97 organizations, and further subclasses) were extracted from the abstract text by named-entity recognition and linked to Wikidata identifiers, descriptions, and URLs. Third, 1,977 `RelationAssertion` instances were produced by running semantic role labeling over abstract sentences, capturing predicate-argument structures through VerbNet thematic roles (Agent, Theme, Location, Time, Source, Destination, Manner, Purpose, Cause, Beneficiary), each anchored to the sentence it came from.

This three-layer enrichment is what lets the symbolic path go beyond keyword lookup. Queries can target the catalog attributes (author, title, publisher, year), but also the semantic content of the abstracts and the relational structure extracted from them. The abstract layer in particular works as a semantic index over the catalog, and it is what the generation constraints of Section 3.3 tell the parser to use for topical questions.

The knowledge base from the parser's point of view.

Two properties of the ontology set the difficulty of the semantic parsing task and the scalability profile of the architecture. The first is the size of the vocabulary $V_{\mathcal{O}}$ the parser has to respect: `bul1000` declares 15 classes, 18 object properties, and 18 datatype properties, so $|V_{\mathcal{O}}| = 51$ admissible IRIs. The second is the size of the schema summary that goes into the parser prompt, about 660 tokens (2,644 characters). What limits the applicability of the approach is the schema summary, not the number of triples. The instance data stays in the graph and is never written into the prompt, whereas the schema grows with the number of distinct classes and properties and competes for context window with the query and the generation constraints. A knowledge base ten times larger in triples but with a comparable schema would

leave the parsing task essentially unchanged; one with a similar number of instances but several thousand properties would not. We return to this distinction, and to the selective schema injection it suggests, in Section 6.

4.2 The Verification Gate in Practice

We illustrate how the architecture behaves through three representative query patterns, chosen to show the three regimes of the verification gate: symbolic retrieval that succeeds, failure that is detected and recovered from sub-symbolically, and the failure mode that accounts for most of what is left.

4.2.1 Gate pass: semantic content search over the abstract layer

The query *"Which books in the catalog discuss the topic of artificial intelligence?"* cannot be answered from catalog metadata alone, since no datatype property records subject matter. Following the abstract-targeting constraint of Section 3.3, the parser targets the `ex:abstractEN` datatype property of `ex:Book` and generates:

```
PREFIX ex: <http://example.org/ex/>
SELECT DISTINCT ?title ?author ?year WHERE {
    ?b a ex:Book ;
       ex:title ?title ;
       ex:author ?author ;
       ex:publication_date ?year ;
       ex:abstractEN ?abs .
    FILTER(CONTAINS(LCASE(?abs),
                "artificial intelligence"))
}
ORDER BY ?year
```

The query respects the closed-world vocabulary constraint: every IRI in it belongs to $V_{\mathcal{O}}$, and no external or inferred property appears. It also follows the containment preference, using a case-insensitive `CONTAINS` instead of a REGEX filter. Execution returns a non-empty result set, so the gate passes ($g(R_i) = \mathsf{pass}$) and the sub-symbolic path is never used. The synthesis stage receives the matching titles and their metadata as verified symbolic evidence and writes an answer that cites the retrieved works. Every factual claim in that answer traces back to a triple in the graph.

4.2.2 Gate fail: detected parse failure and sub-symbolic recovery

The query *"Are there books related to the sociology of liquid modernity?"* shows the opposite regime. The parser produces a syntactically and semantically valid query that filters abstracts on the token *liquid*. That token does not appear verbatim in any abstract, so execution returns an empty result set. The gate fails ($g(R_i) = \mathsf{fail}$), and control passes to the sub-symbolic path.

The fallback retriever queries the vector store, which runs a dense similarity search over the embedded text chunks and returns the most relevant triples together with their neighborhood subgraph. Here it surfaces works by Zygmunt Bauman, whose concept of *liquid modernity* runs through much of his output, by semantic proximity in embedding space and with no exact lexical match. The example shows why the two paths belong together: the symbolic path fails where lexical form and semantic content come apart, which is where dense retrieval is at its best. Note also that the failure is *detected* rather than guessed at. A pure semantic-parsing system would return no answer here, and a pure dense system would have no verifiable path for the queries of the previous example.

4.2.3 Residual failure mode: schema-compliant queries over unpopulated paths

A third regime is neither a clean pass nor a clean recovery, and it accounts for most of the failures that remain. Consider the query *"Which central theme, as illustrated by the study of images in this research, is associated with the notions of 'otherness' and 'identity' that the book explores?"*. The parser produces the following:

```
SELECT DISTINCT ?book ?themeSurface WHERE {
  ?book a :Book .
  ?book :hasAssertion ?ra .
  ?ra :Theme ?mention .
  ?mention :surface ?themeSurface .
  FILTER ( CONTAINS(LCASE(?themeSurface), "alterità")
        || CONTAINS(LCASE(?themeSurface), "identità") )
}
```

Every class and every property in this query belongs to $V_{\mathcal{O}}$, so the closed-world constraint is

respected. The query is syntactically valid, executes without error, and returns nothing. Removing the filter and running it again also returns nothing, which puts the failure in the triple pattern rather than in the lexical match. The four-hop path from a book through a relation assertion, a thematic role, a mention, and its surface form is well formed under the schema, but too few instances in the graph complete it for the pattern to bind. The parser reasoned correctly about the vocabulary it was given, and incorrectly about the data, which that vocabulary does not describe.

This tells us more about the architecture than the vocabulary violation one might expect, and which our measurements do not find (Section 6.3). The schema summary lists classes and properties but says nothing about which combinations of them are populated, so from inside the prompt a schema-compliant path and a productive path look the same. Under the RDF semantics an unsatisfied pattern simply matches nothing, so the gate sees the same thing here as it saw for the lexical mismatch of the previous example, and it would see the same thing again for a vocabulary violation. The gate detects that the parse failed; as it stands, it does not diagnose *why*. This is where the difference lies between constraints that *shape* the output of a neural parser, as prompt-level instructions do, and mechanisms that would *bound* it. What is missing is not a stricter grammar (Willard and Louf 2023), since the grammar was respected, but a description of the data that the parser could consult. We give the relative frequency of the three regimes, and outline the repair loop this distinction makes possible, in Section 6.3.

4.3 System Deployment

ORACOLO runs as a locally hosted system on a dedicated server inside the institution's network. All neural inference happens locally through Ollama, with no calls to external API services. This gives the institution full control over its data, which was a hard requirement here, and it also dominates the latency profile of the deployment, as Section 5.6 shows.

5 Experimental Evaluation

The evaluation answers four questions. First, with the generator held fixed, does grounding retrieval in a formal ontology improve answer quality over dense retrieval, and in what respect (Section 5.4)? Second, which component of the architecture is responsible for the improvement (Sections 5.4 and 5.5)? Third, how do the symbolic and sub-symbolic paths divide the query space, and where does each of them fail (Sections 5.4 and 6)? Fourth, what does the architecture cost to run (Section 5.6)?

5.1 Benchmark Construction

We built a question-answering benchmark directly from the `bul1000` ontology. Each entry has a natural-language question, a gold-standard answer, and the set of RDF triples that support it, so every item can be traced back to the knowledge graph. Questions come in three complexity levels: *single-hop* (SH), answerable from a single triple; *multi-hop* (MH), requiring a chain across several triples; and *reasoning* (REA), requiring the triples to be combined with a broader understanding of the content. All questions and answers are in Italian, which is the language of the catalog and of the institution's users. A representative entry:

```
{
  "question_id": "SH_0001",
  "category": "single-hop",
  "question": "Da chi è stato scritto il libro
              'L'intagliatore di noccioli di pesca'?",
  "expected_answer": "Orengo, Nico"
}
```

Construction and augmentation.

We started from a set of questions written by hand against the ontology, each paired with its evidence triples; 40 multi-hop and 47 reasoning questions from that set are in the final benchmark. We then expanded it in three ways. *Template-based generation* applied fixed syntactic patterns to ontology triples, producing single-hop questions whose corresponding triple is known by construction. *Paraphrasing* produced reformulations that vary the wording while keeping the meaning and the evidence triples. *Generative augmentation* prompted an LLM with ontology content to write reasoning questions. Table 1 summarizes the resulting dataset.

Validation and its scope.

Augmented items went through a two-stage filter. An automated stage removed malformed,

Table 1 Benchmark composition by category and construction method. Manually authored questions carry no `source` field in the dataset; template, paraphrase, and LLM-generated counts are taken directly from it.

Category	Manual	Template	Paraphrase	LLM-generated	Final
SH	0	4,013	93	—	4,106
MH	40	—	161	799	1,000
REA	47	—	179	1,040	1,266
Total	87	4,013	433	1,839	6,372

duplicated, and degenerate entries, and discarded template-generated questions whose evidence triples did not resolve against the graph. A manual stage then reviewed the questions that make up the evaluation sample, correcting artifacts of template generation (unnatural phrasing, ambiguous referents, gold answers not uniquely determined by the evidence triples) and discarding items that could not be repaired. It is worth being clear about what this review does and does not establish. It establishes that the evaluated questions are *correct*: each is well formed, and each gold answer follows from its evidence triples. It does not make them *representative*, since a template-generated question that has been corrected by hand still comes from a template and not from a library patron.

5.2 Experimental Setup

All experiments ran on a local server (Ubuntu 24.04 LTS, NVIDIA A6000 GPU, 64 GB RAM). Neural inference was served through Ollama, using `gpt-oss:120b` as the semantic parser π (temperature 0, fixed seed, `num_predict` = 2048, context window 8192) and `llama3.3:latest` for query decomposition D and answer synthesis γ. Dense embeddings for both the subsymbolic path ϕ and the RAG baseline were computed with `nomic-embed-text`. The RAGAS judge is `llama3.3:latest` (70B, 4-bit quantized, temperature 0, context window 8192), with `qwen3-embedding:8b` for the answer-relevancy metric. No external API calls were made at any point. We evaluated on a stratified random sample of the benchmark drawn with a fixed seed. One multi-hop question was excluded after it exceeded a two-hour wall-clock budget, leaving $n = 97$ (single-hop 40, multi-hop 31, reasoning 26).

The dense index and the ontology are built from the same set of source records, the identifiers overlap completely in both directions, and where an Italian abstract is present in both it is character-identical. Coverage, however, is not symmetric, and we state the asymmetry rather than assume it away. The spreadsheet the dense index is built from keeps Italian abstracts for 403 records whose ontology counterpart has none. The ontology, on the other hand, carries an English translation of every abstract it does contain (597 in total, absent from the spreadsheet), along with the named-entity and semantic-role layer described in Section 4.1. So the comparison varies representation together with a residual difference in coverage, and the two differences pull in opposite directions: the baseline has more raw text, the ontology more derived structure. Since ORACOLO's advantage is measured despite the baseline's larger textual coverage, the estimate is conservative with respect to that confound.

Baselines.

The central baseline is **RAG-only (generator-matched)**: dense retrieval over the same corpus, using the *same* `llama3.3` generator, `nomic-embed-text` embeddings, and synthesis prompt as ORACOLO, so that what varies is the representation of the same source records rather than the capability of the generator. We chose this deliberately. Faithfulness to retrieved evidence is largely a property of the generator, and only a generator-matched comparison can attribute a difference to retrieval rather than to model capacity. A second baseline, **LLM-only**, answers from parametric knowledge with no retrieval.

5.3 Metrics

We compute two metrics with RAGAS (Es et al. 2024). **Faithfulness** is the fraction of the answer's

factual claims that the retrieved evidence supports; it is undefined, and so not reported, where nothing was retrieved. **Answer Relevancy** measures how well the answer aligns semantically with the question. For accuracy we report **Exact Match** and **Token F1** against the gold answer, and **ROUGE-L**.

For the ablation study we pre-register a single *primary metric*, computed per question and chosen to match its category: exact match for single-hop, Token F1 for multi-hop, and ROUGE-L for reasoning. Each category is therefore scored by the measure that fits the form of its expected answer, and the remaining metrics are reported as secondary. We assess significance with a paired bootstrap over questions (10,000 resamples) and a Holm–Bonferroni correction applied within metric families rather than across the whole set of tests: the primary metric forms one family, and each secondary metric forms another. We also report two gate-level statistics, the *gate pass rate* and the *fallback activation rate*.

A small number of questions produced retrieved contexts longer than the judge's context window and were truncated to fit it before evaluation. Two of these returned no faithfulness score and are left out of the corresponding means.

5.4 Main Results

Table 2 reports the comparison on the stratified sample ($n = 97$). The RAG-only baseline is *generator-matched*: it uses the same `llama3.3` model, the same synthesis prompt, the same embedding model, and the same questions as ORACOLO, and differs only in how it retrieves the evidence. The LLM-only baseline answers from parametric knowledge with no retrieval at all. All differences are computed pairwise over questions, with a bootstrap confidence interval (10,000 resamples).

ORACOLO improves Token F1 from 0.289 to 0.471, a paired difference of +0.183 (95% CI $[+0.094, +0.275]$), and ROUGE-L from 0.223 to 0.388 (+0.165, 95% CI $[+0.081, +0.251]$). On the pre-registered primary metric the difference is +0.104 (95% CI $[+0.012, +0.197]$). Exact match improves by +0.072, but its interval includes zero (95% CI $[-0.010, +0.155]$). The symbolic path retrieves the right entity considerably more often than it produces a string identical to the gold answer, and part of that gap is surface form: gold answers are written as "Surname, Name" where the system says "Name Surname".

Against the ungrounded LLM-only baseline the differences are larger and all significant: +0.321 Token F1 (95% CI $[+0.246, +0.402]$) and +0.165 exact match (95% CI $[+0.093, +0.237]$). LLM-only does not produce a single exact match across the 97 questions. That is what we would expect for a catalog whose records are not in any pretraining corpus, and it confirms that the benchmark cannot be answered from parametric knowledge.

The advantage is confined to factual retrieval.

Breaking the results down by category shows where the effect sits. On single-hop questions ORACOLO improves Token F1 over the matched baseline by +0.297 (95% CI $[+0.115, +0.476]$), and on multi-hop by +0.196 (95% CI $[+0.062, +0.341]$). On reasoning questions the two systems are indistinguishable (-0.008, 95% CI $[-0.053, +0.033]$), and the ungrounded baseline is not far behind either (0.199 against 0.251 and 0.242). This fits what symbolic execution does and does not do. It retrieves facts under a verified query, and where the expected answer is an inference over facts rather than the facts themselves, verified retrieval gives no advantage over dense retrieval of the same material. What this architecture improves is the reliability of what gets retrieved, not the reasoning applied to it.

Faithfulness does not separate the systems.

Faithfulness is 0.887 for ORACOLO against 0.848 for the matched baseline, a difference of +0.041 whose interval includes zero (95% CI $[-0.026, +0.109]$). We report this explicitly because an earlier, unmatched comparison suggested a much larger gap: against a baseline built on an 8B model with a different prompt and output language, the same metric separated the systems by more than 0.20. Repeating the comparison with the generator held fixed removes nearly all of that separation, which was therefore an effect of model capacity and not of grounding. This is as much a caution about how baselines are built as it is a result about ORACOLO. Faithfulness

measures whether the evidence put in front of the generator supports the answer, and a capable generator is faithful to whatever evidence it receives, including evidence that is off target. What separates the two systems is not how faithful the synthesis is but how accurate the retrieval is, and that is what the lexical metrics capture and faithfulness does not.

Retrieval path stratification.

ORACOLO resolves 76.3% of questions through the symbolic path, where the gate passes and the answer is synthesized from verified triples, and 23.7% through the sub-symbolic fallback. Table 3 compares ORACOLO against the matched baseline within each stratum. On the symbolic path the advantage is +0.215 Token F1 (95% CI $[+0.102, +0.332]$, $n = 74$); on the fallback path it is +0.079 (95% CI $[+0.009, +0.182]$, $n = 23$), a factor of 2.7 smaller.

These are descriptive strata, not a controlled contrast, and the difference matters. Neither path shares a retrieval mechanism with the baseline: as Section 3.4 describes, the fallback works over the ontology by embedding similarity with local subgraph expansion, whereas the baseline retrieves abstract passages from a vector store. Both branches of ORACOLO are therefore anchored to the ontology, and both beat free-text dense retrieval. What the stratification shows is that the branch under symbolic verification accounts for much the larger share of the advantage. To make the stronger claim, that verification itself is what does the work, we would need a configuration whose fallback retrieves from the same index as the baseline, so that the two systems coincide exactly when the gate fails. We flag this as the natural next experiment, without claiming what it would show.

The relevance paradox.

Answer relevancy ranks the three systems in a way that no accuracy metric supports. The LLM-only baseline scores highest of the three (0.499), above both the dense baseline (0.347) and ORACOLO (0.451), while getting no exact match on any of the 97 questions and a Token F1 of 0.150. Its advantage over ORACOLO is not significant (-0.048, 95% CI $[-0.108, +0.011]$), which means that on this metric a system that answers no question correctly is indistinguishable from one that answers a sixth of them correctly.

The ablation in Section 5.5 reproduces the same divergence under controlled conditions. Adding an intent-routing stage that skips retrieval on roughly a third of the queries *raises* answer relevancy by 0.112 (95% CI $[+0.059, +0.170]$) while significantly lowering the primary accuracy metric. Generator, prompt, embedding model, sample, and evaluation protocol are all held fixed, and only one stage is toggled, so the divergence cannot come from differences between systems.

Together these two observations tell us about the metric, not about the systems. Answer relevancy scores how well the answer aligns with the question, and an ungrounded model writes answers that are fluent, complete, and exactly on topic: they address the question that was asked, and they are wrong. Grounded answers are terse by comparison, and when retrieval returns nothing usable the system may decline to answer at all, which the metric punishes. A measure widely used to evaluate retrieval-augmented generation can therefore be improved by taking the retrieval away. We report it for comparability with the literature, but the results above suggest it should not be read as evidence of grounding quality unless an accuracy metric is reported next to it.

The cost of verification.

The accuracy comes at a substantial cost in latency: 291.6 s mean end-to-end against 34.6 s for the matched baseline and 20.6 s for unaided generation (Section 5.6). Removing the query-decomposition stage, which the ablation shows to be inert on this workload, brings this down to 187.3 s with no measurable loss in accuracy, but the order of magnitude does not change. Verified retrieval on local hardware is expensive, and whether the trade is worth making depends on the deployment rather than on the architecture. For a catalog people consult interactively it probably is not. For one consulted where an unsupported answer carries a real cost, it probably is.

5.5 Ablation Study

To see which components account for the observed accuracy, we evaluate five configurations on the same stratified sample used throughout ($n = 97$,

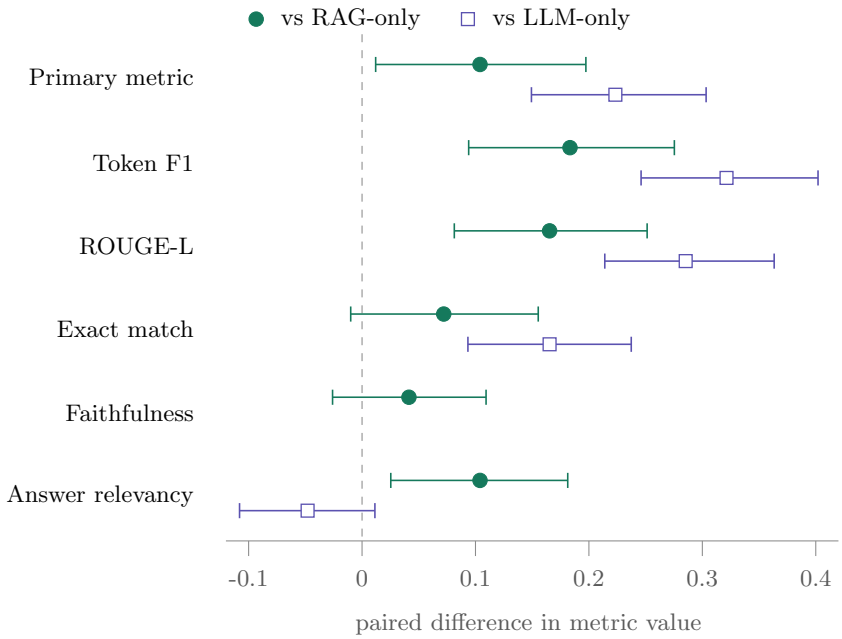


Fig. 2 Paired differences between ORACOLO and each baseline, with 95% bootstrap confidence intervals (10,000 resamples) over the 97 evaluation questions. An interval that crosses the dashed zero line is not significant: ORACOLO does not separate from the generator-matched baseline on exact match or on faithfulness, and does not separate from the ungrounded baseline on answer relevancy. Faithfulness is undefined for LLM-only, so that series has no entry for it. Note also that the RAG-only series sits entirely to the left of the LLM-only series, giving the same ordering across all four accuracy metrics.

Table 2 Main results on the stratified sample ($n = 97$). RAG-only is generator-matched: same model, prompt, embedding model, and questions, differing only in how it retrieves the evidence. LLM-only answers from parametric knowledge with no retrieval. Faithfulness is undefined when nothing was retrieved and is not reported for LLM-only. Figure 2 gives the paired differences and their confidence intervals.

Metric	LLM	RAG	ORACOLO
Primary metric	0.100	0.219	**0.323**
Token F1	0.150	0.289	**0.471**
ROUGE-L	0.102	0.223	**0.388**
Exact match	0.000	0.093	**0.165**
Faithfulness	—	0.848	0.887
Answer relevancy	**0.499**	0.347	0.451
Mean latency (s)	20.6	34.6	291.6

same seed and same questions, so that every comparison is paired). The reference configuration, **Full**, is ORACOLO as deployed. Three configurations remove a component: −**Decomp** passes the query to the parser undecomposed; −**Constraints** keeps only the closed-world vocabulary restriction in the parser prompt and drops the generation constraints of Section 3.3; and −**Fallback** returns no answer when the gate fails, instead of handing control to sub-symbolic retrieval. The fifth, +**Intent**, *adds* a component: an intent-classification stage that sends queries it judges answerable without retrieval straight to the generator. This stage was part of an earlier design and we removed it on the evidence reported below; we keep it here as an ablation because its net effect is itself a result.

We pre-register a primary metric, computed per question and matched to the category being evaluated: exact match for single-hop, Token F1 for multi-hop, and ROUGE-L for reasoning. We test significance against Full with a paired bootstrap over questions (10,000 resamples) and a Holm–Bonferroni correction applied within metric families, where the primary metric is one family and each secondary metric is another. As it turns out, −Decomp sends the parser the same input that Full does (see below), so the difference between those two rows is not an effect at all but a direct measurement of how much the pipeline varies from run to run. We report it as an

Table 3 Token F1 by retrieval path and by question category, ORACOLO against the matched RAG-only baseline, with paired 95% bootstrap confidence intervals. The two paths are descriptive strata: ORACOLO's fallback and the baseline do not share a retrieval index (Section 3.4).

Stratum	n	ORACOLO	RAG-only	Δ (95% CI)
By retrieval path				
Symbolic (gate pass)	74	0.506	0.291	$+0.215$ $[+0.102, +0.332]$
Fallback (sub-symbolic)	23	0.359	0.281	$+0.079$ $[+0.009, +0.182]$
By question category				
Single-hop	40	0.629	0.333	$+0.297$ $[+0.115, +0.476]$
Multi-hop	31	0.460	0.264	$+0.196$ $[+0.062, +0.341]$
Reasoning	26	0.242	0.251	-0.008 $[-0.053, +0.033]$

empirical noise floor: 0.0003 on the primary metric, 0.004 on Token F1, and 0.015 on faithfulness. Every effect we report as significant is at least an order of magnitude above that floor.

Decomposition is inert on this workload.

−Decomp is statistically indistinguishable from Full ($\Delta = 0.0003$, 95% CI $[-0.028, +0.030]$), and the interval is narrow enough that this counts as an equivalence result and not as an absence of evidence: any effect of decomposition on this sample is smaller than 0.03 on the primary metric. The traces show why. The decomposition stage emitted exactly one sub-query for every question in the sample, including all 31 multi-hop items, and in every case that sub-query was word for word the original question. The stage therefore did not restructure the parsing problem at all, while still costing one inference call per question: removing it brings mean latency down from 291.6 s to 187.3 s, a reduction of 35.8%, at no measurable cost in accuracy. This does not refute the argument of Section 3.2, which is about queries that express genuinely independent information needs. It shows that such queries are rarer in this benchmark than the multi-hop label suggests, and that the decomposer is conservative about identifying them. On workloads that look like this one, the stage costs something and returns nothing.

The fallback trades precision for coverage, and the trade is worth making.

Removing the dense fallback leaves 22.9% of questions unanswered, and the aggregate primary metric falls accordingly ($\Delta = -0.065$, 95% CI $[-0.111, -0.017]$). On the questions it does answer, though, −Fallback is *more* accurate than Full (Token F1 0.516 against 0.471) and more faithful (0.927 against 0.887). Neither figure is an improvement: both are selection effects. When the gate fails the system says nothing instead of retrieving a passage, so the answers that survive are exactly the ones backed by verified symbolic evidence. Taken together, the two columns put a number on the trade-off: the fallback buys 22.9 points of coverage for 4.5 points of Token F1 on the answers actually produced. This also tells us what a pure semantic-parsing architecture would give up on this catalog, which is a little under one query in four.

The parsing constraints cost more precision than they buy.

The largest effect in the study goes against what the design intended. Removing the generation constraints *improves* the primary metric by 0.152 (95% CI $[+0.076, +0.233]$), and at the same time lowers the gate pass rate from 76.3% to 57.7%. These two movements do not contradict each other. They are the same mechanism seen at opposite ends of the query distribution. Broken down by category, −Constraints raises exact match on single-hop questions from 0.325 to 0.700, while its gate pass rate on multi-hop and reasoning questions falls from 61.3% and 57.7% to 32.3% and 23.1%.

Looking at the generated SPARQL shows which constraint is responsible. The ban on exact string equality pushes the constrained parser towards single-token containment filters. On single-hop questions it emits `CONTAINS` in 38 of 40 cases, matching on one distinctive term, and the result sets it gets back have a median size of 2 and a mean of 5.6; only 18 of 40 queries return a

Table 4 Ablation results ($n = 97$; $n = 96$ for −Fallback, where one question exceeded the execution budget). *Primary* is the pre-registered per-category metric. An asterisk marks a difference from Full that is significant on the primary metric under Holm–Bonferroni correction within family ($\alpha = 0.05$). The counts in parentheses under faithfulness give the number of questions for which that metric is defined, since it is undefined when nothing was retrieved.

Configuration	n	Primary	Token F1	Faithf.	Relev.	Gate pass	Answered
Full	97	0.323	0.471	0.887 (96)	0.451	76.3%	100.0%
+Intent*	97	0.235	0.320	0.845 (67)	**0.563**	68.7%	100.0%
−Decomp	97	0.323	0.475	0.903 (96)	0.463	74.2%	100.0%
−Constraints*	97	**0.474**	**0.580**	0.899 (97)	0.480	57.7%	100.0%
−Fallback*	96	0.261	0.398	0.927 (74)	0.339	77.1%	77.1%

Table 5 Paired differences from Full on the pre-registered primary metric, with 95% bootstrap confidence intervals (10,000 resamples) and Holm–Bonferroni thresholds (HBT). Rows are ordered by p-value, which is the order in which the Holm procedure tests them.

Config.	Δ Primary	95% CI	HBT
−Constr.	+0.152	[+0.076, +0.233]	0.0125
+Intent	−0.088	[−0.157, −0.026]	0.0167
−Fallback	−0.065	[−0.111, −0.017]	0.0250
−Decomp	+0.000	[−0.028, +0.030]	0.0500

single row. The unconstrained parser instead puts the full literal straight into the triple pattern in all 40 cases, and every one of those queries returns exactly one row. The effect at synthesis time shows up in the answers: asked for the publication year of a specific title, the constrained system retrieves two candidate years and answers "2020 o 2019", while the unconstrained system retrieves one and answers correctly.

This result needs a caveat, and the caveat is as informative as the result itself. In all 40 single-hop cases, the literal the unconstrained parser matched on appears *verbatim* in the question, because the benchmark was built from the ontology (Section 5.1). Exact matching works here because the question already contains the retrieval key. Where it does not, the strategy fails outright: on multi-hop and reasoning questions the unconstrained parser tries the same construction and its queries return a median of zero rows, which is exactly the collapse in gate pass rate reported above. So the constraints do not trade precision for nothing. They trade it for robustness when the question and the data use different words, and on a benchmark where that difference is absent by construction, the trade looks like a bad one. The conclusion is not that the constraints should go, but that they should apply *conditionally*: exact matching where the query carries a verbatim entity mention, containment where it does not. A two-stage strategy that tries exact matching first and relaxes to containment on an empty result set would get both regimes without asking the parser to make that judgment, and it is the most direct improvement this study points to.

Intent routing degrades accuracy while improving relevancy.

Adding the intent-classification stage lowers the primary metric by 0.088 (95% CI $[-0.157, -0.026]$), and the loss is concentrated on single-hop questions, where exact match falls from 0.325 to 0.100. The stage sent 30 of the 97 questions to direct answering. These are catalog lookups whose answers cannot be recovered from parametric knowledge, and the shortfall measures what it costs to skip retrieval on them. On the same configuration, though, answer relevancy *rises* by 0.112 (95% CI $[+0.059, +0.170]$), the highest value of any configuration in the study. This is the controlled counterpart of the relevance paradox discussed in Section 5.4, and it is the clearest single reason for removing the stage from the deployed architecture.

Faithfulness does not discriminate between configurations.

No configuration differs significantly from Full on faithfulness, and all four differences fall below 0.024, inside the noise floor established above. This has to do with what the metric covers, not with how well it was measured. Faithfulness asks whether the retrieved evidence supports the synthesized answer. The ablations change *which* evidence gets retrieved, not how faithfully it is

used. The synthesis stage is the same in all five configurations and is equally faithful to whatever it is handed, which is why the metric is flat across the study, and why the accuracy differences it misses turn up on the lexical metrics instead. Two questions, both in the reasoning category and both with retrieved contexts far longer than the judge's context window, returned no faithfulness score and are left out of the corresponding means.

5.6 Latency Analysis

The mean end-to-end latency of 291.6 s (median 261.7 s) comes from three sequential neural inference calls per query on shared local hardware. Table 6 breaks it down by stage, and two things in that breakdown say something about the design rather than about the implementation.

The first is the symbolic component. Running the generated SPARQL against the ontology takes a median of 0.067 s, three orders of magnitude below any neural stage: the cost of the architecture is in neural inference, and the ontology is almost free. The mean of 3.8 s does not describe this behavior but its tail, a small number of queries whose filters are loose enough to force a scan over the graph, with a maximum of 249.8 s. This matters for deployment: symbolic execution is negligible in the typical case, and in the atypical one it needs a timeout rather than an optimization.

The second is where the neural cost actually falls, which is not where the design expected it. Query decomposition is the most expensive stage at 46.5% of end-to-end latency, ahead of semantic parsing (22.4%) and answer synthesis (22.5%). On this benchmark that stage returns the input query unchanged every time (Section 5.5), and its cost does not come from the difficulty of the decision: it spends a median of 135.3 s producing a one-element list, three times what synthesis spends producing a complete answer. Removing it is therefore not a trade-off but a straight improvement on this workload. The −Decomp configuration brings mean latency down to 187.3 s, a reduction of 35.8%, with no statistically detectable cost in accuracy (Δ primary = 0.000, 95% CI $[-0.028, +0.030]$) or faithfulness.

For reference, the ungrounded LLM-only baseline takes 20.6 s per query on the same hardware and the generator-matched dense baseline 34.6 s. The overhead of the neuro-symbolic machinery is therefore 271.0 s in the deployed configuration, a factor of 14.2 over unaided generation. Removing the query-decomposition stage brings this down to 166.7 s, a factor of 9.1. The architecture buys verified retrieval at roughly an order of magnitude in latency, and the ablation of Section 5.5 shows that a third of that cost currently goes to a stage that contributes nothing on this workload.

6 Discussion and Limitations

6.1 What Symbolic Grounding Contributes

The generator-matched comparison separates the contribution of the retrieval mechanism from that of model capacity, and the first thing it shows is where the contribution is *not*. Faithfulness to retrieved evidence does not separate the two systems (0.887 against 0.848, 95% CI $[-0.026, +0.109]$). This is not a null result to be explained away. It tells us what the metric measures. Faithfulness asks whether the answer follows from the evidence put in front of the generator, and a capable generator is faithful to whatever it gets, including evidence that is off target. Both systems speak accurately about what they retrieved; what they retrieved is where they differ. An earlier, unmatched comparison in which the baseline ran on an 8B model suggested a gap above 0.20 on this metric, and almost all of that gap disappears once the generator is matched. We report this because it is a caution about how baselines are built, which bears directly on how systems of this kind get evaluated.

Where symbolic grounding does contribute is factual precision, and the category breakdown draws a sharp line around the claim. The advantage is +0.297 Token F1 on single-hop and +0.196 on multi-hop questions, and it disappears on reasoning questions (-0.008, 95% CI $[-0.053, +0.033]$), where the ungrounded baseline is itself not far behind. This is the theoretical boundary between symbolic and sub-symbolic competence showing up in the measurements: a verified query retrieves a stored fact, and where the expected answer is an inference over facts rather than a fact, there is nothing for verification to verify. The architecture improves the reliability of what gets retrieved, not the reasoning applied

Table 6 Per-stage latency breakdown for the deployed configuration ($n = 97$). Medians are reported alongside means because several stages are heavy-tailed; the fallback row is conditioned on the 23 queries for which the gate failed and the sub-symbolic path was activated.

Stage	Component	Median (s)	Mean (s)	Max (s)	% of total
Query decomposition D	`llama3.3`	135.3	135.5	321.4	46.5
Semantic parsing π	`gpt-oss:120b`	60.5	65.3	136.2	22.4
Symbolic execution ε	`rdflib`	0.067	3.8	249.8	1.3
Fallback retrieval ϕ	vector store	93.4	89.6	128.8	7.3
Answer synthesis γ	`llama3.3`	44.5	65.6	730.2	22.5
End-to-end		261.7	291.6	—	100.0

to it, and any claim made for it should keep that distinction.

Grouping the questions by the path actually taken narrows the effect down further, though on its own it does not establish causation. On the questions resolved by the symbolic path the advantage is +0.215 Token F1; on those that fall back to sub-symbolic retrieval it is +0.079, smaller by a factor of 2.7. Both branches of ORACOLO are anchored to the ontology, since the fallback retrieves triples with their local neighborhood rather than prose passages, so some advantage on the fallback path is to be expected and the comparison is a stratification, not a controlled contrast. Showing that verification itself is what does the work would take a configuration whose fallback draws from the same index as the baseline, so that the two systems coincide exactly when the gate fails. That is the decisive follow-up experiment.

6.2 The Limits of Relevance-Based Evaluation

Answer relevancy is one of the two RAGAS metrics we report, and one of the most widely used in evaluations of retrieval-augmented generation. It ranks our three systems in a way that no accuracy metric supports. The ungrounded LLM-only baseline scores highest of the three (0.499), above the dense baseline (0.347) and above ORACOLO (0.451), while getting an exact match on none of the 97 questions. Its difference from ORACOLO is not significant (-0.048, 95% CI $[-0.108, +0.011]$): on this metric, a system that answers no question correctly cannot be told apart from one that answers a sixth of them correctly.

A skeptical reader might put this down to differences between the systems being compared. The ablation rules that out. Adding an intent-routing stage to ORACOLO, which sends roughly a third of the queries to the generator without retrieval, raises answer relevancy by 0.112 (95% CI $[+0.059, +0.170]$) while significantly lowering the pre-registered accuracy metric by 0.088. Generator, prompt, embedding model, question sample, and evaluation protocol are identical across the two configurations. One stage is toggled, and the two metrics move significantly in opposite directions. What we are seeing is a property of the metric, not of the comparison.

The mechanism is clear enough once it is spelled out. Relevancy scores how well the answer aligns with the question, and an ungrounded model is very good at producing text that is fluent, complete, well formed, and exactly on topic. Such answers address the question that was asked. They are simply wrong. Grounded answers are terse by comparison, and when retrieval returns nothing usable a well-behaved system declines to answer, which the metric punishes. The −Fallback configuration is a case in point: it is silent on 22.9% of questions, and it records the lowest relevancy in the ablation even though it is the most faithful configuration on the answers it does produce.

What this means in practice is that answer relevancy cannot stand as evidence of grounding quality. It measures fluency and topicality, and on tasks where the failure we care about is confident fabrication it is not just uninformative but misleading: it rewards exactly the behavior that grounding is meant to suppress. We report it for comparability with the literature, and we recommend reading it only alongside a metric that can detect a wrong answer.

6.3 Failure Modes of Constrained Neural Parsing

The verification gate fails on 23.7% of questions (23 of 97). We re-executed every failed query against the ontology, with and without its lexical filters, which sorts the failures into three regimes that call for different remedies.

A minority, 3 of 23 (13%), are queries that do not execute at all: undeclared namespace prefixes and malformed query forms, which the SPARQL engine reports as errors. The remaining 20 execute without complaint and return an empty result set, and diagnosing these takes the ontology and not just the query. When the lexical filters are removed and the queries run again, 13 of the 20 (57% of all gate failures) come back non-empty, in several cases with thousands of rows: the triple patterns are sound, and what matched nothing was the single-token containment filter. The other 7 (30%) return nothing even unfiltered, so there the pattern itself is unsatisfied, not the filter.

No failure in the sample is a vocabulary violation. Of the 33 distinct prefixed names that appear in failed queries, 32 belong to $V_{\mathcal{O}}$ as the schema summary presents it; the one exception, `rdfs:label`, occurs once. Whatever else the closed-world constraint costs, it is respected. The failure mode the architecture was designed to catch, a parser inventing plausible vocabulary, is not the one we see.

What we do see splits cleanly along the structure of the ontology. The 13 filter failures stay on the core bibliographic layer: `Book` and `title` appear in all 13, alongside `author`, `publisher`, and `publication_date`. The 7 pattern failures reach into the semantic-role enrichment layer: `hasAssertion` appears in 5 of the 7, `denotes` and the thematic role `Theme` in 4, `surface` in 3, and `inAbstract` in 2, and none of these appears among the filter failures. One representative case chains `hasAssertion`, a thematic role, `MENTION`, and `surface` into a four-hop path where every class and property exists but the path as a whole binds nothing.

The instance layer explains why these paths bind nothing, and it does so with numbers. Of the 944 bibliographic works in the knowledge base, 558 (59.1%) have an abstract at all, and only 475 (50.3%) take part in at least one relation assertion. The distribution over those that do is steep: 87 works carry a single assertion and 88 carry two. Any query that goes through `hasAssertion` is therefore addressing half the catalog at best, before its filters even apply, and a four-hop path through a thematic role to a mention surface narrows the candidate set further at every step. The parser is not making an unreasonable inference. The path it builds is the one the schema describes, and it would bind on a densely instantiated graph. What defeats it is that the enrichment layer, built as it is from abstracts that half the records do not have, is sparse in a way the schema summary says nothing about.

This gives the parser's error a more specific name than vocabulary hallucination, and a more actionable one: the parser fails where the schema summary misleads it. The summary lists `hasAssertion` and `title` as equally available, when one binds for every record and the other for half of them, so a parser reasoning over vocabulary rather than over data builds paths that are well formed, schema-compliant, and empty. The remedy is not a tighter constraint on generation but a richer schema description, one that says which joins are instantiated and how densely. A summary of that kind would let the parser prefer the dense path, or drop the sparse one, while it generates rather than after execution, and it is cheap to produce: the coverage figures are computed once from the graph. It also addresses the failure mode that neither a stricter grammar nor a syntactic validator can reach. This is worth noting for ontology-grounded architectures in general, since the enrichment layer that makes an ontology more than a table of the same records is exactly the layer whose queryability the schema summary overstates.

The three regimes call for different remedies, and the distinction matters for anyone building a system like this. A generate–validate–repair loop takes care of execution errors, since the engine already reports what is wrong. It does not help with the other two: the gate detects that a parse failed but not *why*, and an empty result set looks the same whether the filter was too narrow or the join was unpopulated. Telling the two apart is mechanical once the ontology is available, since re-executing the query without its filters does it in one extra query, and passing that diagnosis back to the parser would turn one-shot constrained generation into targeted self-correction: relax the

filter in the first case, drop the path in the second. This is the most direct extension of the present architecture, and the measurements above suggest it would reach the 87% of gate failures that a syntactic validator cannot see.

The ablation adds something about the constraints themselves that we did not expect. Removing them *improves* the primary metric by 0.152, and the gain is concentrated on single-hop questions, where exact match rises from 0.325 to 0.700. The constraint responsible is the ban on exact string equality, which pushes the parser towards single-token containment filters: on single-hop questions the constrained parser emits `CONTAINS` in 38 of 40 cases and its result sets have a median size of 2, while the unconstrained parser puts the full literal in the triple pattern and returns a single row in all 40. The same removal, though, collapses the gate pass rate on multi-hop and reasoning questions, from 61.3% and 57.7% to 32.3% and 23.1%, because exact matching has nothing to match when the question does not quote the data.

So the constraints trade precision for robustness when the question and the data use different words. On our benchmark that difference is largely absent by construction, since the questions were generated from the ontology, and the trade therefore looks like a bad one. On queries written independently of the data it would not. The conclusion is not that the constraints should go, but that they should be conditional: exact matching where the query carries a verbatim entity mention, containment where it does not, or better still a two-stage strategy that tries exact matching first and relaxes to containment on an empty result. One mechanism explains both this finding and the dominant failure mode above: over-constrained lexical filters are at once the reason the gate fails and the reason successful queries return imprecise answers.

6.4 Scalability and Generalizability

What limits applicability is not the number of triples, since instance data never enters the prompt, but the size of the schema injected into the parser's context. The `bull1000` schema has 51 IRIs and takes about 660 tokens, roughly 0.5% of a 128k-token context window. A knowledge base ten times larger in triples but with a comparable schema would leave the parsing task essentially unchanged. A graph with several thousand properties would not: it would need selective schema injection, retrieving the relevant fragment of the schema for each query, which is a sub-symbolic solution to a symbolic scaling problem.

A second limit is computational rather than representational. The deployed configuration answers a query in 291.6 s against 20.6 s for unaided generation, a factor of 14.2, and the ablation shows that a third of that cost goes to a decomposition stage that returns the query unchanged on every question in our sample. What is left is dominated by sequential inference across heterogeneous models on a single accelerator, where the models cannot be co-resident and memory contention becomes part of the latency. Architectures of this kind are cheap in their symbolic component, whose median execution time is 0.067 s, and expensive in orchestrating their neural ones, which is the opposite of the intuition that symbolic reasoning is the costly part.

Two limits bound the generality of the evaluation. The first is that it covers a single institution and a single ontology, in a domain, bibliographic catalogs, whose schema is unusually regular; ontologies with deeper class hierarchies, or heavier use of inference, would stress the parser differently.

The second concerns the knowledge base rather than the method. The enrichment layer that makes the ontology more than a table of the same records covers only half of them, and the questions our benchmark asks of that layer can only be answered for that half. A more completely enriched graph would plausibly raise the gate pass rate above the 76.3% we observe, so that figure should be read as a property of this deployment and not as a ceiling on the architecture.

Finally, the evaluation measures what automated metrics can capture: whether an answer matches a reference, and whether it follows from the evidence retrieved. Section 6.2 shows that even within that scope the choice of metric can invert conclusions. Usefulness and trust, which are what determine whether an institution adopts a system over its curated catalog, are not in that set, and a human-centered study with the library's own users remains the necessary complement to this work.

7 Conclusion and Future Work

This paper presented ORACOLO, a neuro-symbolic question-answering architecture in which a neural semantic parser, restricted to the vocabulary of a formal ontology, works together with a symbolic execution engine that acts as a verification gate, with a dense sub-symbolic path as fallback. Against a generator-matched dense-retrieval baseline, faithfulness to retrieved evidence does not distinguish the two systems (0.887 against 0.848), while factual precision does: Token F1 rises from 0.289 to 0.471. Symbolic grounding is therefore not a way to make generation *more faithful*, since a capable generator is already faithful to its evidence. It is a way to make the retrieved evidence itself exact and checkable. The advantage is 2.7 times larger on the questions the symbolic path resolves than on those handed to the fallback, and it disappears on questions that call for inference rather than retrieval, which keeps the claim to what the architecture was built to do.

We also note a practical property of the design: the architecture runs entirely on locally served models, with no dependence on external inference providers. This is not incidental to the setting. Formal ontologies are curated most carefully by institutions that hold data they cannot freely transmit, and an approach to querying them that presupposes a hosted proprietary model is unavailable exactly where the knowledge is richest.

Three directions follow. The first is to close the verification loop: the gate detects that a parse has failed but not why, and passing that diagnosis back to the parser would turn one-shot constrained generation into iterative self-correction, with entity resolution against the ontology's Wikidata layer grounding the named entities. The second is to scale the symbolic side, through selective schema injection and federation across ontologies. The third is a human-centered evaluation with library patrons.

References

Ali W, Saleem M, Yao B, et al (2022) A survey of rdf stores & sparql engines for querying knowledge graphs. The VLDB Journal 31(3):1–26. https://doi.org/10.1007/s00778-021-00711-3

van Bekkum M, de Boer M, van Harmelen F, et al (2021) Modular design patterns for hybrid learning and reasoning systems. Applied Intelligence 51:6528–6546. https://doi.org/10.1007/s10489-021-02394-3

Bhuyan BP, Ramdane-Cherif A, Tomar R, et al (2024) Neuro-symbolic artificial intelligence: a survey. Neural Computing and Applications 36:12809–12844. https://doi.org/10.1007/s00521-024-09960-z

Cai R, Wang C, Cai Q, et al (2025) Boosting knowledge graph-based recommendations through confidence-aware augmentation with large language models. URL https://arxiv.org/abs/2502.03715, arXiv:2502.03715

Clarizia F, De Santo M, Gaeta R, et al (2025) Enhancement large language models domain through ontology-based retrieval-augmented generation. International Journal on Semantic Web and Information Systems (IJSWIS) 21(1):1–29. https://doi.org/10.4018/IJSWIS.392507

Colelough BC, Regli W (2025) Neuro-symbolic AI in 2024: A systematic review. ArXiv:2501.05435

DeBellis M, Dutta N, Gino J, et al (2024) Integrating ontologies and large language models to implement retrieval augmented generation. Applied Ontology 19(4):389–407. https://doi.org/10.1177/15705838241296446

DeLong LN, Fernández Mir R, Fleuriot JD (2024) Neurosymbolic AI for reasoning over knowledge graphs: A survey. IEEE Transactions on Neural Networks and Learning Systems https://doi.org/10.1109/TNNLS.2024.3420218

Edge D, Trinh H, Cheng N, et al (2024) From local to global: A graph RAG approach to query-focused summarization. ArXiv:2404.16130

Emonet V, Bolleman J, Duvaud S, et al (2025) Llm-based sparql query generation from natural language over federated knowledge graphs. URL https://arxiv.org/abs/2410.06062, arXiv:2410.06062

Es S, James J, Espinosa Anke L, et al (2024) RAGAs: Automated evaluation of retrieval augmented generation. In: Aletras N, De Clercq O (eds) Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations. Association for Computational Linguistics, St. Julians, Malta, pp 150–158, https://doi.org/10.18653/v1/2024.eacl-demo.16, URL https://aclanthology.org/2024.eacl-demo.16/

Gao Y, Xiong Y, Gao X, et al (2024) Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:231210997 URL https://arxiv.org/abs/2312.10997

d'Avila Garcez A, Lamb LC (2023) Neurosymbolic AI: The 3rd wave. Artificial Intelligence Review 56:12387–12406. https://doi.org/10.1007/s10462-023-10448-w

Guo Z, Xia L, Yu Y, et al (2025) LightRAG: Simple and fast retrieval-augmented generation. In: Christodoulopoulos C, Chakraborty T, Rose C, et al (eds) Findings of the Association for Computational Linguistics: EMNLP 2025. Association for Computational Linguistics, Suzhou, China, pp 10746–10761, https://doi.org/10.18653/v1/2025.findings-emnlp.568, URL https://aclanthology.org/2025.findings-emnlp.568/

Hitzler P, Eberhart A, Ebrahimi M, et al (2022) Neuro-symbolic approaches in artificial intelligence. National Science Review 9(6). https://doi.org/10.1093/nsr/nwac035

Ji Z, Lee N, Frieske R, et al (2023) Survey of hallucination in natural language generation. ACM Computing Surveys 55(12):1–38. https://doi.org/10.1145/3571730, URL https://dl.acm.org/doi/10.1145/3571730

Jiang L, Huang J, Möller C, et al (2025) Ontology-guided, hybrid prompt learning for generalization in knowledge graph question answering. URL https://arxiv.org/abs/2502.03992, arXiv:2502.03992

Kautz HA (2022) The third AI summer: AAAI Robert S. Engelmore memorial lecture. AI Magazine 43(1):93–104. https://doi.org/10.1002/aaai.12036

Lan Y, He G, Jiang J, et al (2023) Complex knowledge base question answering: A survey. IEEE Transactions on Knowledge and Data Engineering 35(11):11196–11215. https://doi.org/10.1109/TKDE.2022.3223858

Lewis P, Perez E, Piktus A, et al (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, pp 9459–9474, URL https://arxiv.org/abs/2005.11401

Liu H, Wang S, Chen C, et al (2025) Question-aware knowledge graph prompting for enhancing large language models. https://doi.org/10.18653/v1/2025.findings-acl.72, URL https://aclanthology.org/2025.findings-acl.72/

Loffredo R, De Santo M (2024) Using ontologies for llm applications in cultural heritage. In: Proceedings of the 3rd Workshop on Artificial Intelligence for Cultural Heritage (IAI4CH 2024) co-located with the 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024)

Luo H, E H, Tang Z, et al (2024) ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In: Ku LW, Martins A, Srikumar V (eds) Findings of the Association for Computational Linguistics: ACL 2024. Association for Computational Linguistics, Bangkok, Thailand, pp 2039–2056, https://doi.org/10.18653/v1/2024.findings-acl.122, URL https://aclanthology.org/2024.findings-acl.122/

Mountantonakis M, Tzitzikas Y (2025) Generating sparql queries over cidoc-crm using a two-stage ontology path patterns method in llm prompts. J Comput Cult Herit 18(1). https://doi.org/10.1145/3708326, URL https://doi.org/10.1145/3708326

Pan S, Luo L, Wang Y, et al (2024) Unifying large language models and knowledge graphs: A roadmap. IEEE Transactions on Knowledge and Data Engineering 36(7):3580–3599. https://doi.org/10.1109/TKDE.2024.3352100

Peng B, Zhu Y, Liu Y, et al (2025) Graph retrieval-augmented generation: A survey. ACM

Trans Inf Syst 44(2). https://doi.org/10.1145/3777378, URL https://doi.org/10.1145/3777378

Sharma K, Kumar P, Li Y (2025) OG-RAG: Ontology-grounded retrieval-augmented generation for large language models. In: Christodoulopoulos C, Chakraborty T, Rose C, et al (eds) Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Suzhou, China, pp 32962–32981, https://doi.org/10.18653/v1/2025.emnlp-main.1674, URL https://aclanthology.org/2025.emnlp-main.1674/

Sheth A, Roy K, Gaur M (2023) Neurosymbolic AI – why, what, and how. IEEE Intelligent Systems 38(3):56–62. https://doi.org/10.1109/MIS.2023.3268724

Wang W, Yang Y, Wu F (2025) Towards data-and knowledge-driven ai: A survey on neuro-symbolic computing. IEEE Transactions on Pattern Analysis and Machine Intelligence 47(2):878–899. https://doi.org/10.1109/TPAMI.2024.3483273

Willard BT, Louf R (2023) Efficient guided generation for large language models. arXiv preprint arXiv:230709702 URL https://arxiv.org/abs/2307.09702

Yu D, Yang B, Liu D, et al (2023) A survey on neural-symbolic learning systems. Neural Networks 166:105–126. https://doi.org/10.1016/j.neunet.2023.06.028

Zhao WX, Zhou K, Li J, et al (2023) A survey of large language models. arXiv preprint arXiv:230318223 URL https://arxiv.org/abs/2303.18223

Statements and Declarations

Funding.

No funding was received to assist with the preparation of this manuscript.

Competing Interests.

The authors have no relevant financial or non-financial interests to disclose.

Author Contributions.

Rocco Loffredo designed the system architecture, implemented it, designed and ran the experimental evaluation, and wrote the first draft of the manuscript. Benedetta Sabini constructed the knowledge base and the question-answering benchmark. Fabio Clarizia and Massimo De Santo supervised the work and contributed to the conception and design of the study. All authors read, commented on, and approved the final manuscript.

Declaration of Use of Generative AI.

The authors used a generative AI assistant during the preparation of this manuscript, for language editing, for drafting and revising portions of the text under the authors' direction, and for assistance with the analysis and evaluation scripts. The research questions, the system architecture and its implementation, the experimental design, the dataset, and the interpretation of the results are the authors' own. All content produced with such assistance was reviewed and validated by the authors, who take full responsibility for this article.

Data Availability.

The bibliographic knowledge base analyzed in this study was provided by the University Library of Lugano and is not publicly redistributable; it is available from the institution on reasonable request.

Code Availability.

The source code of ORACOLO is available from the corresponding author on reasonable request.

Compliance with Ethical Standards

Disclosure of Potential Conflicts of Interest.

The authors have no relevant financial or non-financial interests to disclose, and no conflicts

of interest that could have influenced the work reported in this manuscript.

Research Involving Human Participants and/or Animals.

This research does not involve human participants or animals. No experiments on humans or animals were performed by any of the authors. The study was conducted on bibliographic catalog metadata provided by the University Library of Lugano; no personal or sensitive data were collected or processed at any stage. Ethics committee approval was therefore not required.

Informed Consent.

Not applicable: the study did not involve human participants, so informed consent was not required.

Consent for Publication.

Not applicable: the manuscript contains no data or images relating to individual persons.