

Making the Implicit Explicit: A Domain Ontology-Based Approach for Comprehensive Attribute-Level Knowledge Extraction in Technical Texts

Vatosoa Razafiniary

Enedis, France

vatosoa.razafiniary@enedis.fr

Abstract

Technical texts heavily rely on implicit domain knowledge that human experts readily infer but computational systems often overlook. To address this challenge, we propose an ontology-driven approach for fine-grained knowledge extraction at the attribute level.

Applied to French electrical Grid Connection Solutions (GCS), the approach operationalizes and quantifies implicit object formulations in technical descriptions. By leveraging domain ontology semantics, it identifies explicit operational entities and their associated attributes, uses this information as evidence for implicit object formulations, and infers operational objects that are not explicitly mentioned in the text.

The resulting extraction framework captures both explicit and implicit knowledge, providing a robust foundation for knowledge graph construction and enabling advanced downstream tasks such as operational pattern discovery and expert-level reasoning.

1 Introduction

Industrial technical descriptions often contain operational knowledge expressed through implicit formulations that are easily interpreted by domain experts but difficult to process automatically (Lamazzi et al., 2026). In the context of electrical grid connection, French textual descriptions of Grid Connection Solutions (GCS) include domain-specific terminology and implicit references to operational objects, complicating automatic information extraction and structuring.

This work addresses the problem of designing a Natural Language Processing (NLP) system capable of identifying operational patterns in GCS descriptions and supporting pattern-based use cases, such as complexity evaluation and solution conformity assessment, in a manner consistent with domain expert reasoning. Following Berry (1987),

we consider implicitness as a central obstacle to knowledge structuring. More specifically, we focus on *attribute-driven implicitness*, where operational objects are indirectly expressed through their attributes or contextual descriptions. We argue that this level of granularity is essential, as such implicit semantics are typically overlooked by standard information extraction pipelines and remain a blind spot for current Language Model (LM)-based approaches (Bakker et al., 2024). To address this problem, we propose a domain ontology-guided NLP pipeline combining Named Entity Recognition (NER), explicit/implicit zone identification, and inference mechanisms for reconstructing operational objects omitted from the text.

Due to industrial confidentiality constraints, the experimental corpus cannot be publicly released. However, reproducibility is supported through detailed documentation of the ontology design process, framework architecture, algorithmic workflow, corpus characteristics, and evaluation protocol.

The contributions of this paper are threefold :

- We derive and formalize the notion of attribute-driven implicitness from an empirical study of French GCS descriptions (Section 3);
- We propose an ontology-guided framework for detecting and reconstructing implicit operational objects (Sections 4, 5, and 6);
- We evaluate the proposed approach in an industrial setting using real-world French GCS descriptions (Section 7).

2 Background

2.1 Ontologies in Technical Text Understanding

In computer science, an ontology is commonly defined as “an explicit specification of a conceptu-

alization” (Gruber, 1993). It provides a formal representation of domain concepts and their relationships, enabling structured semantic interpretation and reasoning over implicit knowledge. Giray and Ünalır (2020), argued that ontology-based representations improve text coherence assessment by modeling semantic relations between concepts and enabling the discovery of implicitly stated facts through inference mechanisms. In our approach, we consider the design of a domain ontology as a prerequisite to the downstream structuring process used to extract attribute-level knowledge from GCS descriptions (see Fig. 1; Section 4).

2.2 Ontology-Based Information Extraction Approaches

Linguistic-Symbolic World Model Approach

Ontology-based information extraction has a rich tradition in technical text understanding. C. and Z. (1992) introduced a *World Model* perspective, viewing sentence comprehension as the construction of structured representations of entities, events, and relations, by pairing a Lexical-Functional Grammar parser with domain-specific semantic reasoning. Yasavur et al. (2013) provide an applied example of this approach through an ontology-based NER framework for the behavioral health domain. Subsequent work (Mahesh et al. (1995), Klein and Potter (2004), Estival et al. (2004)) further incorporated domain vocabularies and semantic constraints, establishing a foundation for domain-grounded symbolic language understanding.

While NLP later shifted strongly toward statistical and connectionist approaches, symbolic approaches based on structured linguistic and domain representations continue to be used in specialized regulatory and engineering applications. For instance, Xu and Cai (2021) combine pattern-based linguistic extraction, ontology mapping, and Description Logic clauses to reason over complex underground utility regulations, illustrating the continued use of linguistically grounded symbolic representations in such domains.

This trajectory illustrates that explicit linguistic structures and domain models remain useful in knowledge-intensive settings, where interpretability, controllability, and formal reasoning are important (Bakker et al., 2024; Benson et al., 2024).

Neuro-Symbolic and LLM-Based Approach

More recent approaches combine ontologies with

neural and LLM-based methods. Lamazzi et al. (2026) use an ontology to constrain LLM-based knowledge graph generation, while Isaradech et al. (2026) use an ontology to define the annotation scheme and entity categories for transformer-based NER. Earlier, Cimiano and Völker (2005) showed that ontologies can support open-domain entity classification without labeled training data, assigning entities to the most contextually similar concepts within a large, exchangeable ontology. These works illustrate complementary uses of ontologies in neural information extraction, from constraining outputs and structuring annotations to supporting entity classification. Recent biomedical system further emphasizes *modularity*: Ratas et al. (2026) introduce an architecture in which tokenization, NER, and entity linking can be independently replaced or extended. At the same time, LLM-based ontology learning can produce more complete structures but may struggle with consistent property modeling and introduce hallucinated relations (Bakker et al., 2024). Together, these works highlight the potential of combining neural methods with structured ontological constraints and modular architectures.

Our approach

Our framework (Fig. 1) aligns with the Linguistic-Symbolic World Model intuition but pivots architecturally to a highly precise task: **making the implicit explicit** through comprehensive, attribute-level knowledge extraction. For this purpose, we employ a deterministic, domain ontology-guided Named Entity Recognition (NER) layer as the primary structuring backbone (see 5), ensuring that extracted named entities correspond directly to formal ontology classes from the outset. This symbolic baseline provides an initial foundation for subsequent processing. Although alignment errors may propagate and affect the resolution of attribute-based implicit formulations (3.1), its deterministic nature facilitates human-in-the-loop correction. To balance paradigms, an optional, complementary neural NER layer can be integrated, but its outputs remain unconditionally bound by the ontological schema. Resolving explicit attributes to uncover unmentioned objects requires strict adherence to the text with no room for hallucination; thus, we prioritize **high precision over recall** to ensure suitability for formal reasoning.

3 GCS description characterization

3.1 Implicit Attribute-Level knowledge framing

A GCS description is a concise, manually authored technical paragraph that defines the scope, nature, and sequential steps of a grid connection project. Created by solution design experts for expert reviewers, it serves as a precise blueprint for evaluating connection solutions. We empirically identified four recurring characterizations of such attribute-level implicit formulations. While this set is not intended to be exhaustive, these four patterns emerged consistently in our data and will serve as the focus of the analysis presented in this paper.

Case 1 – Object referred to through a uniquely identifying attribute

The expert refers to an object through an attribute whose value alone is sufficient to identify the underlying object according to the ontology.

	Implicit formulation	Explicit formulation
FR	<i>Pose d'un <u>REMBT</u></i>	<i>Pose d'un [coffret] REMBT</i>
EN	<i><u>REMIC</u> installation</i>	<i>REMIC [box] installation</i>

Case 2 – Object inferred from a combination of attributes

The expert refers to an object through a set of attributes whose combination allows the associated object to be inferred.

	Implicit formulation	Explicit formulation
FR	<i>Pose de <u>20m</u> <u>BT</u> souterrain.</i>	<i>Pose de [cable] 20m BT souterrain.</i>
EN	<i>Installation of <u>20m</u> <u>underground LV</u>.</i>	<i>Installation of [cable] 20m underground LV.</i>

Case 3 – Implicit object duplication through multiple attribute values

An explicitly mentioned object is associated with multiple values corresponding to the same attribute type.

	Implicit formulation	Explicit formulation
FR	<i>Pose de tableaux <u>HTA</u> et <u>BT</u></i>	<i>Pose de tableaux HTA et [tableau(x)] BT</i>
EN	<i>Installation of <u>HV</u> and <u>LV</u> switchboards</i>	<i>Installation of HV switchboards and LV [switchboard(s)]</i>

Case 4 – Context-dependent attribute-based implicit formulation

The contextual information contained in the sentence allows the implicit object to be identified.

	Implicit formulation	Explicit formulation
FR	<i>Mutation de <u>160</u> à <u>240 kva</u>, au poste DU-RAND</i>	<i>Mutation de [transformateur] 160 à [transformateur] 240 kva, au poste DURAND</i>
EN	<i>Power upgrade from <u>160 to 240 kVA</u>, at DU-RAND substation</i>	<i>Power upgrade from [transformer] 160 to [transformer] 240 kVA, at DURAND substation</i>

3.2 Structuring objectives and target representation

The objective is therefore to identify all objects referenced in a GCS description, whether explicitly or implicitly formulated, and associate them with their corresponding attributes.

Formally, let the area of interest of a GCS description for objects extraction be denoted by :

$$G = (O_E, O_I) \quad (1)$$

where:

$$O_E = o_1^E, o_2^E, \dots, o_n^E \quad (2)$$

is the set of explicitly mentioned objects extracted directly from the textual description, and

$$O_I = o_1^I, o_2^I, \dots, o_m^I \quad (3)$$

is the set of implicitly inferred objects reconstructed through the inference process. Each explicit object o_i^E is associated with a set of explicit attributes:

$$A(o_i^E) = a_{i1}, a_{i2}, \dots, a_{ik} \quad (4)$$

where the attributes correspond exclusively to information explicitly present in the input text. Similarly, each inferred object o_j^I is associated with a set of orphan attributes explicitly present in text :

$$A(o_j^I) = a_{j1}, a_{j2}, \dots, a_{jl} \quad (5)$$

The complete object representation associated with a GCS description can therefore be defined as:

$$O = O_E \cup O_I \quad (6)$$

together with the attribute mapping function:

$$A : O \rightarrow \mathcal{P}(\mathcal{A}) \quad (7)$$

where $\mathcal{A}$ denotes the global attribute space and $\mathcal{P}(\mathcal{A})$ its power set.

The resolution mechanism is illustrated below.

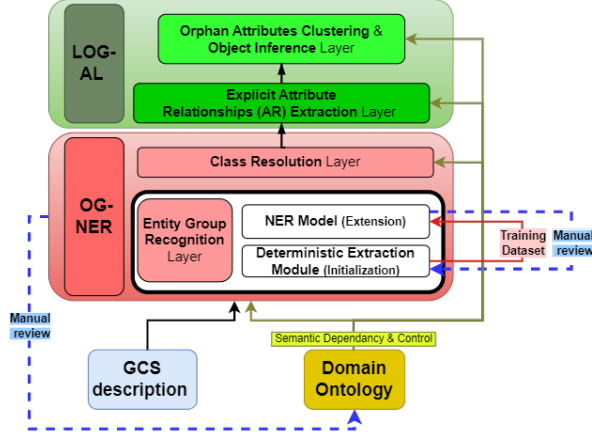


Figure 1: Domain Ontology-Based Framework for Attribute-Level Knowledge Extraction

4 Domain knowledge and structure

4.1 Ontology Design and Material

We formalized an ontology capturing the concepts and relationships required to represent the engineering knowledge expressed in GCS. A GCS can be viewed as a set of operational tasks applied to network equipment, optionally enriched with associated attributes. Accordingly, the initial ontology scope was defined to represent the operational tasks, network equipment, their attributes, and the relationships linking these elements. The initial ontology ($v0$) was manually developed in Protégé (Jain and Singh, 2013) by combining knowledge elicited from domain experts with the French electrical network standard *StaR-Elec*¹ and internal electrical network components catalogues. Although not the focus of this paper, the targeted use cases—primarily involving sub-network scheme and pattern detection—were used to determine the minimal set of concepts and relations required by the ontology, while domain expertise and industrial reference resources ensured semantic consistency and provided an initial level of domain coverage. As this work focuses on a proof of concept, the ontology was intentionally bounded to this initial scope. A more comprehensive assessment of ontology coverage and accuracy through formal expert review and organizational validation procedures would constitute a subsequent step toward industrial deployment.

¹<https://schema.data.gouv.fr/cnigfr/schema-reseaux-electriques/>

4.2 Domain Ontology Structure

Following standard OWL-RDF formalisms, our framework structures domain knowledge across three distinct entity types (Horridge et al., 2009) natively aligned with the Protégé environment :

- **Classes (C):** Structural abstractions mapping core categories such as operational tasks, network equipments, network hierarchy, specialized linguistic classes (Fig. 2).
- **Data Properties (D):** Attribute specifications that bind a class instance to a literal data primitive. Each property is strictly defined by a *Domain* (the applicable source classes) and a *Range* (the literal data type primitive, e.g., `xsd:string`, `xsd:integer`).
- **Object Properties (R):** Directed semantic relationships connecting two distinct class instances, structurally bounded by explicit source *Domain* and target *Range* constraints.

Each entity type encapsulates structural definitions, including class naming conventions, taxonomic heritage via `rdfs:subClassOf` paths, logical equivalence rules, and annotations.

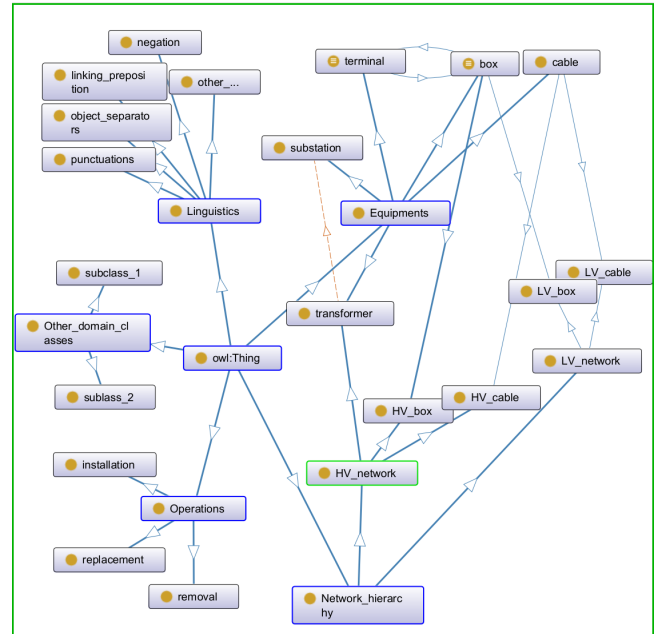


Figure 2: Ontology Class Hierarchy Backbone

Beyond native metadata, we inject domain-specific extraction parameters directly into the schema of classes and attributes of interest (referred to as *entity groups*; see Section 5) via custom

owl:AnnotationProperty hooks. These parameters are designed to programmatically configure our downstream natural language processing pipeline:

- **deterministic_regex**: Explicit regular expressions used for rule-based token capture. Entities carrying exclusively this annotation bypass probabilistic model inference entirely. They suit to Entities with low ambiguity and limited sensitivity to spelling variation, for which surface-form matching is sufficient.
- **hybrid_regex**: Regular expressions assigned to Entities, used to anchor and featurize candidate text spans during Named Entity Recognition (NER) model training. They suit to Entities for which contextual disambiguation can be needed. These cases typically exhibit lexical variability, incomplete pattern coverage, or spelling noise.
- **default_class**: A statistical prior mapping ambiguous or multi-valent attributes to their most frequent baseline class assignment.
- **disambiguation_required**: A boolean flag indicating that an entity was tagged at a higher functional-group level and must be routed to the downstream Class Resolution Layer (Fig. 1) for granular type assignment.

5 Ontology-Guided Named Entity Recognition

We introduce an domain Ontology-Guided Named Entity Recognition (OG-NER) module as the first structuring stage applied to raw text. Given an input sequence of tokens $T = (t_1, \dots, t_n)$, the goal is to produce a set of ontology-grounded entity mentions $\mathcal{E}$, where each mention is linked to an entity group derived from the ontology.

Formally, each extracted entity $e \in \mathcal{E}$ is defined as:

$$e = (k, w, i, j, g, c, s) \quad (8)$$

where:

- k denotes the entity index according to its order of appearance in the text,
- $w = T_{i:j}$ is the surface span,
- i and j denote the start and end character offsets in the text,

- $g \in \mathcal{G}$ corresponds to an ontology-defined *entity group*, i.e., a functional grouping of one or more leaf classes, having at least one of the annotations **deterministic_regex** or **hybrid_regex** in its description (see 4.2).
- $c \in \mathcal{C}$ corresponds to the ontology-defined entity class, produced by the OG-NER Class Resolution Layer, in case g 's **disambiguation_required** annotation value is set to True (see 4.2, Fig. 1).
- $s \in \mathcal{J}$ corresponds to the extraction source, allowing to track if the entity was captured by the deterministic module or the probabilistic model.

5.1 OG-NER architecture

The OG-NER pipeline (Fig. 1) is structured as a two-tiered hybrid system consisting of two complementary, conditionally combined extraction modules. Crucially, the system enforces a strict architectural priority where the deterministic approach governs the pipeline as the mandatory initializing layer:

5.1.1 Deterministic Initializer Module ($\mathcal{E}_{\text{det}}$):

relying on enriching ontology classes and attributes *entity groups* (see section 5), with sets of regular expressions (**deterministic_regex** or **hybrid_regex**) $\mathcal{R}_g$.

Each group $g \in \mathcal{G}$ is associated with one or more patterns:

$$\mathcal{R}_g = \{r_1, \dots, r_k\} \quad (9)$$

An entity is extracted if a span in the text matches at least one pattern:

$$(t_i, \dots, t_j) \models r \Rightarrow e = (T_{i:j}, i, j, g) \quad (10)$$

While this formulation reduces the search space of candidate entities by constraining extraction to ontology-relevant surface forms, it introduces an ambiguity resolution problem (see 5.1.4) at the level of leaf class assignment within each group.

Pragmatically, this module serves as the pipeline base layer; its high-precision extractions anchor the document structure and are subsequently leveraged to bootstrap and generate annotated training datasets for the neural layer.

5.1.2 Contextual Augmentation Model ($\mathcal{E}_{\text{ctx}}$):

To resolve linguistic ambiguities and coverage gaps escaping rigid rule matching, this layer leverages

Enebert, a pretrained transformer-based language model (Eunice et al., 2023). We frame named entity extraction as a token classification task utilizing the standard BIO (Begin, Inside, Outside) tagging scheme (Wang et al., 2022). To optimize parameter efficiency and preserve the pretrained representations, we freeze the base model weights and attach a Low-Rank Adaptation (LoRA) layer (Hu et al., 2022) for supervised fine-tuning. Crucially, the training dataset is programmatically bootstrapped by executing the deterministic initializer ($\mathcal{E}_{\text{det}}$) across a large GCS descriptions corpus, targeting text spans matching the designated `hybrid_regex` annotations.

5.1.3 $\mathcal{E}_{\text{det}}$ and $\mathcal{E}_{\text{ctx}}$ Merging

The final resolved entity set $\mathcal{E}$ is synthesized via a deterministic merging function:

$$\mathcal{E} = \text{Merge}(\mathcal{E}_{\text{det}}, \mathcal{E}_{\text{ctx}}) \quad (11)$$

where $\mathcal{E}_{\text{det}}$ and $\mathcal{E}_{\text{ctx}}$ denote the entity sets extracted by the deterministic and contextual components, respectively. To guarantee absolute compliance with core domain constraints, the merging logic applies a strict priority override: in the event of spatial or boundary conflicts between overlapping candidate spans, $\mathcal{E}_{\text{det}}$ implicitly supersedes $\mathcal{E}_{\text{ctx}}$, ensuring deterministic rules systematically override probabilistic predictions. This merging system enables a hybrid design where rule-based extraction provides high-precision anchors, while neural models improve recall in more ambiguous settings.

5.1.4 Class Resolution Layer

Given a mention w associated with an entity group g having a `disambiguation_required` annotation set to `True`, and a set of candidate leaf classes $\mathcal{C}_g = \{c_1, \dots, c_m\}$, this step performs class alignment in a string-edit space.

The final class c^* is selected via constrained Levenshtein optimization:

$$c^* = \arg \min_{c \in \mathcal{C}_g} \text{Lev}(\phi(w), c) \quad (12)$$

subject to the dual structural constraints:

$$\text{Prefix}(\phi(w), c^*) = \text{True} \quad (13)$$

$$1 - \frac{\text{Lev}(\phi(w), c^*)}{\max(|\phi(w)|, |c^*|)} \geq \tau \quad (14)$$

$\phi(\cdot)$ is a preprocessing function applied **only** to the extracted mention w to perform surface normalization (e.g., verb nominalization, acronym expansion) and reduce lexical variability.

To optimize precision over recall and eliminate false-positive misalignments, candidate matches must satisfy a dual-validation filter: (13) an initial character-prefix validation to enforce structural token alignment, and (14) a similarity threshold τ . If these constraints fail, rather than accepting a noisy raw match, the pipeline triggers a *human-in-the-loop correction loop*. This prompts a domain expert to manually correct errors by adding missing leaf classes to the ontology schema or refining the preprocessing rules within $\phi(\cdot)$.

6 Linguistic- and Ontology-Guided Attribute Linking (LOG-AL)

This layer transforms OG-NER outputs $\mathcal{E}$ into structured, ontology-compliant knowledge representations by first identifying explicit attribute–object relations and subsequently recovering implicit ones (Perfetti and Stafura, 2015).

We segment $\mathcal{E}$ into disjoint subsets: (i) attribute entities $\mathcal{A}$, and (ii) class entities $\mathcal{C}$. Based on the target ontology, $\mathcal{C}$ is further partitioned into actionable domain objects $\mathcal{O}$ capable of possessing attributes, and non-possessing classes $\mathcal{N}$ which represent lexical anchor entities or abstract concepts that do not accept attribute configurations ($\mathcal{C} = \mathcal{O} \cup \mathcal{N}$).

6.1 Explicit Object - Attribute Links extraction

Algorithm 1 Explicit Object–Attribute Relation Reconstruction

Input: Objects of interest $\mathcal{O}$, attribute entities $\mathcal{A}$

Output: `rel_cls_att`

`rel_cls_att` $\leftarrow \emptyset$

for each $att \in \mathcal{A}$ **do**

 mark att as unlinked

for each $obj \in \mathcal{O}$ **do**

for each $att \in \mathcal{A}$ **do**

if `LexicalCompat(obj, att)` and att is unlinked **then**

if `SemanticValid(obj, att)` **then**

 mark att as linked

 add $(obj, att, \text{hasAttribute}, \text{True})$

 to `rel_cls_att`

else

 mark att as unlinked

 add $(obj, att, \text{hasAttribute}, \text{False})$

 to `rel_cls_att`

end if

end if

end for

 add $(obj, att, \text{hasAttribute}, \text{False})$

 to `rel_cls_att`

end if

end for

end for

return `rel_cls_att`

Algorithm 1 formally outlines the sequential resolution of explicit object–attribute pairs $(obj, att) \in \mathcal{O} \times \mathcal{A}$. The framework evaluates candidate pairs through a two-stage validation filter:

6.1.1 Lexical and Spatial Compatibility (LexicalCompat):

Validates the raw textual span bounded between the *obj* and *att* mentions. A candidate relation is instantly rejected if the span contains an element from a predefined configuration set of *attribute separators* $\mathcal{S}$ (e.g., punctuation, explicit operations or equipments, cardinal attributes, ...), except in specific linguistic configurations where such elements do not constitute an actual separator. A uniqueness constraint guarantees that an *att* token can only map to a single *obj*, discarding already-linked attributes.

6.1.2 Ontological Semantic Validation (SemanticValid):

Verifies semantic domain compliance against the ontology. A relation is accepted if and only if the extracted *obj* represents a valid element within the structural intersection domain dictated by the ontology definitions of the existing attributes it is currently linked to.

Successfully validated links are captured as positive semantic triples $(obj, att, \text{hasAttribute}, \text{True})$, while failed or filtered candidate pairs are explicitly preserved as negative assertions $(obj, att, \text{hasAttribute}, \text{False})$ to guarantee pipeline tracking.

This step allows the isolation of *orphan attributes* ($\mathcal{A}_{\text{orphan}}$) to serve as inputs for subsequent resolution (Section 6.2), while identifying explicit objects fully described with their explicit attributes to distinguish them from complementary inferred objects and compute evaluation metrics (Section 7.2).

Algorithm 2 Derivation of Orphan Attributes

Input: rel_cls_att , attribute set $\mathcal{A}$
Output: orphan attribute set $\mathcal{A}_{\text{orphan}}$

```

 $\mathcal{A}_{\text{linked}} \leftarrow \emptyset$ 
for each  $(obj, att, \text{hasAttribute}, \text{True})$  in  $\text{rel\_cls\_att}$  do
    add  $att$  to  $\mathcal{A}_{\text{linked}}$ 
end for
 $\mathcal{A}_{\text{orphan}} \leftarrow \mathcal{A} \setminus \mathcal{A}_{\text{linked}}$ 
return  $\mathcal{A}_{\text{orphan}}$ 

```

6.2 Clustering Orphan Attributes and Implicit Object Inference

Orphan attributes are grouped into coherent semantic clusters, where each cluster represents an implicit object mention inferred from contextual and ontological cues. The intuition is that attributes appearing in close textual proximity and not separated by *attribute separators* (see 6.1.1) are likely to describe the same underlying object. To achieve this, we introduce Algorithm 3 combining *linguistic patterns* with *ontology-based constraints*.

Algorithm 3 Objects Inference from Orphan Attributes Clusters

Input: $\mathcal{A}_{\text{orphan}}$, $\mathcal{E}$, ontology, linguistic_patterns, attribute_separators
Output: inferred_objects

```

Sort  $\mathcal{A}_{\text{orphan}}$  by textual order
inferred_objects  $\leftarrow \emptyset$ 
current_att_cluster  $\leftarrow \emptyset$ 
current_obj  $\leftarrow \emptyset$ 
for each  $att \in \mathcal{A}_{\text{orphan}}$  do
    if current_att_cluster =  $\emptyset$  then
        current_att_cluster  $\leftarrow [att]$ 
        current_obj  $\leftarrow \text{infer\_implicit\_obj}(\text{current\_att\_cluster}, \mathcal{E}, \text{ontology}, \text{linguistic\_patterns})$ 
        continue
    end if
    prev  $\leftarrow \text{last}(\text{current\_att\_cluster})$ 
    interm  $\leftarrow \text{get\_intermediate\_entities}(\text{prev}, att)$ 
    groups  $\leftarrow \{x.\text{entity\_group} \mid x \in \text{current\_att\_cluster}\}$ 
    if  $\exists e \in \text{interm}$  such that
         $e \in \text{attribute\_separators}$ 
        and  $att.\text{entity\_group} \notin \text{groups}$  then
        tmp  $\leftarrow \text{current\_att\_cluster} \cup \{att\}$ 
        obj  $\leftarrow \text{infer\_implicit\_obj}(\text{tmp})$ 
        if obj  $\neq \emptyset$  then
            current_att_cluster  $\leftarrow \text{tmp}$ 
            current_obj  $\leftarrow \text{obj}$ 
        else
            add (current_att_cluster, current_obj)
            to inferred_objects
            current_att_cluster  $\leftarrow [att]$ 
            current_obj  $\leftarrow \text{infer\_implicit\_obj}(\text{current\_att\_cluster}, \mathcal{E}, \text{ontology}, \text{linguistic\_patterns})$ 
        end if
    else if
        add (current_att_cluster, current_obj)
        to inferred_objects
        current_att_cluster  $\leftarrow [att]$ 
        current_obj  $\leftarrow \text{infer\_implicit\_obj}(\text{current\_att\_cluster}, \mathcal{E}, \text{ontology}, \text{linguistic\_patterns})$ 
    end if
end for
if current_att_cluster  $\neq \emptyset$  then
    add (current_att_cluster, current_obj)
    to inferred_objects
end if
return inferred_objects

```

We define *linguistic patterns* (see A) as a set of contextual rule-based validation heuristics applied

over attribute sequences and local textual context:

$$\text{linguistic_patterns} = \{P_1, P_2, \dots, P_n\} \quad (15)$$

Each pattern captures a linguistic mechanism for resolving implicit object references where ontology constraints alone are insufficient or ambiguous.

We define *ontology-based constraints* as a set of logical consistency rules over the ontology schema, ensuring that an orphan attribute cluster is considered valid only if there exists at least one class whose domain satisfies a consistent intersection with all attributes in the cluster.

The algorithm sequentially processes text-ordered orphan attributes ($\mathcal{A}_{\text{orphan}}$) to programmatically rebuild latent object associations. It incrementally extends a running attribute cluster with a candidate attribute if: (i) the intervening textual span is devoid of any designated *attribute separators*, and (ii) the candidate introduces a new semantic attribute entity group not yet present in the current cluster.

Upon passing these structural filters, the `infer_implicit_obj` routine applies ontology constraints and linguistic patterns to determine whether the updated cluster is semantically consistent. If valid, the attribute candidate is integrated and the target object is updated; otherwise, the running cluster is saved into the `inferred_objects` list and a new cluster is initialized.

7 Results

Experimental setting. The evaluation is conducted on the GCS Archive, a real-world collection of 6,723 paragraphs spanning 2023–2025. The archive is used for corpus-level analyses, including recall-oriented assessment and the characterization of implicit and explicit object references. Precision and F_1 -score are computed on a stratified GCS Evaluation Sample of 100 manually annotated paragraphs, selected to ensure balanced coverage of highly explicit, highly implicit, and mixed cases based on explicit/implicit object formulation ratios.

The ontology comprises 5,404 axioms (3,481 logical), 564 classes, 55 object properties, and 286 data properties, resulting in a rich semantic structure that significantly increases the difficulty of both explicit and implicit extraction.

Key insight: measuring implicit object formulations. GCS descriptions do not always explicitly list all operational objects, as domain expertise often relies on implicit formulations. The proposed

pipeline provides a scope-bounded operationalization of such implicit references, enabling their identification and quantification within a constrained context. This motivates evaluation of NER coverage and, more importantly, implicit object recovery.

7.1 OG-NER performance

We evaluate the OG-NER layer by comparing $\mathcal{E}_{\text{det}}$, $\mathcal{E}_{\text{ctx}}$, and $\mathcal{E}$ (5.1) extraction strategies on the GCS Evaluation Sample. For each extraction strategy, Precision, recall, and F_1 are computed at the entity level (mainly based on i, j, g , following) with the gold annotations serving as the reference standard (see B). The reported support corresponds to the total number of gold entity instances in the evaluation sample ($N = 3,444$). Specifically,

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN},$$

and

$$F_1 = \frac{2 \text{ Precision Recall}}{\text{Precision} + \text{Recall}}.$$

Approach	Recall	Precision	F_1
$\mathcal{E}_{\text{det}}$	0.9518	0.9753	0.9634
$\mathcal{E}_{\text{ctx}}$	0.4202	0.7501	0.5386
$\mathcal{E}$	0.9832	0.9511	0.9669

Table 1: OG-NER performance on the GCS Evaluation Sample

$\mathcal{E}_{\text{ctx}}$ approach alone achieves lower performance, which is expected as some entity classes are not intended to be identified through contextual inference. The hybrid configuration $\mathcal{E}$ improves recall by combining rule-based and contextual signals, but at the cost of a slight reduction in precision. The choice between the $\mathcal{E}_{\text{det}}$ and $\mathcal{E}$ approaches therefore depends on application requirements and available computational resources. Future improvements may come from refining the regular expressions set and/or retraining the model to better complement deterministic recognition.

7.2 LOG-AL implicit object recovery

We run inference with $\mathcal{E}_{\text{det}}$ on the GCS Archive augmented with LOG-AL outputs. The resulting extractions are injected as GCS nodes in a Neo4j graph database, where Cypher queries compute the implicit object recovery metrics (see C).

Metric	Value
Implicit object ratio	27.17%
GCS with implicit objects	72.6%
Avg. implicit objects per GCS	28.1%

Table 2: GCS Archive Implicit Objects characterization

Results suggest that 27.17% of GCS operational objects are identified as implicit, while 72.6% of GCS instances contain at least one implicit object. The close alignment between the implicit object ratio and the average per-GCS implicit proportion (28.1%) suggests that implicit objects are consistently distributed across GCS instances rather than concentrated in a subset of cases. Evaluation on the GCS sample showed no ontology-scheme violations in object-attribute linking; however, this does not by itself establish the correctness of attribute linking. The remaining errors primarily concern missed objects, mainly due to limitations in NER coverage, including undefined ontology concepts or rules, as well as unsupported linguistic phenomena. Given the richness of the underlying ontology, extending the evaluation to these cases is therefore required and will be addressed in the transition toward concrete use cases and large-scale knowledge graph construction, where these limitations can be more systematically assessed and handled.

Summary. The analysis reveals a substantial presence of implicit operational objects in GCS descriptions, supporting the need for inference-aware extraction methods.

8 Conclusion and Future Work

This work introduced an ontology-driven framework for fine-grained extraction of explicit and implicit operational attribute-level knowledge from technical texts. Applied to French electrical Grid Connection Solutions (GCS), the approach enables the detection of attribute-based implicit object formulations and the recovery of operational objects not explicitly stated in the text. The proposed framework provides a structured representation of domain knowledge that can support downstream knowledge graph construction. Future work will focus on large-scale knowledge graph construction and the development of decision-support tools for domain experts.

9 Limitations

While the domain-ontology-driven framework offers high explainability and supports precise error correction, its modular architecture allows individual components to be adapted or reused according to the requirements of other sources and application settings. The framework does, however, rely on structured domain resources, and its deployment in a new setting may therefore involve additional engineering effort, particularly for ontology construction and adaptation of the NER component to domain-specific terminology and linguistic phenomena. The modularity of the framework is intended to facilitate such adaptations by allowing these components to be developed or refined independently, rather than requiring the entire extraction pipeline to be redesigned. Evaluating this transferability across other technical domains and levels of resource availability is therefore an important direction for assessing the general applicability of the approach.

10 Ethical Considerations

This framework relies on a human-centered design in which the domain ontology and regular expression assets are developed and maintained through expert input. These components are integrated into proprietary infrastructure. This design reflects the role of human expertise in NLP systems, particularly in industrial settings where explainability, controllability, and structural reliability are important. Expert oversight enables traceability and supports controlled intervention in downstream processing.

References

- Roos M Bakker, Daan L Di Scala, and Maaïke HT de Boer. 2024. [Ontology learning from text: an analysis on llm performance](#). In *NLP4KGC@ SEMANTiCS*, pages 70–87.
- Carter Benson, Alec Sculley, Austin Liebers, and John Beverley. 2024. My ontologist: evaluating bfo-based ai for definition support. *arXiv preprint arXiv:2407.17657*.
- Dianne C. Berry. 1987. [The problem of implicit knowledge](#). *Expert Systems*, 4(3):144–151.
- Marc C. and Pierre Z. 1992. [Extracting implicit information from free text technical reports](#). *Information Processing & Management*, 28(5):609–618.
- Philipp Cimiano and Johanna Völker. 2005. Towards large-scale, open-domain and ontology-based named

- entity classification. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*.
- Dominique Estival, Chris Nowak, and Andrew Zschorn. 2004. Towards ontology-based natural language processing. In *Proceedings of the Workshop on NLP and XML (NLPXML-2004): RDF/RDFS and OWL in Language Technology*, pages 59–66.
- Akani Eunice, Gemignani Romain, and Abrougui Rim. 2023. Enebert: a state-of-the-art language model trained on a corpus of texts generated from the set of dso activities. In *IET Conference Proceedings CP823*, volume 2023, pages 2903–2907. IET.
- Gökhan Giray and M. Ozan Ünalır. 2020. [Assessment of text coherence using an ontology-based relatedness measurement method](#). *Expert Systems*, 37(6):e12505.
- Thomas R. Gruber. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.
- Matthew Horridge, Simon Jupp, Georgina Moulton, Alan Rector, Robert Stevens, and Chris Wroe. 2009. A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1. 2. *The university of Manchester*, 107.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.
- Natthanaphop Isaradech, Wachiranun Sirikul, Stefan Schulz, and Markus Kreuzthaler. 2026. Ontology-based medication named entity recognition using pretrained transformer models from a thai hospital: Model fine-tuning and validation study. *JMIR Formative Research*, 10(1):e82685.
- Vishal Jain and Mayank Singh. 2013. Ontology development and query retrieval using protégé tool. *International Journal of Intelligent Systems and Applications*, 9(9):67–75.
- Ewan Klein and Stephen Potter. 2004. An ontology for nlp services. In *The Workshop Programme*, volume 10, page 3.
- Lorenzo Lamazzi, Aldo Gangemi, Alessio Giberti, Andrea Giovanni Nuzzolese, Vittorio Andrea Rocca, Mattia Torta, and Francesco Poggi. 2026. [Tacit knowledge extraction via logic augmented generation and active inference](#). *arXiv preprint arXiv:2605.07639*.
- Kavi Mahesh, Sergei Nirenburg, and 1 others. 1995. ated ontology for practical nlp. In *Proceedings of the IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, volume 19, page 21.
- Charles A Perfetti and Joseph Z Stafura. 2015. Comprehending implicit meanings in text without making inferences.
- Mart Ratas, Thomas Searle, Adam Sutton, and Richard Dobson. 2026. Medcat v2: a modular, extensible architecture for clinical named entity recognition and linking under real-world privacy and compute constraints. In *BioNLP 2026*, pages 191–198.
- Yu Wang, Hanghang Tong, Ziyi Zhu, and Yun Li. 2022. Nested named entity recognition: a survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(6):18–19.
- Xin Xu and Hubo Cai. 2021. Ontology and rule-based natural language processing approach for interpreting textual regulations on underground utility infrastructure. *Advanced Engineering Informatics*, 48:101288.
- Ugan Yasavur, Reza Amini, Christine L Lisetti, and Naphtali Rische. 2013. Ontology-based named entity recognizer for behavioral health. In *FLAIRS*.

A Linguistic Patterns Definition

This appendix provides two illustrative examples of linguistic patterns used in Algorithm 3. These patterns are not intended to be exhaustive or formally complete rules, but rather representative heuristics that capture recurring regularities observed in GCS descriptions. In practice, their interpretation and resolution may be handled either by a model-based component, preprocessing strategies, or deterministic rule-based enforcement, depending on the system design. The notion of linguistic patterns here should therefore be understood broadly, including phenomena such as local coreference, attribute repetition, and context-dependent attribute disambiguation.

A.1 Attribute Repetition Pattern P_1 :

Pattern P_1 deals with attribute repetition across consecutive mentions and is applied in Case 3 implicit formulations resolution.

$$\text{attr}(a_i) = \text{attr}(a_{i+1}) \Rightarrow \text{obj}(a_{i+1}) = \text{obj}(a_i) \quad (16)$$

According to ontology axioms, several electrical objects may be associated with a voltage level (LV or HV). However, in Case 3, P_1 resolves the ambiguity by leveraging repetition across consecutive attribute mentions, thereby recovering the implicit object structure.

A.2 Attribute–Operation Constraint Pattern P_2 :

Pattern P_2 deals with attribute–operation contextual disambiguation. The following rule is applied to explain Case 4 example :

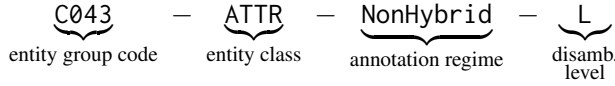
$$\begin{aligned}
&\text{orphan}(c) \wedge \text{attr}(c) = \text{power} \\
&\wedge \text{operation}(c) = \text{mutation} \\
&\Rightarrow \text{obj}(c) = \text{Transformer}
\end{aligned} \tag{17}$$

Although multiple electrical objects may be associated with power, the operation *mutation* constrains the interpretation to *Transformer*, enabling the linguistic pattern to resolve the ambiguity.

B OG-NER Inference evaluation

The entity groups reported in this appendix follow the notation defined as g in section 5. Each identifier (Entity group ID) encodes four properties: the entity class (OBJ or ATTR), the annotation regime (Hybrid or NonHybrid), and the entity level (L or G). Hybrid indicates that the entity group is included in the NER-based extraction mode, whereas NonHybrid indicates that it is handled exclusively by the deterministic extraction component. The L/G distinction refers to the presence of the `disambiguation_required` annotation: L denotes a leaf entity, for which disambiguation is not required, while G denotes a group entity, for which disambiguation is required.

Thus, an identifier such as C043_ATTR_NonHybrid_L can be decomposed as:



The identifier therefore directly encodes the entity class, extraction regime, and disambiguation level.

B.1 Deterministic $\mathcal{E}_{\text{det}}$ inference results overview

Measure	Value
Total entity occurrences	3,361
Distinct entity groups	98
Distinct Hybrid leaf objects	14
Distinct Non-hybrid leaf objects	17
Distinct Hybrid group objects	5
Distinct Non-hybrid group objects	0
Distinct Hybrid leaf attributes	22
Distinct Non-hybrid leaf attributes	24
Distinct Hybrid group attributes	11
Distinct Non-hybrid group attributes	5

Table 3: $\mathcal{E}_{\text{det}}$ entity level extraction statistics

Entity group ID	Actual matches	Distinct text spans
C005_OBJ_Hybrid_L	1	1
C031_OBJ_Hybrid_L	13	2
C043_ATTR_NonHybrid_L	3	2
C044_ATTR_NonHybrid_L	1	1
C045_ATTR_NonHybrid_L	1	1
C048_ATTR_NonHybrid_L	2	2
C048_OBJ_Hybrid_L	3	3
C049_ATTR_Hybrid_L	5	3
C066_ATTR_NonHybrid_L	7	6
C068_OBJ_Hybrid_L	1	1
C074_OBJ_Hybrid_L	3	2
C075_OBJ_Hybrid_L	6	2
C112_ATTR_NonHybrid_L	2	2
C114_OBJ_Hybrid_L	4	1
C118_ATTR_NonHybrid_L	2	1
C123_OBJ_NonHybrid_L	47	4
C124_ATTR_Hybrid_L	15	4
C125_ATTR_Hybrid_L	40	11
C126_ATTR_Hybrid_L	15	3
C127_OBJ_Hybrid_L	17	4
C128_OBJ_Hybrid_L	4	1
C129_ATTR_Hybrid_L	35	11
C132_ATTR_NonHybrid_L	3	1
C134_ATTR_Hybrid_L	28	23
C135_ATTR_NonHybrid_L	4	4
C136_ATTR_Hybrid_L	2	2
C137_ATTR_Hybrid_L	50	40
C141_ATTR_NonHybrid_L	1	1
C151_OBJ_Hybrid_L	57	5
C152_ATTR_Hybrid_L	1	1
C154_ATTR_NonHybrid_L	2	1
C155_ATTR_Hybrid_L	1	1
C157_ATTR_NonHybrid_G	3	1
C158_ATTR_Hybrid_G	26	2
C163_ATTR_Hybrid_G	123	7
C169_OBJ_Hybrid_L	1	1
C178_ATTR_NonHybrid_L	19	4
C181_OBJ_Hybrid_L	1	1
C187_OBJ_NonHybrid_L	8	3
C194_ATTR_NonHybrid_L	2	2
C195_ATTR_Hybrid_L	135	96
C197_ATTR_Hybrid_L	1	1
C199_ATTR_Hybrid_L	79	4
C200_ATTR_NonHybrid_G	4	2
C204_ATTR_Hybrid_G	11	2
C212_ATTR_NonHybrid_L	34	30
C213_ATTR_NonHybrid_L	1	1
C215_ATTR_NonHybrid_L	68	51
C216_ATTR_Hybrid_L	25	22
C219_ATTR_NonHybrid_L	16	3
C220_ATTR_Hybrid_L	5	2
C227_ATTR_Hybrid_L	4	3
C230_ATTR_NonHybrid_L	12	7
C232_ATTR_Hybrid_L	108	10
C236_OBJ_NonHybrid_L	2	2
C240_ATTR_Hybrid_L	58	35
C248_ATTR_NonHybrid_L	1	1
C252_ATTR_Hybrid_L	127	10
C254_ATTR_NonHybrid_L	6	5
C256_ATTR_NonHybrid_L	3	1
C257_ATTR_Hybrid_G	47	6
C258_ATTR_NonHybrid_G	1	1
C258_OBJ_NonHybrid_L	35	2
C263_ATTR_Hybrid_G	6	4
C268_ATTR_Hybrid_L	42	6
C270_ATTR_NonHybrid_L	5	3
C274_ATTR_Hybrid_G	16	1
C275_ATTR_NonHybrid_G	5	1
C277_ATTR_NonHybrid_G	13	1
C278_ATTR_Hybrid_G	104	11
C281_ATTR_Hybrid_G	1	1
C283_ATTR_NonHybrid_L	14	5
C287_ATTR_NonHybrid_L	4	2
C289_ATTR_Hybrid_G	25	7
C290_ATTR_Hybrid_L	1	1
C291_ATTR_Hybrid_L	3	1
C292_ATTR_Hybrid_G	78	11
C294_ATTR_Hybrid_G	9	2
C309_OBJ_NonHybrid_L	3	3
C329_OBJ_NonHybrid_L	30	1
C348_OBJ_NonHybrid_L	1	1
C357_OBJ_Hybrid_G	2	1
C375_OBJ_NonHybrid_L	6	2
C381_OBJ_Hybrid_L	1	1
C402_OBJ_NonHybrid_L	13	1
C406_OBJ_NonHybrid_L	19	8
C408_OBJ_Hybrid_G	113	4
C412_OBJ_Hybrid_G	381	67
C433_OBJ_Hybrid_L	271	17
C457_OBJ_NonHybrid_L	28	1
C458_OBJ_NonHybrid_L	88	7
C459_OBJ_NonHybrid_L	69	8
C496_OBJ_NonHybrid_L	6	4
C520_OBJ_NonHybrid_L	45	3
C521_OBJ_NonHybrid_L	6	3
C540_OBJ_NonHybrid_L	112	15
C556_OBJ_Hybrid_G	410	71
C559_OBJ_Hybrid_G	89	26

Table 4: $\mathcal{E}_{\text{det}}$ detailed inference results by entity group

The *Actual matches* column reports the total number of extracted entity occurrences, while *Distinct text spans* reports the number of distinct normalized surface forms associated with each entity group.

B.2 Contextual $\mathcal{E}_{\text{ctx}}$ inference results overview

Measure	Value
Total entity occurrences	1,929
Distinct entity groups	54
Distinct Hybrid leaf objects	13
Distinct Non-hybrid leaf objects	7
Distinct Hybrid group objects	6
Distinct Non-hybrid group objects	0
Distinct Hybrid leaf attributes	17
Distinct Non-hybrid leaf attributes	2
Distinct Hybrid group attributes	8
Distinct Non-hybrid group attributes	1

Table 5: $\mathcal{E}_{\text{ctx}}$ entity level extraction statistics

Entity group ID	Actual matches	Distinct text spans
C005_OBJ_Hybrid_L	1	1
C006_OBJ_Hybrid_L	5	4
C031_OBJ_Hybrid_L	13	4
C047_ATTR_Hybrid_L	16	14
C048_OBJ_Hybrid_L	2	2
C074_OBJ_Hybrid_L	3	2
C075_OBJ_Hybrid_L	10	4
C096_OBJ_NonHybrid_L	6	5
C124_ATTR_Hybrid_L	18	7
C125_ATTR_Hybrid_L	43	18
C126_ATTR_Hybrid_L	14	4
C127_OBJ_Hybrid_L	18	6
C129_ATTR_Hybrid_L	23	7
C136_ATTR_Hybrid_L	3	3
C137_ATTR_Hybrid_L	50	41
C139_ATTR_Hybrid_L	3	2
C140_ATTR_NonHybrid_L	1	1
C141_ATTR_NonHybrid_L	1	1
C152_ATTR_Hybrid_L	1	1
C155_ATTR_Hybrid_L	5	5
C163_ATTR_Hybrid_G	150	12
C178_OBJ_NonHybrid_L	12	2
C195_ATTR_Hybrid_L	115	88
C220_ATTR_Hybrid_L	3	1
C228_ATTR_Hybrid_L	7	5
C232_ATTR_Hybrid_L	51	14
C239_OBJ_NonHybrid_L	2	1
C240_ATTR_Hybrid_L	61	36
C252_ATTR_Hybrid_L	130	58
C257_ATTR_Hybrid_G	63	17
C263_ATTR_Hybrid_G	9	5
C268_ATTR_Hybrid_L	1	1
C268_OBJ_Hybrid_G	4	2
C274_ATTR_Hybrid_G	18	8
C275_ATTR_NonHybrid_G	5	1
C278_ATTR_Hybrid_G	6	5
C282_ATTR_Hybrid_G	1	1
C289_ATTR_Hybrid_G	27	10
C292_ATTR_Hybrid_G	33	10
C319_OBJ_Hybrid_L	7	4
C333_OBJ_Hybrid_L	2	1
C354_OBJ_Hybrid_L	25	16
C357_OBJ_Hybrid_G	1	1
C381_OBJ_Hybrid_L	4	3
C408_OBJ_Hybrid_G	110	6
C412_OBJ_Hybrid_G	334	66
C431_OBJ_Hybrid_L	1	1

Table 6: $\mathcal{E}_{\text{ctx}}$ detailed inference results by entity group (continued in Table 7).

Entity group ID	Actual matches	Distinct text spans
C448_OBJ_NonHybrid_L	42	29
C458_OBJ_NonHybrid_L	31	5
C483_OBJ_Hybrid_L	3	3
C520_OBJ_NonHybrid_L	22	4
C528_OBJ_NonHybrid_L	3	1
C556_OBJ_Hybrid_G	377	74
C559_OBJ_Hybrid_G	33	18

Table 7: $\mathcal{E}_{\text{ctx}}$ detailed inference results by entity group

The model-inferred entities include 10 NonHybrid entity groups, reflecting a typical version misalignment between the current ontology and the ontology configuration used for NER model training. Such discrepancies arise when an entity class originally handled by the NER model is subsequently reclassified as deterministic, or when an entity class is renamed while the previously trained model retains the original label.

B.3 Merged $\mathcal{E}$ inference results overview

Measure	Value
Total entity occurrences	3,560
Distinct entity groups	107
Distinct Hybrid leaf objects	18
Distinct Non-hybrid leaf objects	18
Distinct Hybrid group objects	5
Distinct Non-hybrid group objects	0
Distinct Hybrid leaf attributes	24
Distinct Non-hybrid leaf attributes	25
Distinct Hybrid group attributes	12
Distinct Non-hybrid group attributes	5

Table 8: $\mathcal{E}$ entity level extraction statistics

Entity group ID	Actual matches	Distinct text spans
C005_OBJ_Hybrid_L	1	1
C006_OBJ_Hybrid_L	5	4
C031_OBJ_Hybrid_L	15	4
C043_ATTR_NonHybrid_L	3	2
C044_ATTR_NonHybrid_L	1	1
C045_ATTR_NonHybrid_L	1	1
C047_ATTR_Hybrid_L	4	4
C048_ATTR_NonHybrid_L	2	2
C048_OBJ_Hybrid_L	3	3
C049_ATTR_Hybrid_L	5	3
C066_ATTR_NonHybrid_L	7	6
C068_OBJ_Hybrid_L	1	1
C074_OBJ_Hybrid_L	3	2
C075_OBJ_Hybrid_L	10	4
C112_ATTR_NonHybrid_L	2	2
C114_OBJ_Hybrid_L	4	1
C118_ATTR_NonHybrid_L	2	1
C123_OBJ_NonHybrid_L	47	4
C124_ATTR_Hybrid_L	18	6
C125_ATTR_Hybrid_L	55	16
C126_ATTR_Hybrid_L	16	4
C127_OBJ_Hybrid_L	19	5

Table 9: $\mathcal{E}$ detailed inference results by entity group (continued in Table 10).

Entity group ID	Actual matches	Distinct text spans
C128_OBJ_Hybrid_L	4	1
C129_ATTR_Hybrid_L	36	11
C132_ATTR_NonHybrid_L	3	1
C134_ATTR_Hybrid_L	28	23
C135_ATTR_NonHybrid_L	4	4
C136_ATTR_Hybrid_L	3	3
C137_ATTR_Hybrid_L	50	40
C139_ATTR_Hybrid_L	3	2
C140_ATTR_NonHybrid_L	1	1
C141_ATTR_NonHybrid_L	1	1
C151_OBJ_Hybrid_L	57	5
C152_ATTR_Hybrid_L	2	2
C154_ATTR_NonHybrid_L	2	1
C155_ATTR_Hybrid_L	3	3
C157_ATTR_NonHybrid_G	3	1
C158_ATTR_Hybrid_G	26	2
C163_ATTR_Hybrid_G	124	8
C169_OBJ_Hybrid_L	1	1
C178_ATTR_NonHybrid_L	19	4
C181_OBJ_Hybrid_L	1	1
C187_OBJ_NonHybrid_L	8	3
C194_ATTR_NonHybrid_L	2	2
C195_ATTR_Hybrid_L	138	99
C197_ATTR_Hybrid_L	1	1
C199_ATTR_Hybrid_L	79	4
C200_ATTR_NonHybrid_G	4	2
C204_ATTR_Hybrid_G	11	2
C212_ATTR_NonHybrid_L	34	30
C213_ATTR_NonHybrid_L	1	1
C215_ATTR_NonHybrid_L	68	51
C216_ATTR_Hybrid_L	25	22
C219_ATTR_NonHybrid_L	16	3
C220_ATTR_Hybrid_L	5	2
C227_ATTR_Hybrid_L	4	3
C230_ATTR_NonHybrid_L	12	7
C232_ATTR_Hybrid_L	113	12
C236_OBJ_NonHybrid_L	2	2
C240_ATTR_Hybrid_L	58	35
C248_ATTR_NonHybrid_L	1	1
C252_ATTR_Hybrid_L	143	19
C254_ATTR_NonHybrid_L	6	5
C256_ATTR_NonHybrid_L	3	1
C257_ATTR_Hybrid_G	60	14
C258_ATTR_NonHybrid_G	1	1
C258_OBJ_NonHybrid_L	35	2
C263_ATTR_Hybrid_G	9	6
C268_ATTR_Hybrid_L	42	6
C270_ATTR_NonHybrid_L	5	3
C274_ATTR_Hybrid_G	17	2
C275_ATTR_NonHybrid_G	5	1
C277_ATTR_NonHybrid_G	13	1
C278_ATTR_Hybrid_G	107	14
C281_ATTR_Hybrid_G	1	1
C282_ATTR_Hybrid_G	1	1
C283_ATTR_NonHybrid_L	14	5
C287_ATTR_NonHybrid_L	4	2
C289_ATTR_Hybrid_G	26	8
C290_ATTR_Hybrid_L	1	1
C291_ATTR_Hybrid_L	3	1
C292_ATTR_Hybrid_G	79	12
C294_ATTR_Hybrid_G	9	2
C309_OBJ_NonHybrid_L	3	3
C319_OBJ_Hybrid_L	7	4
C329_OBJ_NonHybrid_L	30	1
C348_OBJ_NonHybrid_L	1	1
C354_OBJ_Hybrid_L	25	16
C357_OBJ_Hybrid_G	3	2
C375_OBJ_NonHybrid_L	6	2
C381_OBJ_Hybrid_L	4	3
C402_OBJ_NonHybrid_L	13	1
C406_OBJ_NonHybrid_L	19	8
C408_OBJ_Hybrid_G	114	4
C412_OBJ_Hybrid_G	409	80
C433_OBJ_Hybrid_L	271	17
C448_OBJ_NonHybrid_L	15	6
C457_OBJ_NonHybrid_L	28	1
C458_OBJ_NonHybrid_L	89	8
C459_OBJ_NonHybrid_L	69	8
C483_OBJ_Hybrid_L	1	1
C496_OBJ_NonHybrid_L	6	4
C520_OBJ_NonHybrid_L	45	3
C521_OBJ_NonHybrid_L	6	3
C540_OBJ_NonHybrid_L	112	15
C556_OBJ_Hybrid_G	433	86
C559_OBJ_Hybrid_G	90	27

Table 10: $\mathcal{E}$ detailed inference results by entity group

B.4 Golden GCS Evaluation Sample overview

Measure	Value
Total entity occurrences	3,444
Distinct entity groups	108
Distinct Hybrid leaf objects	16
Distinct Non-hybrid leaf objects	18
Distinct Hybrid group objects	5
Distinct Non-hybrid group objects	0
Distinct Hybrid leaf attributes	24
Distinct Non-hybrid leaf attributes	25
Distinct Hybrid group attributes	12
Distinct Non-hybrid group attributes	5
Distinct NEW entity groups	3

Table 11: Golden GCS Evaluation Sample entity level statistics

Entity group ID	Actual matches	Distinct text spans
C005_OBJ_Hybrid_L	1	1
C031_ATTR_NonHybrid_L	1	1
C031_OBJ_Hybrid_L	15	5
C043_ATTR_NonHybrid_L	3	2
C044_ATTR_NonHybrid_L	1	1
C045_ATTR_NonHybrid_L	1	1
C047_ATTR_Hybrid_L	4	4
C048_ATTR_NonHybrid_L	1	1
C048_OBJ_Hybrid_L	3	3
C049_ATTR_Hybrid_L	5	3
C066_ATTR_NonHybrid_L	7	6
C068_OBJ_Hybrid_L	1	1
C074_OBJ_Hybrid_L	3	2
C075_OBJ_Hybrid_L	7	3
C112_ATTR_NonHybrid_L	2	2
C114_OBJ_Hybrid_L	4	1
C118_ATTR_NonHybrid_L	2	1
C123_OBJ_NonHybrid_L	47	4
C124_ATTR_Hybrid_L	18	6
C125_ATTR_Hybrid_L	54	15
C126_ATTR_Hybrid_L	15	3
C127_OBJ_Hybrid_L	19	5
C128_OBJ_Hybrid_L	4	1
C129_ATTR_Hybrid_L	36	11
C132_ATTR_NonHybrid_L	3	1
C134_ATTR_Hybrid_L	28	23
C135_ATTR_NonHybrid_L	4	4
C136_ATTR_Hybrid_L	3	3
C137_ATTR_Hybrid_L	50	40
C139_ATTR_Hybrid_L	1	1
C141_ATTR_NonHybrid_L	1	1
C151_OBJ_Hybrid_L	56	5
C152_ATTR_Hybrid_L	1	1
C154_ATTR_NonHybrid_L	2	1
C155_ATTR_Hybrid_L	3	3
C157_ATTR_NonHybrid_G	3	1
C158_ATTR_Hybrid_G	26	2
C163_ATTR_Hybrid_G	126	7
C169_OBJ_Hybrid_L	1	1
C178_ATTR_NonHybrid_L	19	4
C181_OBJ_Hybrid_L	1	1
C187_OBJ_NonHybrid_L	8	3
C194_ATTR_NonHybrid_L	2	2
C195_ATTR_Hybrid_L	130	93
C197_ATTR_Hybrid_L	1	1
C199_ATTR_Hybrid_L	77	3
C200_ATTR_NonHybrid_G	4	2
C204_ATTR_Hybrid_G	11	2
C212_ATTR_NonHybrid_L	37	32
C213_ATTR_NonHybrid_L	1	1
C215_ATTR_NonHybrid_L	64	51
C216_ATTR_Hybrid_L	26	23
C219_ATTR_NonHybrid_L	16	3
C220_ATTR_Hybrid_L	4	1
C227_ATTR_Hybrid_L	4	3

Table 12: Golden GCS Evaluation Sample detailed inference results by entity group (continued in Table 13).

Entity group ID	Actual matches	Distinct text spans
C230_ATTR_NonHybrid_L	12	7
C232_ATTR_Hybrid_L	112	11
C236_OBJ_NonHybrid_L	2	2
C240_ATTR_Hybrid_L	57	33
C248_ATTR_NonHybrid_L	1	1
C252_ATTR_Hybrid_L	137	16
C254_ATTR_NonHybrid_L	6	5
C256_ATTR_NonHybrid_L	3	1
C257_ATTR_Hybrid_G	50	9
C258_ATTR_NonHybrid_G	1	1
C258_OBJ_NonHybrid_L	35	2
C263_ATTR_Hybrid_G	9	6
C268_ATTR_Hybrid_L	42	6
C270_ATTR_NonHybrid_L	5	3
C274_ATTR_Hybrid_G	17	2
C275_ATTR_NonHybrid_G	5	1
C277_ATTR_NonHybrid_G	13	1
C278_ATTR_Hybrid_G	104	11
C281_ATTR_Hybrid_G	1	1
C282_ATTR_Hybrid_G	1	1
C283_ATTR_NonHybrid_L	15	6
C287_ATTR_NonHybrid_L	4	2
C289_ATTR_Hybrid_G	26	7
C290_ATTR_Hybrid_L	1	1
C291_ATTR_Hybrid_L	3	1
C292_ATTR_Hybrid_G	79	10
C294_ATTR_Hybrid_G	9	2
C309_OBJ_NonHybrid_L	3	3
C319_OBJ_Hybrid_L	6	4
C329_OBJ_NonHybrid_L	30	1
C348_OBJ_NonHybrid_L	1	1
C354_OBJ_Hybrid_L	18	10
C357_OBJ_Hybrid_G	2	1
C375_OBJ_NonHybrid_L	6	2
C381_OBJ_Hybrid_L	3	3
C402_OBJ_NonHybrid_L	13	1
C406_OBJ_NonHybrid_L	18	8
C408_OBJ_Hybrid_G	112	4
C412_OBJ_Hybrid_G	389	76
C433_OBJ_Hybrid_L	261	17
C448_OBJ_NonHybrid_L	2	2
C457_OBJ_NonHybrid_L	28	1
C458_OBJ_NonHybrid_L	85	7
C459_OBJ_NonHybrid_L	61	8
C496_OBJ_NonHybrid_L	8	5
C520_OBJ_NonHybrid_L	45	3
C521_OBJ_NonHybrid_L	5	2
C540_OBJ_NonHybrid_L	113	15
C556_OBJ_Hybrid_G	425	85
C559_OBJ_Hybrid_G	88	27
NEW001_GOLD	2	2
NEW002_GOLD	1	1
NEW003_GOLD	2	1

Table 13: Golden GCS Evaluation Sample detailed inference results by entity group

C Evaluation Queries for Implicit Object Recovery

C.1 Total recovered objects

```
MATCH (objects:Instance) where objects.is_explicit is not null
return count(distinct objects)
```

count(distinct objects)
78671

Figure 3: Total number of recovered operational objects (implicit or explicit) extracted from the GCS archive.

C.2 Explicit vs implicit distribution

```
MATCH (objects:Instance) where objects.is_explicit is not null
WITH collect(DISTINCT objects) AS objs

RETURN
    size([o IN objs WHERE o.is_explicit = "True"]) AS explicit_nodes,
    size([o IN objs WHERE o.is_explicit = "False"]) AS implicit_nodes;
```

explicit_nodes	implicit_nodes
57290	21381

Figure 4: Global distribution of recovered operational objects into explicit and implicit categories.

C.3 Per-GCS aggregation statistics

```
MATCH (n:Instance:GCS_solution)
OPTIONAL MATCH (n)-[>]-(obj:Instance)
WITH n, collect(DISTINCT obj) AS objs
WITH
    n,
    size([o IN objs WHERE o.is_explicit = "False"]) AS implicit_count
RETURN
    100.0 * sum(CASE WHEN implicit_count > 0 THEN 1 ELSE 0 END) / count(n)
    AS GCS_proportion_with_implicit_object
```

GCS_proportion_with_implicit_object
72.61870503597122

Figure 5: Proportion of GCS instances containing at least one implicit operational object.

```
MATCH (n:Instance:GCS_solution)
OPTIONAL MATCH (n)-[>]-(obj:Instance)
WITH n, collect(DISTINCT obj) AS objs
WITH
    size([o IN objs WHERE o.is_explicit = "True"]) AS explicit_count,
    size([o IN objs WHERE o.is_explicit = "False"]) AS implicit_count
WHERE explicit_count > 0 OR implicit_count > 0
RETURN
    avg(explicit_count) AS avg_explicit,
    avg(implicit_count) AS avg_implicit,
    avg(explicit_count + implicit_count) AS avg_total,
    avg(1.0 * implicit_count / (explicit_count + implicit_count)) AS avg_implicit_ratio
```

avg_explicit	avg_implicit	avg_total	avg_implicit_ratio
9.548477829252136	3.577597617471874	13.12607544672404	0.2812888866349383

Figure 6: Average proportion of implicit operational objects per GCS instance.