

Where Structure Helps: RDF vs. Property Graphs for Legal GraphRAG Retrieval

Vishnu Sankarakumar¹, Anelia Kurteva^{1,*}

¹*School of Computer Science, University of Birmingham, UK*

Abstract

Large Language Models (LLMs) are increasingly explored as advisory tools for navigating law. However, their tendency to hallucinate and not fully understand complex contexts makes them unsafe, as incorrect or incomplete legal guidance can carry severe consequences. Recent efforts have shifted toward integrating knowledge graphs (KGs) and LLMs via the Graph Retrieval-Augmented Generation (GraphRAG) approach. By providing a structured data infrastructure, KGs offer a mechanism to constrain LLM outputs and ensure they remain grounded in verifiable information. In such an approach, data can be represented as a Labelled Property Graph (LPG) or as an RDF KG. We found no prior work directly comparing the two for retrieval quality in GraphRAG for legal question answering. We present such a comparison over the EU AI Act. We propose a custom pipeline for reverse-engineering an automatically generated LPG and extracting an RDF ontology without loss of semantic data. GraphRAG systems are then implemented with both the LPG and the RDF KG backends. Both systems are evaluated on 50 AI Act-related questions across five complexity tiers using RAGAS, scoring faithfulness, context precision, and context recall. The result is a complexity-dependent crossover. The LPG-based GraphRAG is more faithful at every tier and stronger overall on simple factual questions. In comparison, the RDF-based system achieves higher context recall on the three hardest tiers, yielding 0.76 against 0.59 on the analytical tier. Structure is not uniformly better: the LPG's schema-less retrieval wins on simple questions, and the ontology's breadth only pays off once questions require traversal. In most legal compliance checks and guidance-seeking, the hard questions are the ones that matter. On those questions, the RDF KG recovers more of the needed information while maintaining inspectability.

Keywords

GraphRAG, Labelled Property Graph, RDF Knowledge Graph, Inspectable Retrieval, Auditability

1. Introduction

Large Language Models are increasingly being explored as advisory tools for navigating complex regulation, but their core limitation of hallucination makes them unsuitable for safety-critical domains without additional safeguards. A professional needing concrete advice about their AI system cannot tolerate a confidently wrong answer because there can be legal and financial consequences to bad guidance. The technology is not trustworthy enough to deploy without concrete domain knowledge.

Retrieval-Augmented Generation (RAG) addresses this by taking relevant context from a separate knowledge base and passing it to the LLM with the user's query, to ensure the response is grounded in source material rather than the model's pre-training. In standard RAG, the knowledge base is a collection of text chunks indexed by vector embeddings. GraphRAG replaces the flat data store with a structured knowledge graph made up of entities and relationships, enabling the system to traverse connections between concepts rather than relying on semantic similarity. This is important for a domain like the EU AI Act [1] where relevant information spans a dense and highly cross-referenced document containing hundreds of recitals and articles. In practice, a compliance auditor, lawyer, or student queries a natural language interface about the EU AI Act. They ask a question, the system retrieves relevant context from the knowledge graph, and an LLM generates an answer grounded in specific domain knowledge.

1st Semantic Architectures for Governance and Explainability (SAGE), co-located with SEMANTiCS 2026: 22nd International Conference on Semantic Systems, September 15, 2026, Ghent, Belgium

*Corresponding author.

✉ vishnusankarakumar@gmail.com (V. Sankarakumar); a.kurteva@bham.ac.uk (A. Kurteva)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Knowledge graphs can be implemented in two main ways: the first is a Labelled Property Graph (LPG) storing entities as nodes connected by directed and typed relationships that can store properties on edges. Neo4j is a widely used LPG database, queried with the Cypher language. The second is the Resource Description Framework (RDF) [2], which stores knowledge as subject-predicate-object triples with globally unique URIs, and is supplemented with the Web Ontology Language (OWL) [3] to declare formal class hierarchies, relationship rules and constraints. The main query language for RDF graphs is SPARQL [4]. Both implementations have different strengths.

Motivated by this, our paper aims to answer the following research question: how does the choice of knowledge graph representation affect both retrieval quality and the inspectability of regulatory question answering as questions grow more complex? To answer this, we make three contributions. First, an empirical one: across 50 questions in five complexity tiers, our LPG and RDF retrieval pipelines cross over, with the property graph system stronger on simple lookups and the RDF system stronger on the complex, multi-provision questions that matter for compliance. Second, an explanatory one: we show why this happens, discussing the inspectable paths of ontology structure and the leaner retrieval of the schema-less graph. Third, an architectural hypothesis: we propose a complexity-dependent system that selectively applies context post-processing. We make the direct comparison forming the foundation of this paper possible by developing a transformation pipeline that converts an LPG into a formally structured RDF ontology without loss of extracted entities or relationships.

2. Background

2.1. LPG vs. RDF for GraphRAG

An LPG stores entities as nodes joined by typed, directed edges that can carry properties, queried with Cypher. It does not require a formal schema, and here labels are free strings. RDF stores knowledge as subject-predicate-object triples with global URIs, and OWL adds a formal layer of class and property hierarchies on top, queried with SPARQL. The distinction that matters for retrieval is that with an RDF-OWL system the relationships between types, and between predicates, are themselves declared and queryable, whereas in a schema-less LPG they are not. Section 4.1.1 shows how this shapes retrieval.

2.2. Retrieval Approaches in Literature

The retrieval component of a RAG system determines how context is selected from the knowledge graph before being passed to the LLM. Four existing retrieval architectures informed project design.

PharmaBot [5] combines semantic and structural (Node2Vec [6]) embeddings, which motivated using vector similarity in both pipelines, rather than relying on query translation alone. PolyG [7] matches query type to question complexity, handling any question at the cost of latency. Iterative retrieval [8] re-ranks context over several rounds to cut noise: the most capable multi-hop approach, and the most expensive. A graph-enhanced e-commerce system [9] runs vector and Cypher retrieval in parallel. While these papers are concerned with highly relevant ideas, they do not directly compare an LPG against an RDF backend on the same data for retrieval quality. The LPG-versus-RDF comparisons we found measure storage and query performance, not the quality of retrieved context. This paper addresses that gap.

Question-type classification and iterative re-ranking were scoped out of both pipelines during development, but the results nonetheless support the principles behind them.

2.3. Semantic Representations of the AI Act

A parallel line of work encodes regulation itself as machine-readable knowledge. AIRO [10] models AI risks against the AI Act and ISO standards, the Data Privacy Vocabulary [11] has been extended with an AI Act module, and further efforts represent AI systems, their risks, and their permitted uses as machine-readable documentation and policies [12, 13, 14, 15]. Using these resources would further

the interests of actually deploying a RAG system on regulation. We aim to instead measure how the chosen form of knowledge graph affects retrieval and inspectability in RAG. The bespoke ontology built here is therefore an experimental control, rather than a proposed standard. It is derived from the LPG’s own vocabulary, and so holds identical instance data which an external vocabulary could not without re-extraction.

3. Methodology

To answer the main research question, we build two GraphRAG systems over the same section of the EU AI Act: one backed by a Neo4j LPG, and one backed by an RDF ontology and graph derived from the LPG with the same underlying data. Both are evaluated on the same set of questions. The methodology comprises four stages, each designed to keep the comparison as fair as possible.

Stage 1: LPG Construction The EU AI Act [1] (first 44 pages, recitals 1-179) is ingested and transformed into a Neo4j LPG using Neo4j’s native LLM Graph Builder [16], which automates chunking, entity extraction, relationship extraction, and native vector embedding creation. We use an existing tool to represent what an end user would arrive at using standard LPG tools. The first 44 pages are chosen due to the full document’s length of 400+ pages; the LLM extraction and query cost would be prohibitive.

Stage 2: LPG Analysis We analyse the created LPG using Python scripts, characterising label and predicate frequencies and their formatting variants. Without understanding the specific patterns and noise in the unconstrained LPG, accurate mapping is impossible.

Stage 3: Ontology Design and Mapping A target OWL ontology is designed through domain analysis of the EU AI Act, in tandem with the previous stage’s analysis. Explicit mappings are created to translate the noisy LPG data to this clean ontology, collapsing synonyms and unnecessary granularity into broader classes while preserving the underlying data entirely. These mappings are executed to create the RDF knowledge graph. The RDF RAG system uses Ontotext GraphDB [17].

Stage 4: Evaluation We use RAGAS [18] rather than ROUGE [19]/BLEU [20]/METEOR [21]. Syntax-overlap metrics fail on legal text (“must” and “must not” both score highly on word overlap). RAGAS uses LLM-as-judge to separately score three metrics: **faithfulness** (is the answer grounded in retrieved context rather than the LLM’s own knowledge?), **context precision** (what proportion of the retrieved content was actually relevant to the question?), and **context recall** (of the information needed to fully answer the question, how much did the retrieval actually find?). These metrics measure retrieval quality (precision and recall) and generation quality (faithfulness) *separately*, which is essential when the two systems share the same LLM and differ in retrieval. Because the LLM is a constant (GPT-5.2 [22]), any difference in output reflects the retrieval and representation pipeline rather than model quirks.

3.1. Neo4j LPG

The unconstrained extraction produced 3,816 entities under 298 labels and 6,552 relationships across 1,861 predicate types: a dense but messy output. The same concept appears under several labels (AI_System, AISystem, AI System), and relationships include synonyms (REGULATES, GOVERNS, LEGISLATES). This inconsistency is expected of schema-less LLM extraction, directly motivating the need for ontology work.

The LPG RAG pipeline uses the native vector index for initial semantic search, then Cypher graph and full-text traversal for context retrieval. An LLM translates natural language into Cypher, retrieves relevant entities, relationships, and chunk text, and passes the context to another agent for answer generation based on curated specific context.

3.2. RDF Ontology and Graph Construction

The RDF ontology was hand-designed through domain analysis, following established ontology engineering principles [23, 24].

Mapping decisions were then made through a human-in-the-loop workflow: an agent proposed semantic groupings of entities and predicates, these notions were manually fleshed out into detailed hierarchies, and a second agent validated the output for internal consistency across all 3,816 entities and 6,552 relationships. The goal of ‘mapping’ is to determine how each LPG entity/relationship should map to the RDF ontology: whether to assign it its own class, merge it into an existing one, or place it as a subclass under a parent. The general rule was to merge where the distinction has no consequence in the Act’s obligations, and to keep separate or as a subclass otherwise.

These decisions required genuine judgement, and in some cases, further domain research. For example, *Right* and *Interest* were kept as separate hierarchy branches due to their distinct legal definitions.

The results are class and property hierarchies with explicit label and predicate mappings that a Python script (`rdflib` [25]) applies to the LPG data dump to emit the ontology (TBox) and instances (ABox) as 27,340 triples. Entity, chunk, and relationship counts (3,816, 615, 6,552) all match the LPG source exactly. The only removed edges are the 2,672 LPG-internal items (*SIMILARITY*, *NEXT_CHUNK*, *FIRST_CHUNK*) which encode index structure rather than semantic content, and the vector embeddings are rebuilt natively in RDF. The two graphs therefore hold the same instance data, so evaluation variation cannot be attributed to differences in underlying extracted data.

The repository uses the RDFS-Plus (Optimized) ruleset with `owl:sameAs` inference enabled. Retrieval combines full-text search (capped at 25 triples per call) with similarity search over the native vector index, plus label-based full-text search and autocomplete for IRI discovery.

4. Experimental Results

Our first contribution is empirical. Both systems were evaluated on the same 50 questions split into five tiers of ten questions each, ordered by how many distinct pieces of information and graph hops the ground truth answer requires. The tiers are as follows: Simple Factual, Definitional, Multi-hop, Analytical, and Critical. A Simple Factual answer requires one fact retrieved from one provision; a Critical one requires four or more across three or more, as in Section 4.1.2. Each question has a manually written reference answer grounded in specific recitals of the EU AI Act [1]. Evaluation uses the RAGAS framework [18].

Table 1

Aggregate RAGAS scores across all 50 questions.

Metric	Neo4j (LPG)	GraphDB (RDF)	Difference
Faithfulness	0.836	0.595	+0.241 (Neo4j)
Context Precision	0.852	0.818	+0.034 (Neo4j)
Context Recall	0.599	0.524	+0.075 (Neo4j)

The LPG leads on all three aggregate metrics, most clearly on faithfulness (see Table 1). These overall numbers are misleading. The per-tier breakdown shows the two systems crossing over as questions get harder (see Figure 1). The LPG is more faithful than RDF at every tier, scoring a perfect 1.0 on the simple questions and degrading steadily with complexity. This degradation is a retrieval problem, not a generation one: when the LPG retrieves the right context its answer is almost always faithful to it, so what fails on hard questions is finding the context, not using it.

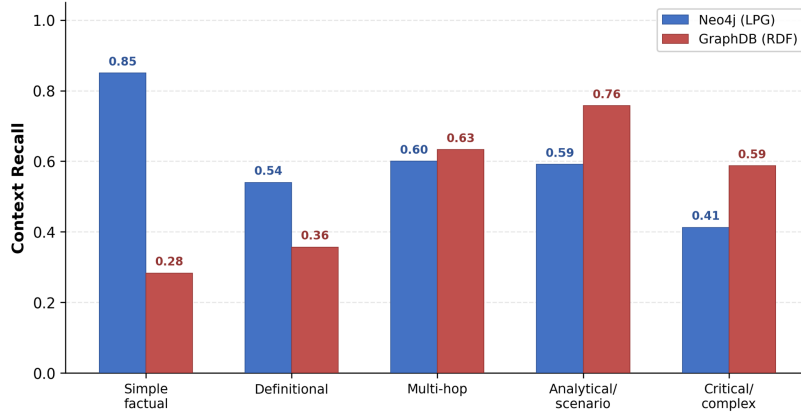


Figure 1: Context recall per difficulty tier. LPG wins on simple questions and the RDF KG wins on harder questions, with a complexity threshold in the middle.

Recall tells the opposite story. On the three hardest tiers the RDF system leads (multi-hop 0.63 vs. 0.60, analytical 0.76 vs. 0.59, critical 0.59 vs. 0.41), while the LPG leads on the two easiest (simple 0.85 vs. 0.28, definitional 0.54 vs. 0.36). These hardest tiers are exactly the questions that need traversal across several parts of the graph, and there the ontology backend finds more of the information the answer requires. The underlying mechanics of this are covered in Section 4.1.1.

The LPG is also erratic where it is weakest (critical-tier recall SD 0.507) against RDF’s steadier hard-tier performance (analytical SD 0.179), so RDF’s advantage is in both level and consistency. Precision follows the same crossover: the LPG is more precise on simple questions (0.82 vs. 0.53), RDF on harder ones (0.96 vs. 0.86 on multi-hop). RDF’s low precision on simple questions is the cost of its breadth. The same broad retrieval that helps on hard questions pulls in triples a simple question does not need. Structure is not free, and below the complexity threshold it is a net loss. On complex questions that breadth instead returns a higher proportion of relevant content.

4.1. Discussion

4.1.1. Why Structure Helps on Complex Queries

The practical advantage of ontology structure is that it makes retrieval inspectable; every retrieved item arrives as a typed IRI carrying its declared parents, so its structural position in the hierarchy is visible in the returned context itself. For example: in the RDF predicate hierarchy, `regulatoryRelation` is the parent of `regulates`, `prohibits`, `restricts`, and `allows`.

Because the repository uses RDFS-Plus reasoning, a `DESCRIBE` over `prohibits` returns its parent `regulatoryRelation`, alongside asserted relations and types. This means it is one step away from `regulates`, `restricts`, and `allows`. The LLM receiving this context can see not just what the entity/relation is, but where it sits in the ontology and what it connects to, which makes the retrieval path auditable.

In the LPG system, relationships are just labels, and there is no traversable ontology. `REGULATES` and `PROHIBITS` are unrelated strings unless the query explicitly lists both. If the LLM generating the query writes `[ :REGULATES ]`, only `[ :REGULATES ]` is returned, whether or not other semantically associated predicates exist.

The same applies to entity types: a SPARQL query over `Actor` implicitly covers `Organisation`, `GovernmentBody`, `Person`, `Stakeholder`, and `MarketActor` as related children. Schema-less LPG cannot do this. The class/predicate hierarchy widens retrieval coverage at each step, catching relevant items that a schema-less system misses.

4.1.2. Worked Example: Question 42

Question 42 asks: *“Define what constitutes a ‘serious incident’, connect it to post-market monitoring obligations, and trace the governance chain: who monitors, who receives reports, and how does this differ for AI systems built on general-purpose AI models?”*

This is a critical-tier question. The ground truth lists four pieces of information spanning recitals 153, 155, and 161.

```
DESCRIBE ?iri {  
  ?x onto:fts ('serious incident definition EU AI Act' '*')  
  { ?x ?p ?iri . } UNION { ?iri ?p ?x . } }
```

Listing 1: First SPARQL query for Question 42.

The RDF pipeline used full-text search as its entry point: `onto:fts` lands on the matched `serious_incident` entity, then a `UNION` pulls in everything connected to it in both directions, and `DESCRIBE` expands on those. This returned its definition, the causal chain (use of a GPAI model causes a serious incident), the responsible actor, and its place in the class hierarchy (`Incident subclassOf Event`).

Two further queries retrieved the remaining context needed. RDF scored 1.0 on recall and 1.0 on faithfulness. The LPG scored 0.0 on both. Its vector search retrieved content adjacent to “monitoring” and “governance” but structurally unrelated to serious incidents, because semantic similarity is not the same as a structural connection.

4.2. What is Being Compared: Representation, Schema, or Retrieval?

The two systems differ in three ways at once: representation and engine (Neo4j against GraphDB/RDF-OWL), schema (LLM-generated labels against a hand-engineered ontology), and retrieval (vector and Cypher against full-text, vector and `DESCRIBE`). However, retrieval strategy is not a wholly free variable here. RDF queries return the relevant entity’s hierarchy as well because the schema declares it. Cypher has no equivalent, so an LPG can only imitate that by encoding the hierarchy as data and traversing it in the query, at which point the structure is bolted on rather than intrinsic to the graph. Equalisation is only available in the other direction: disabling RDFS-Plus reasoning degrades one system to match the other, and tests neither as it would be deployed. The choice of entry point is free and the ontology’s quality is not controlled, so the claim still has boundaries: the pipelines evaluated here show RDF recovering more of the context that complex questions need, without separating how much of that is owed to the quality of the ontology rather than to the representation. Where this paper refers to ‘LPG’ or ‘RDF’, it means these two systems.

4.3. LPG Failure Modes

The LPG’s failures on hard questions are repeatable, and stem from its dependence on initial vector localisation. Vector localisation works when the query maps to one close chunk. However, on Q42 the words “monitoring” and “governance” are semantically close to many irrelevant chunks, so vector search starts from the wrong entity and Cypher/full-text search then traverse correctly from the wrong place, with no way to recover. Separately, the Act’s long recitals are split across chunks, so the chunk holding the query terms and the chunk holding the answer are often different: vector search finds the former and stops. RDF is less exposed to both problems because vector similarity is one of two entry points. Full-text search is the alternative, landing on the matched entity and pulling in everything structurally connected to it, regardless of which chunk the text came from. The ontology’s structure offsets the LPG system’s over-reliance on semantic similarity, which leaves the LPG behaving like the flat vector RAG a graph is meant to improve on.

4.4. Retrieval/Utilisation Gap

In 10 of the 30 questions where RDF retrieved adequate context ($\text{recall} \geq 0.5$), the answer still scored below 0.5 on faithfulness. The right information was retrieved but did not survive into the answer. The cause is volume. RDF’s DESCRIBE returns full subgraphs averaging 5,544 characters against the LPG’s 2,097, padded with namespace declarations, sibling classes, and full URIs, so the answer is present but buried in noise the LLM does not extract. This suggests a presentation issue, fixable without major architectural changes.

4.5. Limitations

We report per-tier means over 50 questions without confidence intervals or significance testing. The questions, reference answers, and tier assignments were authored internally without independent expert validation, and RAGAS’s LLM-as-judge risks sycophancy bias, so results should be read as indicative. Two constraints are accepted as the cost of a controlled comparison. The RDF graph is derived from the LPG, so extraction errors carry over. The ontology and transformation pipeline are reusable in principle, but labour-intensive to apply in a new domain. They are released as an experimental artefact for reproducibility¹, not as a proposed AI Act vocabulary. Quantifying inspectability, running ablations isolating representation from ontology quality and retrieval strategy, and adding a human-judged evaluation subset are the natural next steps. Extending beyond the first 44 pages to the full AI Act, the GDPR [26] and the NIST AI RMF [27] should put to use RDF’s other structural advantage of global URIs.

5. Conclusion

Empirically, the two systems have complementary retrieval strengths that cross over between simple and complex questions. The ontology’s hierarchies widen retrieval and recover more context on hard questions, but this is not a uniform good. The breadth afforded by ontology structure is dead weight on simple ones, degrading precision. Meanwhile, despite being a graph, the LPG inherits vector RAG’s weakness of reliance on vector similarity to find its starting point, hurting recall on hard questions.

These findings motivate an untested logical solution: a single RDF system with complexity-dependent post-processing, stripping DESCRIBE noise where the question does not need it and keeping full context where it does. The premise is that RDF’s faithfulness weakness is one of presentation rather than retrieval. In 10 of 30 questions where RDF’s recall was adequate but faithfulness was not, the needed content was already retrieved albeit lost in context noise. This also matches design intuitions in existing work: PolyG’s [7] assertion that no single retrieval strategy serves both simple and complex questions, and the use of re-ranking to filter broad retrieval [8]. In a compliance setting the hard questions are the ones that matter, and on those the RDF system both retrieves more of what is needed and makes the retrieval inspectable. When mistakes carry legal consequences, structure earns its cost.

Declaration on Generative AI

No GenAI tools were used in the preparation of this manuscript.

References

- [1] European Parliament and Council of the European Union, Regulation (EU) 2024/1689 (Artificial Intelligence Act), *Official Journal of the European Union*, L 2024/1689, 2024.
- [2] R. Cyganiak, D. Wood, M. Lanthaler, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, 2014. <https://www.w3.org/TR/rdf11-concepts/>
- [3] W3C OWL Working Group, OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation, 2012. <https://www.w3.org/TR/owl2-overview/>

¹<https://github.com/VishnuSankarakumar/SAGE-2026-LPG-vs.-RDF>

- [4] S. Harris, A. Seaborne, SPARQL 1.1 Query Language, W3C Recommendation, 2013. <https://www.w3.org/TR/sparql11-query/>
- [5] B. Menaouer, C. Younes, E. E. Mokhtar, et al., Pharmaceutical Chatbot Assistant Using Generative AI and Knowledge Graph with Specific Pharmaceutical Resources, *Iran Journal of Computer Science* 9 (2026), article 40. <https://doi.org/10.1007/s42044-026-00396-3>
- [6] A. Grover, J. Leskovec, node2vec: Scalable Feature Learning for Networks, in: *Proc. ACM SIGKDD (KDD)*, 2016, pp. 855–864.
- [7] R. Liu, H. Jiang, X. Yan, B. Tang, J. Li, PolyG: Adaptive Graph Traversal for Diverse GraphRAG Questions, arXiv:2504.02112, 2025.
- [8] K. Guo, X. Dai, S. Zeng, et al., Beyond Static Retrieval: Opportunities and Pitfalls of Iterative Retrieval in GraphRAG, arXiv:2509.25530, 2025.
- [9] P. Patel, Graph-Enhanced Retrieval-Augmented Question Answering for E-Commerce Customer Support, arXiv:2509.14267, 2025.
- [10] D. Golpayegani, H. J. Pandit, D. Lewis, AIRO: An Ontology for Representing AI Risks Based on the Proposed EU AI Act and ISO Risk Management Standards, in: *Towards a Knowledge-Aware AI (SEMANTiCS 2022)*, vol. 55, IOS Press, 2022, pp. 51–65.
- [11] H. J. Pandit, B. Esteves, G. P. Krog, et al., Data Privacy Vocabulary (DPV) Version 2, EU AI Act Extension (EU-AIAct), W3C DPVCG, 2024. <https://w3c.github.io/dpv/2.2/legal/eu/aiact/>
- [12] D. Golpayegani, I. Hupont, C. Panigutti, et al., AI Cards: Towards an Applied Framework for Machine-Readable AI and Risk Documentation Inspired by the EU AI Act, in: *Privacy Technologies and Policy (APF 2024)*, Springer, 2024, pp. 48–72.
- [13] D. Golpayegani, H. J. Pandit, D. Lewis, AICat: An AI Cataloguing Approach to Support the EU AI Act, arXiv:2501.04014, 2025.
- [14] D. Golpayegani, B. Esteves, H. J. Pandit, D. Lewis, AIUP: An ODRL Profile for Expressing AI Use Policies to Support the EU AI Act, *CEUR Workshop Proceedings*, vol. 3759, paper 17, 2024.
- [15] D. Golpayegani, H. J. Pandit, D. Lewis, Semantic Patterns of Prohibited AI Systems in the EU AI Act, in: *SEMANTiCS 2025 (Posters, Demos, Workshops & Tutorials)*, *CEUR Workshop Proceedings*, vol. 4064, 2025.
- [16] Neo4j, Inc., Neo4j LLM Graph Builder Documentation, 2024. <https://neo4j.com/labs/genai-ecosystem/llm-graph-builder/>
- [17] Ontotext, GraphDB Free Edition, 2024. <https://www.ontotext.com/products/graphdb/>
- [18] S. Es, J. James, L. Espinosa-Anke, S. Schockaert, RAGAS: Automated Evaluation of Retrieval Augmented Generation, in: *Proc. 18th Conf. EACL: System Demonstrations*, 2024, pp. 150–158.
- [19] C.-Y. Lin, ROUGE: A Package for Automatic Evaluation of Summaries, in: *Text Summarization Branches Out*, 2004, pp. 74–81.
- [20] K. Papineni, S. Roukos, T. Ward, et al., BLEU: A Method for Automatic Evaluation of Machine Translation, in: *Proc. 40th Annual Meeting of the ACL*, 2002, pp. 311–318.
- [21] S. Banerjee, A. Lavie, METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments, in: *Proc. ACL Workshop on Evaluation Measures for MT and Summarization*, 2005, pp. 65–72.
- [22] OpenAI, Introducing GPT-5.2, 2025. <https://openai.com/index/introducing-gpt-5-2/>
- [23] T. R. Gruber, A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition* 5(2) (1993) 199–220.
- [24] N. F. Noy, D. L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, Stanford KSL Technical Report KSL-01-05, 2001.
- [25] RDFLib Team, RDFLib: A Python library for working with RDF, 2024. <https://github.com/RDFLib/rdfliib>
- [26] European Parliament and Council of the European Union, Regulation (EU) 2016/679 (General Data Protection Regulation), *Official Journal of the European Union*, L 119, 2016, pp. 1–88.
- [27] National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1, 2023. <https://doi.org/10.6028/NIST.AI.100-1>