A Benchmark for Gap and Overlap Analysis as a Test of KG Task Readiness

Maruf Ahmed Mridul[1], Rohit Kapa[2] and Oshani Seneviratne[1]

[1]*Rensselaer Polytechnic Institute, Troy, New York, United States*
[2]*Prudential Financial, Inc.*

Abstract

Task-oriented evaluation of knowledge graph (KG) quality increasingly asks whether an ontology-based representation can answer the competency questions that users actually care about, in a manner that is reproducible, explainable, and traceable to evidence. This paper adopts that perspective and focuses on gap and overlap analysis for policy-like documents (e.g., insurance contracts), where given a scenario, which documents support it (overlap) and which do not (gap), with defensible justifications. The resulting gap/overlap determinations are typically driven by genuine differences in coverage and restrictions rather than missing data, making the task a direct test of KG task readiness rather than a test of missing facts or query expressiveness. We present an executable and auditable benchmark that aligns natural-language contract text with a formal ontology and evidence-linked ground truth, enabling systematic comparison of methods. The benchmark includes: (i) ten simplified yet diverse life-insurance contracts reviewed by a domain expert, (ii) a domain ontology (TBox) with an instantiated knowledge base (ABox) populated from contract facts, and (iii) 58 structured scenarios paired with SPARQL queries with contract-level outcomes and clause-level excerpts that justify each label. Using this resource, we compare a text-only LLM baseline that infers outcomes directly from contract text against an ontology-driven pipeline that answers the same scenarios over the instantiated KG, demonstrating that explicit modeling improves consistency and diagnosis for gap/overlap analyses. Although demonstrated for gap and overlap analysis, the benchmark is intended as a reusable template for evaluating KG quality and supporting downstream work such as ontology learning, KG population, and evidence-grounded question answering. The benchmark and accompanying codebase are available at https://github.com/brains-group/gap_and_overlap_analysis_insurance_contract_benchmark.

Keywords

life insurance dataset, knowledge graph quality, task-oriented evaluation, ontology engineering, OWL (TBox/ABox), gap and overlap analysis, competency questions, evidence-grounded evaluation, SPARQL query answering, scenario-based testing

1. Introduction

Assessing and improving the quality of Knowledge Graphs (KGs) is an active research area, motivated by the growing use of KGs as reusable and interoperable resources for downstream analysis and AI systems [1, 2]. In addition to structural and logical criteria, many evaluation efforts now take a *task-oriented* view of quality: a representation is valuable if it supports the competency questions [3, 4] that users care about, with answers that are reproducible, explainable, and traceable to evidence. This paper adopts that perspective and studies a concrete task family centered on *gap and overlap analysis*.

In many domains, important rules and exceptions are written in *policy-like documents*, such as contracts, regulations, service terms, or product specifications. These documents state what is supported under certain conditions, what is excluded or denied, and what is outside scope. Stakeholders frequently need to compare multiple documents: for a given scenario, which documents support it (overlap), and which do not (gap). For example, a scenario in which an insured dies by suicide 13 months after policy issuance would be covered by contracts with a 12-month exclusion period but not by those with a 24-month exclusion. The overlapping contracts are those that agree on covering it, while the gap is the subset that do not. Here, gaps and overlaps are usually *substantive*: they occur because documents differ in their intended coverage and restrictions, not because a KG is missing facts or because a query

QKG @ ESWC'26
mridum@rpi.edu (M. A. Mridul); rohit.kapa@prudential.com (R. Kapa); senevo@rpi.edu (O. Seneviratne)
0009-0003-7501-4714 (M. A. Mridul); 0000-0001-8518-917X (O. Seneviratne)

cannot be written. From a KG perspective, gap and overlap analysis is therefore a useful test of task readiness: can an ontology-based representation show these differences consistently, and can it provide evidence that supports each outcome?

Reproducible evaluation of such analyses is nevertheless difficult without aligned resources that connect natural language documents to formal representations and queryable ground truth. In practice, it is common for such document collections to be complex, hard to share, or insufficiently annotated for detailed benchmarking, which limits the ability to evaluate both explainability and correctness of the results produced by different methods. It also limits the evaluation of automatic ontology generation and refinement methods, since their outputs cannot be tested against shared, executable competency questions with evidence. To address this barrier, we contribute a benchmark dataset that supports such evaluation purposes, and we demonstrate its use through a workflow that makes gap and overlap analysis executable and auditable. In this paper, we focus on the life insurance domain, where contracts are legally structured yet heterogeneous in their benefits, exclusions, conditions, and temporal constraints, making them a suitable setting for auditable gap and overlap analysis.

Our contributions are four-fold.

- First, we release a manually curated corpus of ten simplified but diverse *life insurance contracts*, verified by a domain expert (who is also a co-author on this paper), designed to cover a range of clause patterns while remaining manageable for systematic study.
- Second, we provide a domain ontology (TBox) that formalizes the main contract concepts and relations, together with an aligned ABox populated from the contract facts, enabling deterministic, query-based analysis.
- Third, we release a suite of 58 structured scenarios and a corresponding ground truth for gap and overlap analysis, which is also reviewed and verified by a domain expert. For each scenario, we include contract-level outcomes and clause-level source excerpts that justify those outcomes, supporting traceability and diagnosis.
- Fourth, we evaluate gap and overlap determination using two approaches: a text-only LLM pipeline that infers contract responses for each scenario, and an ontology-driven pipeline that answers the same scenarios using SPARQL over the instantiated ontology. This comparison measures the value of explicit ontological modeling for producing consistent and auditable gap and overlap analyses.

Although the benchmark is built in the life insurance domain, the main goal is more general. It shows how aligned document text, an ontology with instances, scenario probes, and evidence-linked labels can form a reusable resource for KG quality evaluation and for downstream tasks such as ontology learning, KG population, and explainable question answering. More broadly, this benchmark aims to address key challenges in modern AI: the limits of LLM reasoning, the need for explainable and auditable outputs, and the integration of structured-logic based and neural methods. By providing a dataset where every answer is traceable to structured evidence, we enable faithful evaluation of reasoning systems beyond surface-level accuracy. Our results demonstrate that while LLMs can approximate coverage judgments, they are less reliable when a scenario requires strict handling of structural applicability, contractual scope, or clause-level justification. By contrast, the ontology-backed pipeline makes these commitments explicit and executable. In this sense, the benchmark is not only a resource for KG quality evaluation, but also a controlled testbed for studying how structured, logic-based representations can complement neural methods in high-stakes domains where correctness and justification matter.

2. Related Work

KG quality evaluation has received attention since Zaveri et al. [5] systematized 18 quality dimensions for linked data, and Xue and Zou [1] extended this framework across the full KG lifecycle. Paulheim [6]

surveys refinement approaches for correcting ABox-level errors through statistical and embedding-based methods. More recently, Tsaneva et al. [2] combine LLM suggestions with human-in-the-loop review to detect structural issues in populated KGs. Complementing these structural and statistical approaches, recent work has explored explainability-driven methods for improving rule-based reasoning systems. Seneviratne et al. [7] introduce a framework that leverages trace-based, contextual, contrastive, and counterfactual explanations to debug and refine rules, enabling human-in-the-loop validation of reasoning outcomes. This line of work highlights the importance of interpretability not only for model transparency but also for systematic quality improvement in knowledge-driven systems.

A consistent observation across this work is that structural metrics do not assess whether a KG supports the analytical tasks it was built for. The task-oriented view of quality: a representation is valuable if it correctly answers the competency questions that motivated its construction [3, 4], follows directly from the methodology introduced by Grüninger and Fox [8], who argued that an ontology's correctness should be characterized by the queries it can answer. Our benchmark operationalizes this view: the 58 scenarios function as executable competency questions whose outcomes serve as a task-oriented quality test for both the TBox and ABox. Unlike structural validators such as SHACL [9], this benchmark tests whether a KG can produce consistent and evidence-grounded coverage determinations across heterogeneous contracts. Performance directly reflects the ability to support reproducible gap and overlap determinations, with strong results implying task-scoped consistency within the defined scenario suite, though broader completeness beyond that is not guaranteed. Therefore our benchmark can be positioned as a complementary, task-oriented layer in any KG quality workflow.

The closest work to our gap and overlap task comes from the legal NLP community. Koreeda and Manning [10] introduce ContractNLI, a document-level natural language inference dataset of annotated contracts, where a system must determine whether a hypothesis is entailed by, contradicts, or is not mentioned in a contract, with supporting evidence spans. Hendrycks et al. [11] introduce CUAD, a large expert-annotated dataset for identifying 41 types of salient clauses across 500+ contracts. Both works address the problem of extracting and verifying claims against contract text, which motivates our use case. However, neither provides cross-contract comparison with formal semantics: they treat each contract independently and rely on neural models rather than a shared formal representation. Kang et al. [12] similarly find that LLM-generated structured outputs from health insurance policies require human oversight, further motivating formal representations. This limitation has motivated prior efforts to encode contractual logic in structured, machine-executable forms. Seneviratne et al. [13] advocate for computable contracts in insurance marketplaces, emphasizing transparency, automation, and auditable decision-making. Building on this vision, Van Woensel et al. [14] demonstrate how clinical decision logic represented in KGs can be translated into executable smart contracts, enabling consistent enforcement across distributed systems. Our work is complementary: we contribute a structured representation in which the same query evaluates all ten contracts under identical logic, producing coverage determinations with traceable, clause-level evidence rather than span-level predictions.

The domain representation underlying this work is informed by prior efforts to formalize financial and legal knowledge. FIBO [15] provides a structured vocabulary for financial instruments, entities, and contractual obligations, and LKIF [16] models the structure of legislation and regulatory rules. Both are oriented toward standardization and interoperability at the schema level. Recent work has further explored how KGs can act as an intermediate semantic layer for executable systems. Van Woensel and Seneviratne [17] propose generating blockchain smart contracts directly from KGs, enabling semantic interoperability across distributed healthcare systems while preserving alignment with domain standards. Our TBox draws on the same ontology engineering principles but is purpose-built for the life insurance contract corpus, prioritizing instance-level population and query-based evaluation. Pan et al. [18] frame the complementarity of KGs and LLMs as a general research program, arguing that structured representations provide verifiability that LLMs alone cannot guarantee. Our comparison of SPARQL-based and text-only LLM pipelines is a concrete empirical instance of that argument in the insurance domain.

Automated construction of ontologies and knowledge bases from text is a rapidly growing area of research. Babaei Giglou et al. [19] introduce the LLMs4OL paradigm and evaluate nine LLM families

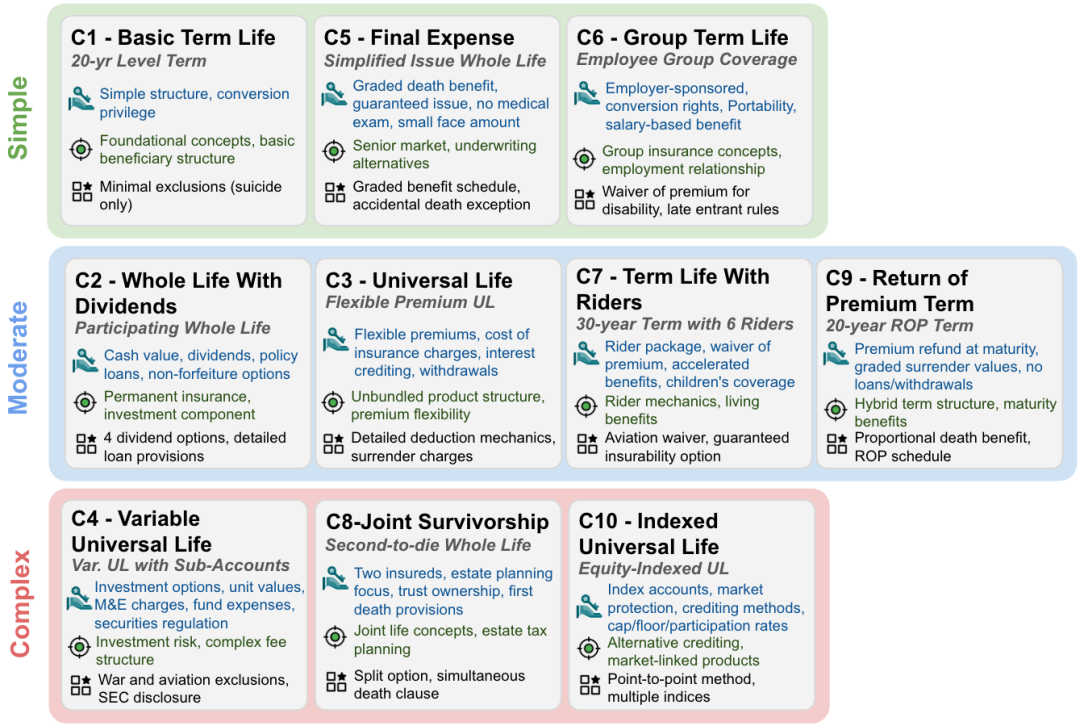

Figure 1: Overview of the 10 life insurance contracts categorized by complexity. The contracts are grouped into three levels: *Simple*, *Moderate*, and *Complex*, based on their complexity and range of features. , , and represent the *Key Features*, *Focus*, and *Uniqueness*, respectively. C1-C10 are the identifiers of the contracts.

on term typing, taxonomy discovery, and non-taxonomic relation extraction, finding that while LLMs capture ontological structure from text, performance varies substantially by domain and task. Saeedizade and Blomqvist [20] show that GPT-4 can produce OWL modeling suggestions of reasonable quality when guided by competency questions, though complex axiom patterns remain difficult. Mihindukulasooriya et al. [21] provide Text2KGBench, a benchmark for evaluating LLM-driven KG generation from text, measuring whether extracted triples conform to a given schema. Collectively, this work demonstrates that automated construction is feasible but that evaluation remains a bottleneck: resources that provide aligned domain-specific NL documents, a verified TBox, and a populated ABox together as a gold standard against which automated pipelines can be measured are very scarce. Our benchmark provides exactly that, and the executable scenarios with clause-level ground truth add a further layer of functional evaluation on top of it.

3. The Dataset: A Benchmark for Insurance KGs

The proposed benchmark is composed of three primary, interlinked components: (i) a curated corpus of diverse life insurance contracts, (ii) a verified domain ontology (TBox) with a corresponding instantiated knowledge base (ABox), and (iii) a comprehensive suite of executable competency questions. Together, these artifacts provide the necessary substrate for evaluating the reproducibility, explainability, and evidence-linked traceability of coverage determinations across heterogeneous contract types.

3.1. The Contracts

We release ten synthetic life insurance contracts, each representing a distinct product type found in the life insurance market. The contracts were developed by first cataloging the major product categories through domain research and then generating representative contract text for each using *Claude Sonnet*

4.6 [22] with carefully structured prompts. Each generated contract was subsequently reviewed and validated by a domain expert to ensure terminological consistency, legal plausibility, and internal coherence across contract clauses. Together, the ten contracts cover a wide range of product types, spanning most major categories of individual and group life insurance commonly found in the market. They range from simple term products to highly structured products such as variable and indexed universal life, providing a diverse but tractable corpus for systematic study. See Figure 1 for a detailed overview of each contract, highlighting their coverage, diversity, and complexity.

The set of contracts is designed with two complementary goals: breadth of coverage and tractability for systematic study. For breadth, the contracts collectively instantiate a wide variety of product structures, underwriting approaches, and provision types. For tractability, each contract is simplified enough that every substantive provision can be mapped to ontology concepts, enabling complete manual annotation. Diversity is evident not only in product type but also in structural variation across shared concepts, which is precisely the kind of variation that motivates gap and overlap analysis. For example, variations like the suicide exclusion clause: most contracts use a 24-month exclusion, but one uses 12 months, with varying benefits. Similarly, the grace period varies between premium non-payment (30-31 days) and account value depletion (61 days). These differences reveal the gaps and overlaps the benchmark seeks to highlight.

3.2. The Ontology

The ontology was constructed through a structured, iterative process. We first used Claude Sonnet 4.6 to generate a base OWL/Turtle ontology from the ten contracts and a detailed prompt specifying the required domain coverage and design constraints. The resulting draft was then subjected to multiple rounds of LLM-assisted review: we independently queried Claude, Gemini, and ChatGPT to identify structural issues, missing classes, incorrect property domains and ranges, and violations of design patterns, and manually resolved each reported issue. Finally, we conducted a complete manual review of the TBox, resolving any remaining inconsistencies. The outcome is a verified ontology whose structure is grounded in standard Semantic Web practice.

3.2.1. TBox and ABox

The TBox defines the vocabulary and structure of the life insurance domain in OWL/Turtle. It is organized around a root `Policy` class from which all product-type subclasses descend via `rdfs:subClassOf`, covering the full range of product families represented in the corpus. Alongside the policy hierarchy, the TBox covers parties (`Insured`, `PolicyOwner`, `Beneficiary`, `Trust`, `Employer`), coverage and benefit types, riders, premiums, grace periods, cash value and crediting mechanisms, charges, exclusions, and provisions. All classes and properties carry `rdfs:label` and `rdfs:comment` annotations; the comments record which contracts instantiate each concept and under what conditions. Following is an example that shows the high level modeling of suicide exclusion in the life insurance contracts.

```
li:SuicideExclusion a owl:Class ; rdfs:subClassOf li:Exclusion ;
    rdfs:label "Suicide Exclusion" ;
    rdfs:comment "Present in ALL 10 contracts. Period: 12 months (C6 only); 24 months (
        C1,C2,C3,C4,C5,C7,C8,C9,C10). Benefit outcome varies - use li:suicideBenefitType
         object property with li:SuicideBenefitType individuals." .
```

The ABox instantiates one policy individual per contract (C1–C10), each linked through object properties to supporting individuals representing the insured, policy owner, insurer, beneficiaries, death benefit, premium, grace period, exclusions, provisions, riders, cash value, charges, and crediting mechanisms. Product-specific structures such as investment sub-accounts, index accounts, irrevocable trust ownership, etc receive dedicated individuals where required. The following excerpt from C1's ABox shows an example of how policy individuals are instantiated.

```
li_abox:SuicideExclusion_C1 a li:SuicideExclusion ;
    li:suicideExclusionPeriodMonths 24 ;
    li:suicideBenefitType li:ReturnPremiumsPaid ;
    li:sourceContractID "C1" ;
    li:sourceContractSection "Section 7.1" ;
    li:coverageSourceText "SUICIDE CLAUSE: If the insured commits suicide within two
        years from the issue date, our liability is limited to a refund of premiums paid
        ." ; .
```

Every ABox individual is annotated with three traceability properties: `sourceContractID`, `sourceContractSection`, and `coverageSourceText` (verbatim or close-paraphrase of the originating contract clause). These annotations ensure that any SPARQL query result can be traced directly to the supporting contract text, which is a prerequisite for explainable gap and overlap analysis. Instance density varies with product complexity: simpler term policies require fewer individuals covering their essential provisions, while the variable universal life and indexed universal life policies require substantially more to represent sub-accounts, multiple charge types, exclusions, and crediting mechanisms in full.

3.2.2. Qualitative Analysis

The correctness of gap and overlap determinations in this context depends heavily on the precision of the ontology's structural commitments. Poorly defined class boundaries or inconsistent property representations can cause contracts with genuinely different clauses to produce identical query results, obscuring real gaps or introducing false overlaps. A well-structured TBox ensures that parametric differences such as exclusion periods or benefit types are encoded as typed, queryable values, enabling deterministic, evidence-linked analysis. Therefore, our TBox design choices follow established ontology engineering practices while introducing domain-specific qualities that make it suitable as a ground-truth reference. Its class hierarchy is organized around the **subsumption pattern**, with `Policy`, `Party`, `Coverage`, `Exclusion`, `Provision`, and `Rider` serving as abstract superclasses from which domain-specific subclasses inherit. Depth is proportional to domain complexity: the `Policy` branch reaches four levels (e.g., *Policy* → *PermanentLifePolicy* → *UniversalLifePolicy* → *IndexedUniversalLifePolicy*), while shallower branches reflect concepts where the contracts do not require further distinction. Party roles such as `FirstInsured`, `SecondInsured`, `IrrevocableTrust`, `RevocableTrust`, etc., are modeled as distinct classes rather than collapsed into generic role-bearing individuals, keeping property domain and range constraints precise.

Where a concept admits a fixed set of mutually exclusive values, the TBox applies the **value partition pattern**: named individuals typed against a dedicated class replace free-text literals. For instance, the `SuicideBenefitType` class enumerates five named individuals, each mapped to specific contracts via a typed object property, enabling a reasoner or SPARQL endpoint to test for contract behaviors formally rather than through string matching. This process also maintains a clean **TBox/ABox separation**: the TBox defines only classes, properties, and controlled-vocabulary individuals, leaving all contract-specific assertions to the ABox, which allows the schema to function as a stable, reusable vocabulary against which multiple ABoxes can be independently validated.

Object and datatype properties carry explicit `rdfs:domain` and `rdfs:range` declarations, with datatype properties bound to appropriate XSD types (`xsd:integer`, `xsd:decimal`, `xsd:boolean`, `xsd:date`). This type discipline is functional: properties such as `exclusionIsTimeBounded`, `exclusionPeriodMonths`, and `suicideExclusionPeriodMonths` work in combination to let a reasoner distinguish permanent from time-bounded exclusions without parsing narrative text.

At the schema level, every class and property carries an `rdfs:comment` annotation recording which contracts instantiate it and under what parametric conditions, including cross-contract variation and known modeling ambiguities, serving as human-readable documentation embedded directly in the ontology. More importantly, this annotation is operationalized at the ABox level through three dedicated datatype properties defined in the TBox: `sourceContractID`, `sourceContractSection`, and

coverageSourceText. These are declared with rdfs:domain owl:Thing, making them applicable to any instance, and are required on every ABox individual to anchor each assertion to a specific contract identifier, section reference, and supporting contract language. This design ensures that every populated fact in the knowledge base is traceable to its documentary source, which is particularly valuable in legal and regulatory domains where the auditability of extracted knowledge matters.

Figure 2 presents a representative excerpt that jointly illustrates the qualities described above. The Exclusion superclass and SuicideExclusion subclass demonstrate the subsumption pattern; the SuicideBenefitType class with its four named individuals demonstrates the value partition pattern; the suicideBenefitType object property and suicideExclusionPeriodMonths datatype property demonstrate typed property constraints with explicit domain and range declarations; and the rdfs:comment on SuicideExclusion, recording per-contract exclusion periods across all ten contracts, demonstrates the annotation practice.

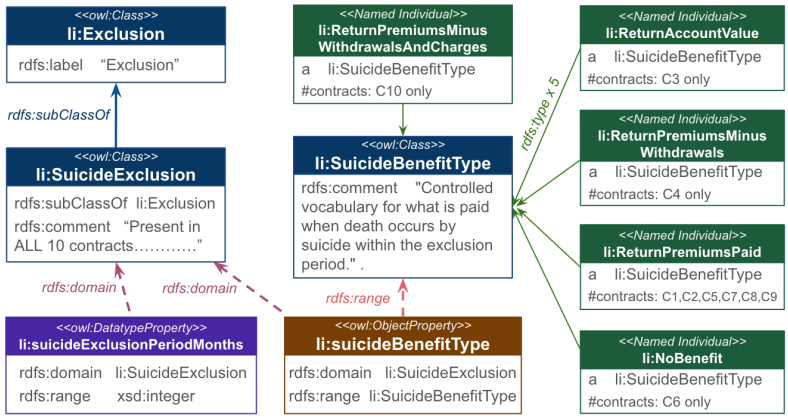

Figure 2: A representative TBox excerpt.

3.2.3. Ontology Coverage Analysis

To assess how well the TBox captures the domain vocabulary expressed in the contracts, we conducted a keyphrase-based coverage analysis. The keyphrases were extracted from each contract using Claude Sonnet 4.6, resulting in a set of 416 phrases drawn across all ten contracts. These were matched against human-readable TBox artifacts: classes, properties, and labels using three complementary methods:

1. **Exact / substring matching** — all component words of a keyphrase are present in the target label (e.g., *face amount* → faceAmount);
2. **Fuzzy matching** — RapidFuzz[1] token-sort ratio $\geq$ 70 (e.g., *current credited interest rate* → creditingCurrentRate);
3. **Semantic similarity** — SentenceTransformer all-MiniLM-L6-v2 [2] cosine similarity ≥ 0.70 (e.g., *time-bounded exclusion* → exclusionIsTimeBounded).

A keyphrase is considered covered if *any* method produces a match.

As shown in Figure 3, 352 of 416 keyphrases (84.6%) were successfully mapped. The majority were captured by exact or substring matching (244), with fuzzy matching and semantic similarity contributing a further 84 and 24 matches respectively.

However, manual inspection shows that the 64 keyphrases that were not matched by any automated method do not necessarily represent gaps in the ontology. We found that in most of these cases, the concept is captured by the TBox under a different label or through a richer representational structure that the string-level matching pipeline cannot resolve. A concrete example is the keyphrase *'coverage continues during grace period'*, which no method maps to an ontology label. Yet the rdfs:comment of class PremiumNonpaymentGracePeriod explicitly states:

> *"Triggered by failure to pay a scheduled premium.* ***Coverage continues*** *for specified days after premium due date. C1: 31 days, C2: 31 days, C5: 30 days, C7: 31 days …, C9: 31 days."*

[1] https://pypi.org/project/RapidFuzz
[2] https://pypi.org/project/sentence-transformers

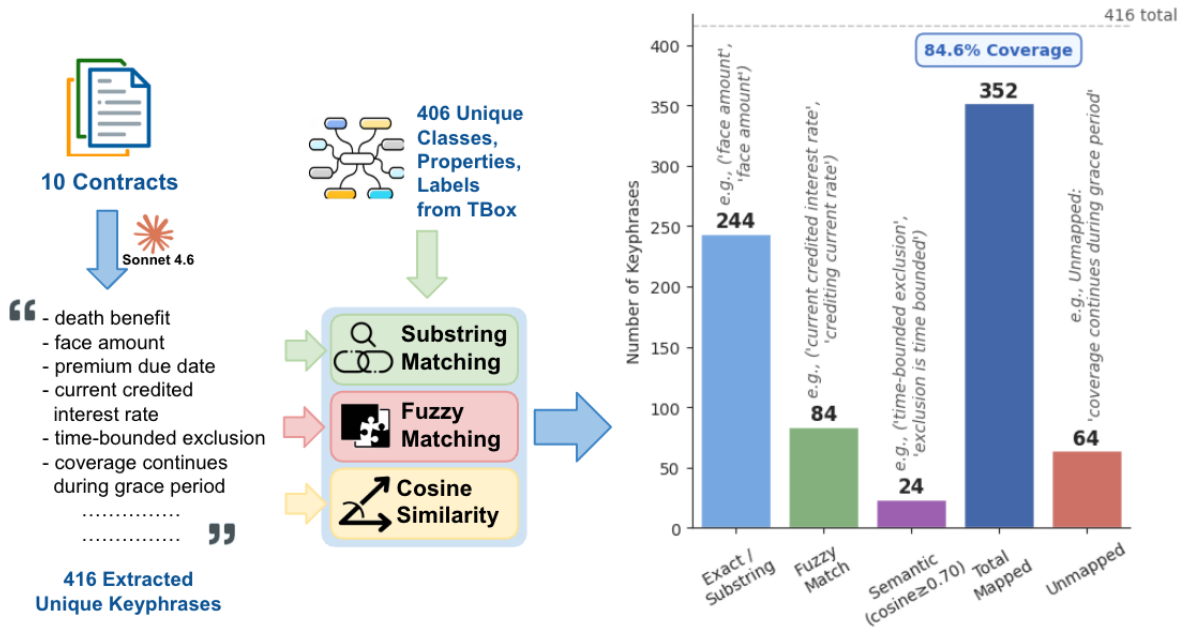

Figure 3: Ontology Coverage Analysis. Extracted keyphrases from the contracts are matched against the ontology TBox artifacts (classes, properties, labels) using three different matching techniques. Example tuples (s, o) denote the **source contract term** (s) and its corresponding **mapped ontology artifact** (o).

The semantic content of the keyphrase is fully encoded; it is the surface-form divergence between the phrase and the class label that causes the miss. Similar patterns hold for other flagged terms: *lump sum payment* is captured by `SettlementOptionProvision` with a `coverageSourceText` annotation stating the default lump-sum payout; *proof of death* maps to `DeathCertificate` under the `ClaimRequirement` hierarchy; *outstanding loan* is encoded through `PolicyLoan` and the `deathBenefitFormula` property. The coverage analysis therefore confirms that the ontology is comprehensively aligned with the contract corpus.

3.3. Competency Question Suite

To evaluate the task-readiness of the TBox and ABox, we developed a suite of **58 structured scenarios** that represent common competency questions a stakeholder might ask when performing gap and overlap analysis. These scenarios are organized into the following key categories: Basic Death Claims, Loans and Conversions, Exclusions and Riders, Joint/Special Features, Advanced Product Mechanics, and Expert Provided Edge Cases.

Each scenario follows a standardized JSON structure designed for systematic benchmarking. Each entry includes:

- **Scenario Description**: A natural language prompt detailing a specific insurance event (e.g., SCEN-003: "Insured dies by suicide exactly 13 months after the policy issue date. All premiums paid on time.").
- **Double-Query Ground Truth**: Two distinct SPARQL queries, one to identify contracts where the claim is **covered** and one where it is **denied**.
- **Contract-Level Outcomes**: Manually verified ground truth for each of the ten contracts (C1–C10), detailing the specific status: COVERED, DENIED, or NOT_APPLICABLE.
- **Traceability Evidence**: For every outcome, we include the verbatim `sourceText` and `sourceSection` from the original contract to ensure results are auditable and explainable.

By providing two SPARQL queries per scenario, we enable a robust deterministic evaluation that measures the accuracy of both positive and negative claim determinations with evidence-based traceability.

The suite was initially generated using the same LLM-assisted workflow as the contracts, followed by **manual inspection** to ensure they are realistic and cover a good portion of the cases within the contracts. To further ensure domain accuracy, we had the suite **reviewed by a domain expert** and included several specialized test cases (8 out of the 58 total). These additions probe complex legal and administrative nuances, such as simultaneous death determination, lapses during grace periods on joint policies, the impact of pre-existing conditions on accelerated benefits.

4. Analysis Through Use Case: Gap and Overlap Analysis

This section operationalizes the benchmark through a concrete downstream use case: *gap and overlap analysis* across a set of insurance contracts. The goal is not to prescribe a single *correct* notion of gap/overlap, but to show that once contracts are represented using an ontology (TBox) with populated instances (ABox) and paired with executable scenario queries, the dataset supports precise, repeatable such analyses.

In this paper, *gap* and *overlap* are defined *relative to a scenario* s (e.g., a claim situation with conditions such as timing, cause, and policy state) and a contract set $\mathcal{C}$. The scenarios may originate from different perspectives.

From a policyholder perspective, an insured typically asks: "If this scenario happens to me, which contracts would help me, and where would I be unprotected?" We define:

$$\mathrm{Overlap}_{\text{insured}}(s) \;=\; \{\, c \in \mathcal{C} \mid s \text{ is } \textsc{Covered} \text{ under } c \,\},$$

i.e., *overlap* is the set of contracts that cover the scenario. Intuitively, overlap indicates multiple alternatives (or redundancies) that would pay out under the same scenario.

We define the insured-centric *gap* as the set of contracts that do *not* provide protection for the scenario, either due to explicit denial or because the scenario does not meaningfully apply under that product's structure:

$$\mathrm{Gap}_{\text{insured}}(s) \;=\; \{\, c \in \mathcal{C} \mid s \text{ is } \textsc{Denied} \text{ or } \textsc{Not\_Applicable} \text{ under } c \,\}.$$

This definition is scenario-centric: *gap* is not *missing text*, but rather *lack of coverage for the insured under the scenario*, which can arise via exclusionary conditions or structural non-applicability.

Other stakeholders such as analysts, product designers, or regulators might ask: "How consistently are certain risks addressed across products, and where do products diverge?" In this view, gap/overlap can be defined in multiple ways, for example: overlap as common coverage across many contracts, overlap as common denial patterns (shared exclusion regimes), or gap as portfolio absence (no contract covers).

However, our scenario set is primarily insured-centric. Since the benchmark releases the full set of aligned artifacts: contract text, a shared TBox, contract-specific ABox populations, and executable scenario queries with clause-level annotation, additional viewpoints can be incorporated by simply adding new scenarios and corresponding query pairs. In this sense, the benchmark does not claim to exhaust all perspectives in its current scenario set; rather, it provides a representation and evaluation substrate that makes such extensions straightforward and interoperable with the existing resources.

4.1. Query Outcomes

For each scenario s, the benchmark provides two SPARQL queries: (i) a coverage query Q_s^+ that retrieves contracts for which the scenario is Covered, and (ii) a denial query Q_s^- that retrieves contracts for which the scenario is Denied. Thus, for each scenario–contract pair (s, c), the benchmark assigns exactly one outcome label: COVERED, DENIED, or NOT_APPLICABLE. Collecting these labels over all

scenarios and contracts yields a scenario-by-contract outcome matrix (M). Let $\mathrm{Ans}(Q)$ denote the set of contracts returned by query Q. The outcome for each contract $c \in \mathcal{C}$ is computed as:

$$\begin{aligned} M[s,c] &= \textsc{Covered} && \text{if } c \in \mathrm{Ans}(Q_s^+), \\ M[s,c] &= \textsc{Denied} && \text{if } c \in \mathrm{Ans}(Q_s^-), \\ M[s,c] &= \textsc{Not\_Applicable} && \text{if } c \notin \mathrm{Ans}(Q_s^+) \ \wedge \ c \notin \mathrm{Ans}(Q_s^-). \end{aligned}$$

In this design, Not_Applicable is not a "third query," but the residual class induced by the pair (Q_s^+, Q_s^-). It captures situations where the scenario does not match the product structure or the relevant mechanism is outside the contract's modeled scope for that scenario family.

4.2. Gap/Overlap Summaries

Executing the scenario queries over all $s \in \mathcal{S}$ yields a scenario-by-contract outcome matrix:

$$M[s,c] \in \{\textsc{Covered}, \textsc{Denied}, \textsc{Not\_Applicable}\}.$$

From M, insured-centric overlap and gap can be computed directly:

$$\begin{aligned} |\mathrm{Overlap}_{\text{insured}}(s)| &= |\{\, c \in \mathcal{C} \mid M[s,c] = \textsc{Covered} \,\}|, \\ |\mathrm{Gap}_{\text{insured}}(s)| &= |\{\, c \in \mathcal{C} \mid M[s,c] \in \{\textsc{Denied}, \textsc{Not\_Applicable}\} \,\}|. \end{aligned}$$

At the same time, the same matrix supports other stakeholder summaries without modifying the benchmark, e.g., *denial overlap* (contracts that share denial outcomes for s), *portfolio coverage gaps* (no contract covers a scenario),

$$\forall c \in \mathcal{C}, \ M[s,c] \neq \textsc{Covered},$$

or patterns of structural non-applicability (scenarios systematically Not_Applicable for certain product families), reflecting genuine product design differences rather than errors. The key point is that the benchmark provides a foundation (explicit structure plus executable scenario queries) upon which multiple gap/overlap interpretations can be layered.

4.3. Comparison with a Text-Only LLM Pipeline

To assess how well current large language models perform on the task, we evaluated the same 58 scenarios using a text-only LLM pipeline. We used three state-of-the-art LLMs: *ChatGPT-5.3* [23], *Gemini-3* [24], and *Claude Sonnet-4.6* [22]. Each model was accessed through its public web interface and prompted with the same structured instructions to analyze the contracts and scenarios. The prompt required the model to determine, for every contract-scenario pair, whether the scenario was COVERED, DENIED, or NOT_APPLICABLE, and to support the classification with a relevant clause or excerpt from the contract text.

The 58 scenarios, applied to 10 contracts, produces **580 contract-scenario evaluations per model**. Since three models were tested, the evaluation comprises **1,740 total classifications**. Model outputs were parsed into the required JSON format and compared against the ground-truth labels derived from the ontology-backed contract representations.

Table 1 summarizes the overall accuracy of each model. Claude achieved the strongest performance at **87.76%** ($509/580$), followed by ChatGPT at **72.93%** ($423/580$) and Gemini at **65.17%** ($378/580$). While these results indicate that these LLMs can mostly understand contracts terms, a more detailed analysis reveals variability in both reasoning behavior and interpretation of contractual scope.

A useful way to further understand this variability is through **inter-model agreement**, which captures whether the models arrive at the same judgment for a given contract–scenario pair. As shown in Figure 4, among the 580 evaluated pairs, the models produced identical correct answers in **283 cases (48.8%)**, while **203 cases (35.0%)** were resolved correctly by a majority of models but not unanimously. At the same time, **55 cases (9.5%)** produced mixed outcomes, **33 cases (5.7%)** resulted in unanimous but

incorrect answers, and **6 cases (1.0%)** exhibited complete disagreement across all three models. These results indicate that while LLMs often agree on straightforward scenarios, their reasoning becomes less stable when scenarios are complex.

Further insight emerges when examining which contracts produce the largest number of errors. For Claude, the five most error-prone contracts are C8, C9, C6, C10, and C4. For ChatGPT, they are C8, C4, C10, C7, and C9. For Gemini, they are C6, C1, C9, C8, and C7. Notably, **C4** (Variable Universal Life), **C8** (Joint Survivorship), and **C10** (Indexed Universal Life), **the three most complex contracts** in the benchmark **all appear among the top-five most error-prone contracts** for a majority of the evaluated models. These results suggest, as contractual complexity increases, LLMs become more prone to misinterpretation or incomplete reasoning.

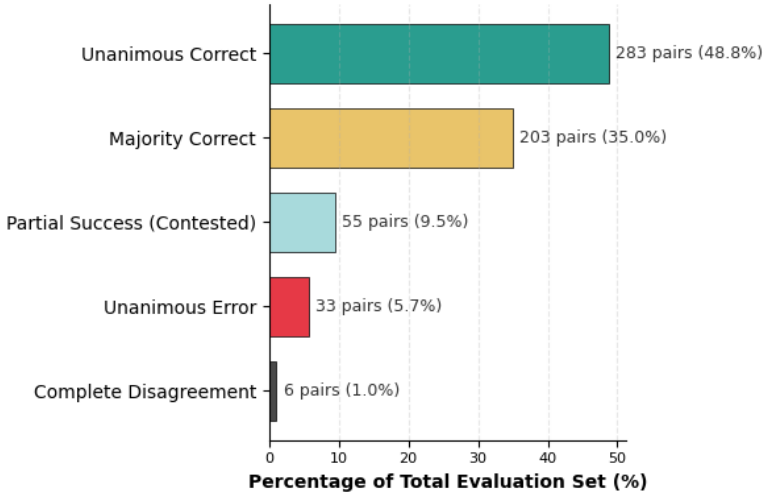

Figure 4: Inter-LLM agreement patterns across the 580 contract-scenario pairs.

To better understand the nature of these failures, we examined the distribution of mismatch types between model predictions and ground truth.

As shown in Table 2, the most common errors are NOT_APPLICABLE → DENIED (**207**) and NOT_APPLICABLE → COVERED (**126**), followed by much smaller numbers of COVERED → DENIED (**30**), COVERED → NOT_APPLICABLE (**30**), and DENIED → COVERED (**23**). The dominance of the NOT_APPLICABLE → DENIED pattern highlights a recurring difference between the benchmark's contract interpretation logic and the reasoning heuristics used by LLMs. In the benchmark annotations, NOT_APPLICABLE indicates that the contract does not contain clauses governing the scenario. In contrast, LLMs frequently interpret the absence of such clauses as grounds for denying the claim.

For example, for the scenario: *"The insured dies in a car accident. Toxicology confirms the insured had a blood alcohol concentration above the legal limit at the time of the accident. The beneficiary files an accidental death benefit claim"*, as the ground truth, most contracts classify this scenario as COVERED. The reason is that these policies do *not* include an accidental death rider or a drug/alcohol **exclusion** clause, meaning that the standard death benefit remains payable. However, many LLM responses classified the scenario as DENIED or NOT_APPLICABLE, apparently because of no mention of intoxication-related exclusions. This also helps explain the observed COVERED → NOT_APPLICABLE and COVERED → DENIED mismatches.

A second example illustrates how inter-LLM disagreement becomes especially visible when the scenario depends on a specific policy structure. Consider the scenario: *"On a joint survivorship policy, the first insured dies with an outstanding policy loan of $50,000 at 5.5% annual interest. The surviving*

Table 1
Overall accuracy of the evaluated LLMs on the benchmark.

Model	Accuracy
Claude Sonnet 4.6	**87.76%** (509/580)
ChatGPT-5.3	72.93% (423/580)
Gemini-3	65.17% (378/580)

Table 2
Most frequent mismatch types between ground truth and LLM predictions.

Mismatch type (TRUE → PRED.)	**Count**
NOT_APPLICABLE → DENIED	207
NOT_APPLICABLE → COVERED	126
COVERED → DENIED	30
COVERED → NOT_APPLICABLE	30
DENIED → COVERED	23

insured and trustee want to know how the loan is treated during the continuation period and its impact on the eventual death benefit." This is **one of the benchmark's eight special-case** scenarios. Only **Contract C8** is a *joint survivorship policy*, so the correct labeling is COVERED for C8 and NOT_APPLICABLE for all other contracts. In this case, the three models exhibited completely different reasoning behaviors. Claude correctly recognized the structural constraint, identified that the scenario is meaningful only for the joint policy, and cited the relevant loan clauses governing treatment before and after the first death. ChatGPT instead marked nearly all contracts as COVERED, apparently inferring that any death-benefit clause was sufficient for applicability, while Gemini marked most contracts as DENIED, often citing generic product descriptions that do not substantively justify the classification. This example demonstrates a setting in which the models diverge completely in how they interpret applicability, coverage, and supporting evidence.

A third representative case explains why NOT_APPLICABLE → DENIED is by far the most common mismatch. Consider the scenario: *"The policy owner wishes to take out a loan against the policy's cash value or account value shortly after the policy has been in force for one year."* For Contract **C9**, the correct label is NOT_APPLICABLE, because the contract does not have any clauses regarding policy loans. However, all three LLMs classified the scenario as DENIED, using the absence of such clauses as justification. This suggests that the models are not necessarily missing the relevant text; rather, they are applying a different decision logic from the benchmark.

Beyond the classification errors themselves, the examples above reveal another important pattern: the **quality of evidence citations** provided by LLMs varies substantially. In several cases, the cited text is only loosely connected to the assigned label, such as generic product descriptions or broad death-benefit statements. This is especially visible in the joint survivorship example, where Gemini and ChatGPT often cited short descriptions like "term life insurance" or "whole life insurance" for contracts to which the scenario does not even apply. Such outputs may appear superficially well-formed, but they are not evidence-grounded in the sense required for reproducible analysis.

Taken together, these findings highlight a fundamental distinction between **ontology-driven reasoning** and **LLM-based interpretation**. In the ontology-based approach described earlier, scenarios are evaluated through explicit queries over formally defined policy concepts and relationships. Because these representations encode the structure of the contract, query results depend solely on the modeled terms present in the policy. In contrast, LLMs may incorporate implicit assumptions, background knowledge, or heuristic interpretations that lead to plausible but unsupported conclusions. This distinction helps explain why LLMs can perform reasonably well on straightforward cases while remaining inconsistent on cases that require strict handling of contractual scope and structural constraints.

4.4. Gap/Overlap Analysis as a Test of KG Task Readiness

The analysis above speaks directly to the claim made in the introduction: that gap and overlap analysis serves as a useful test of *task readiness* for a KG, asking whether an ontology-based representation can surface coverage differences consistently and supply evidence for each outcome. The SPARQL-driven pipeline answers both parts of that question affirmatively and in a way that the text-only LLM baseline cannot match. Consistency is guaranteed structurally: because the ABox encodes each relevant provision as a typed, property-bearing individual with explicit domain and range constraints,

the same query logic applies uniformly across all ten contracts, with no variation in how parametric differences such as exclusion periods, grace period triggers, benefit outcome types, etc., are evaluated. Evidence is produced automatically and traceably: every query result carries the `sourceContractID`, `sourceContractSection`, and `coverageSourceText` of the supporting clause, so each outcome is not merely asserted but justified by the originating contract text. We note that this consistency is a direct consequence of the manual knowledge modeling effort, and that is exactly what we require for a reproducible and auditable benchmark.

As demonstrated in the previous section, the LLM pipeline, operating on unstructured text, can approximate coverage determinations in straightforward cases but cannot guarantee consistency for complex cases, and cannot always produce structured evidence trails. The difference between the two approaches is therefore not merely a gap in language understanding, but a gap in *representational commitment*: the ontology forces the modeler to resolve ambiguities, enumerate value-partition members, and declare domain and range constraints at construction time. It is that modeling effort that enables deterministic, evidence-linked analysis at query time. In this sense, the benchmark validates the original motivation: structured KG representation does not merely organize contract knowledge, it makes that knowledge queriable with evidence.

5. Conclusion and Future Work

This paper presents an executable benchmark for KG quality evaluation through gap and overlap analysis on life insurance contracts. The benchmark aligns three resources: ten domain expert verified synthetic contracts, a domain ontology (TBox) with a fully populated and evidence-annotated ABox in which every instance is anchored to a contract identifier, section reference, and verbatim clause text, and 58 structured scenarios with SPARQL-based ground truth and clause-level annotation. We demonstrated the benchmark through a gap and overlap use case, showing that the ontology-driven pipeline produces deterministic, fully traceable coverage determinations across all ten contracts, and compared it against a text-only LLM baseline to illustrate where unstructured inference degrades and why representational commitment matters for consistent, auditable analysis.

However, we acknowledge several limitations of the present benchmark. The contracts are synthetic, although expert-reviewed for legal plausibility and internal consistency, external validity may be limited when compared to proprietary, naturally occurring contracts. Also, the benchmark is instantiated particularly in life insurance, though the methodology is applicable beyond this domain. The ontology is purpose-built to serve the benchmark rather than as a general domain ontology, and extending it toward broader coverage remains a natural direction for future work. Moreover, the ontology was bootstrapped with LLM assistance, which may introduce some modeling bias despite multi-model review and manual validation. Additionally, the scenario set is diverse but not exhaustive: it can be extended to broaden coverage. Finally, the LLM evaluation uses a text-only baseline with fixed prompting, and stronger prompting or retrieval-augmented methods may perform differently.

Beyond gap and overlap analysis, the benchmark's aligned resources support a range of downstream tasks. The contract corpus paired with the verified TBox and ABox provides a natural testbed for ontology learning and KG population methods, where automatically constructed representations can be evaluated against the ground truth end-to-end. The clause-level annotations make the benchmark directly applicable to evidence-grounded question answering and explainable retrieval tasks. The scenario suite, with its structured outcomes across ten contracts, also supports contract reasoning and entailment tasks in the legal text processing research. More broadly, the benchmark's structure: aligned document text, a verified TBox, an annotated ABox, and executable scenario queries, is a domain-agnostic template, and extending it to other policy-like domains such as healthcare coverage or financial regulation is a natural next step toward a shared infrastructure for KG quality evaluation across legal and regulatory document collections. It also opens a direction toward hybrid neuro-symbolic pipelines. Our evaluation deliberately contrasts a pure neural approach with a pure symbolic one, but the two need not be mutually exclusive: an LLM could extract structured features that drive ontology-backed

reasoning, or translate natural language scenarios into formal SPARQL queries, combining the language fluency of neural models with the consistency and traceability guarantees of the KG. The benchmark's clause-level ground truth and executable queries are the resources needed to develop and evaluate such approaches, and this could be a promising direction for future work.

Acknowledgments

We acknowledge the support from NSF IUCRC CRAFT center research grant (CRAFT Grant #22022) for this research. The opinions expressed in this publication do not necessarily represent the views of NSF IUCRC CRAFT. We are also grateful for the advice and resources from our CRAFT Industry Board members in shaping this work.

Declaration on Generative AI

During the preparation of this work, the authors used Claude Sonnet, ChatGPT, and Gemini for dataset construction purposes, as described in detail within the paper. Additionally, the authors used Claude Sonnet and ChatGPT to polish and improve the clarity of the text originally drafted by the authors, and for assistance with minor code segments during implementation. After using these services, the authors reviewed and edited all AI-assisted content as needed and take full responsibility for the publication's content.

References

[1] B. Xue, L. Zou, Knowledge graph quality management: a comprehensive survey, IEEE Transactions on Knowledge and Data Engineering 35 (2022) 4969–4988.

[2] S. Tsaneva, D. Dessì, F. Osborne, M. Sabou, Knowledge graph validation by integrating llms and human-in-the-loop, Information Processing & Management 62 (2025) 104145.

[3] C. Bezerra, F. Freitas, F. Santana, Evaluating ontologies with competency questions, in: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), volume 3, IEEE, 2013, pp. 284–285.

[4] W. Araújo, G. Lima, I. Pierozzi Jr, Data-driven ontology evaluation based on competency questions: A study in the agricultural domain, in: Knowledge Organization for a Sustainable World: Challenges and Perspectives for Cultural, Scientific, and Technological Sharing in a Connected Society, Ergon-Verlag, 2016, pp. 326–332.

[5] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, Quality assessment for linked data: A survey, Semantic Web 7 (2016) 63–93. doi:10.3233/SW-150175.

[6] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semantic web 8 (2016) 489–508.

[7] O. Seneviratne, B. Capuzzo, W. Van Woensel, Explainability-driven quality assessment for rule-based systems, in: Companion Proceedings of the ACM on Web Conference 2025, WWW '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 2133–2140. URL: https://doi.org/10.1145/3701716.3717571. doi:10.1145/3701716.3717571.

[8] M. Gruninger, Methodology for the design and evaluation of ontologies, in: Proc. IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.

[9] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), W3C Recommendation, W3C, 2017. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/.

[10] Y. Koreeda, C. D. Manning, Contractnli: A dataset for document-level natural language inference for contracts, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 1907–1919.

[11] D. Hendrycks, C. Burns, A. Chen, S. Ball, Cuad: An expert-annotated nlp dataset for legal contract review, arXiv preprint arXiv:2103.06268 (2021).

[12] I. Kang, W. V. Woensel, O. Seneviratne, Using Large Language Models for Generating Smart Contracts for Health Insurance from Textual Policies, Springer Nature Switzerland, Cham, 2024, pp. 129–146. URL: https://doi.org/10.1007/978-3-031-63592-2_11. doi:10.1007/978-3-031-63592-2_11.
[13] O. Seneviratne, A. Gupta, M. Ahmed, Towards Smarter, Efficient and Trusted Insurance Marketplaces through Computable Contracts, in: CapGemini White Papers, 2022. URL: https://prod.ucwe.capgemini.com/wp-content/uploads/2022/06/Computable_Contracts_20.pdf.
[14] W. V. Woensel, M. Shukla, O. Seneviratne, Translating clinical decision logic within knowledge graphs to smart contracts, in: SeWeBMeDa@ ESWC, 2023. URL: https://ceur-ws.org/Vol-3466/paper3.pdf.
[15] M. Bennett, The financial industry business ontology: Best practice for big data, Journal of Banking Regulation 14 (2013) 255–268.
[16] R. Hoekstra, J. Breuker, M. Di Bello, A. Boer, et al., The lkif core ontology of basic legal concepts., LOAIT 321 (2007) 43–63.
[17] W. Van Woensel, O. Seneviratne, Semantic interoperability on blockchain by generating smart contracts based on knowledge graphs, Blockchain: Research and Applications 7 (2025) 100320. URL: https://www.sciencedirect.com/science/article/pii/S2096720925000478. doi:https://doi.org/10.1016/j.bcra.2025.100320.
[18] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, IEEE Transactions on Knowledge and Data Engineering 36 (2024) 3580–3599.
[19] H. Babaei Giglou, J. D'Souza, S. Auer, Llms4ol: Large language models for ontology learning, in: International semantic web conference, Springer, 2023, pp. 408–427.
[20] M. J. Saeedizade, E. Blomqvist, Navigating ontology development with large language models, in: European semantic web conference, Springer, 2024, pp. 143–161.
[21] N. Mihindukulasooriya, S. Tiwari, C. F. Enguix, K. Lata, Text2kgbench: A benchmark for ontology-driven knowledge graph generation from text, in: International semantic web conference, Springer, 2023, pp. 247–265.
[22] Anthropic, Claude sonnet 4.6, 2026. URL: https://claude.ai/, accessed: 2026-03-08.
[23] OpenAI, Chatgpt (version 5.3), 2026. URL: https://chatgpt.com/, model: gpt-5.3-chat-latest. Accessed: 2026-03-08.
[24] Google DeepMind, Gemini 3 flash, 2025. URL: https://gemini.google.com/, large Language Model. Accessed: 2026-03-08.