A Clinical SKOS Ontology and Evaluation Benchmark for LLM Query Generation over ICU Knowledge Graphs

Khurrum Ali
Indiana University
alikh@iu.edu

Abstract

When clinicians query databases using everyday language—*"code blue patients"* or *"sugar disease"*—Large Language Models must bridge a lexical gap between colloquial speech and formal clinical terminology. While highly capable cloud models can leverage external ontologies like SKOS to resolve these terms via SPARQL queries, hospital privacy regulations often mandate the use of air-gapped local LLMs (4–8B parameters). We evaluate query generation across scales (Gemini 2.0 Flash vs. LLaMA 3.1 8B) using **ClinSKOS-ICU**, a curated ontology of 421 ICU concepts, and **ClinNLU**, an evaluation benchmark. We identify a critical **"Privacy Penalty"**: while Gemini achieves 90.2% ontology deferral under an RDF+SKOS architecture, local LLMs exhibit a 100% **"Semantic Bypass"** vulnerability, hardcoding formal terms into queries rather than deferring to the graph. To improve local LLM grounding, we introduce **Architectural Decomposition**, a pipeline that restricts the LLM to Grammar-Constrained JSON entity extraction and delegates query generation to deterministic code. This structural pivot entirely eliminates Semantic Bypass (0%) and achieves an 80.4% ontology deferral rate on an 8B model, suggesting that decoupled extraction is highly effective for enforcing W3C semantic compliance on privacy-preserving local hardware.

Keywords: SKOS, knowledge graph, clinical NLU, Semantic Bypass, Architectural Decomposition, local LLMs, eICU

1. Introduction

Clinical documentation and bedside conversation are filled with jargon, abbreviations, and colloquialisms that diverge from the formal terminology stored in electronic health record databases. A clinician asking about *"code blue patients"* means cardiac arrest; *"sugar disease"* means diabetes mellitus. When Large Language Models (LLMs) are used to translate such natural language into database queries, they must bridge this *lexical gap*.

While state-of-the-art cloud models perform well on such tasks when given access to semantic web standards (e.g., generating SPARQL queries that traverse a SKOS ontology), hospital environments face a severe deployment constraint: strict Protected Health Information (PHI) regulations generally prohibit transmitting patient-level database slices to external cloud APIs (Thirunavukarasu et al., 2023). As a result, clinical deployments are often forced to rely on air-gapped local LLMs (4–8B parameters) running on local hardware.

We investigate the intersection of ontology grounding and this **Privacy Penalty**. Leveraging the eICU database, we compare query generation across model scales (Gemini 2.0 Flash vs. LLaMA 3.1 8B/Mistral 7B) to determine if local LLMs can safely utilize external synonym infrastructure. We discover a **Semantic Bypass** vulnerability: whereas cloud models correctly let the ontology translate slang terms, local LLMs bypass the ontology entirely, hardcoding formal medical terms directly into the query syntax, which bypasses the intended graph traversal.

To resolve this, we propose and evaluate **Architectural Decomposition**, a methodology that isolates entity extraction from query generation. By restricting the LLM to a strict JSON extraction task via Grammar-Constrained Decoding, the ontology traversal is deterministically enforced by external code rather than generated syntax.

This paper makes three contributions:

1. **ClinSKOS-ICU** and **ClinNLU**: A curated SKOS clinical ontology (421 concepts, 60+ colloquial mappings) and an evaluation benchmark designed to rigorously test clinical query generation.

2. **Identification of Semantic Bypass**: We empirically demonstrate that local LLMs (LLaMA 3.1 8B, Mistral 7B) exhibit a 95–100% Semantic Bypass failure rate on end-to-end SPARQL generation tasks, overriding structural instructions with parametric guessing.

3. **Architectural Decomposition**: We show that decoupling entity extraction from graph traversal structurally limits Semantic Bypass, improving ontology deferral rates to 80.4% on a local 8B-parameter model.

2. Related Work

LLMs for Clinical NLU. Large language models have demonstrated substantial medical knowledge, performing at expert level on clinical benchmarks (Singhal et al., 2023). However, translating this knowledge into correct database queries remains challenging: the model must map its parametric understanding of clinical concepts to the specific schema patterns of the target database

Proceedings of the Knowledge Graphs and Large Language Models Workshop (KG-LLM) @ LREC26, pages 82–92
May 16, 2026.

(Thirunavukarasu et al., 2023). Text-to-SQL approaches (Gu et al., 2023) and text-to-SPARQL methods (Kovriguina et al., 2023) have been explored, but few studies compare these paradigms in clinical settings where terminology varies between colloquial and formal registers.

Knowledge Graphs in Clinical Decision Support. Knowledge graphs constructed from clinical databases such as eICU (Pollard et al., 2018) and MIMIC-III (Johnson et al., 2016) provide structured representations of patient data. Two dominant paradigms exist: Labeled Property Graphs (LPG), exemplified by Neo4j with Cypher queries, which store rich node/edge properties; and RDF triple stores queried via SPARQL, which natively support ontology integration through standards like SKOS (Miles and Bechhofer, 2009) and SNOMED-CT (Donnelly, 2006). While prior work has compared these paradigms on expressiveness and performance (Angles and Gutierrez, 2018; Hogan et al., 2021), few studies empirically evaluate their impact on LLM-generated clinical queries.

Grounding and Hallucination Mitigation. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) grounds LLM outputs in retrieved evidence, reducing hallucination. Surveys of LLM hallucination (Ji et al., 2023) identify factual fabrication as a persistent risk, particularly in high-stakes domains. Our work extends RAG by grounding not only the LLM's *answers* but its *queries*: the SKOS ontology resolves terminology *before* database execution, ensuring that the query itself is semantically correct, not just the generation that follows.

3. Resources and System Architecture

This section describes the language resources, ontology design, and experimental conditions that underpin our evaluation. We first introduce the data source (Section 3.1), then the ClinSKOS-ICU ontology (Section 3.2), and finally the four query conditions under evaluation (Section 3.3).

3.1. Data Source

All data and evaluation in this work are in English. We use the eICU Collaborative Research Database (Pollard et al., 2018), a multi-center critical care dataset containing de-identified health data from over 200,000 ICU admissions across 208 US hospitals. A curated cohort of approximately 2,000 patient stays spans seven organ systems: cardiovascular, pulmonary, neurologic, infectious disease, renal, gastrointestinal, and endocrine. The knowledge graph encodes five entity types (Hospital, Patient, Diagnosis, OrganSystem, Drug) with five relationship types (Figure 1), and diagnoses follow a hierarchical, pipe-delimited encoding with five levels: *organ_system*|*category*|*problem*|*detail*|*qualifier*.

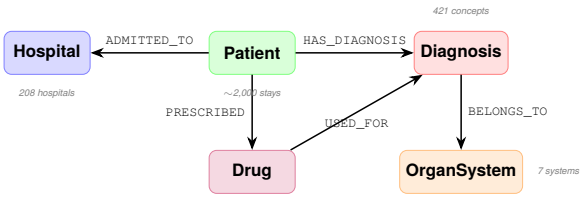

Figure 1: Knowledge graph schema showing five entity types and their relationships. Node counts reflect the curated eICU cohort.

3.2. ClinSKOS-ICU: A Clinical SKOS Ontology

ClinSKOS-ICU is a curated SKOS ontology designed to bridge the lexical gap between clinical colloquialisms and formal database terminology. It comprises **421 SKOS concepts** extracted from the unique `problem` values of the eICU diagnosis hierarchy, enriched with two layers of synonyms:

Curated synonyms (60+ mappings). Domain-expert mappings from colloquial terms to formal diagnoses, stored as `skos:altLabel` entries (Table 1).

Table 1: Representative ClinSKOS-ICU synonym mappings.

Canonical Term (`prefLabel`)	**Colloquial Synonyms (`altLabel`)**
cardiac arrest	code blue, asystole, pulseless, PEA
diabetes mellitus	sugar disease, diabetic, hyperglycemia, dm
cardiogenic shock	pump failure, heart pump failure
atrial fibrillation	irregular heartbeat, a-fib, AF
congestive heart failure	fluid overload, heart failure, CHF

Auto-generated synonyms. Individual tokens extracted from multi-word problem names (excluding stop words), providing partial-match capability for formal terminology.

Resolution mechanism. A clinician's query for *"code blue patients"* is resolved as: query term → `skos:altLabel "code blue"` → `skos:prefLabel "cardiac arrest"` →

`eicu:problem "cardiac arrest"` → matching patient records.

3.3. Four Conditions Under Evaluation

We evaluate four query conditions that isolate the effect of synonym infrastructure on LLM-generated clinical queries. All conditions query the same underlying eICU data; they differ only in synonym resolution mechanism:

Condition 1: Ungrounded SQL (Baseline). The LLM generates SQL queries against PostgreSQL using `ILIKE` pattern matching. The LLM relies on its parametric knowledge to expand clinical terms—e.g., generating `ILIKE '%cardiac arrest%'` when asked about "code blue." This system has no formal synonym resource; synonym resolution depends entirely on the LLM's medical knowledge.

Condition 2: SPARQL without SKOS (Ablation). The LLM generates SPARQL queries against the OxiGraph RDF store, but the prompt explicitly instructs the model *not* to use SKOS vocabulary: the system prompt omits all `skos:` prefixes and few-shot examples demonstrate only direct `FILTER(CONTAINS(...))` matching against `eicu:problem` values. This ablation isolates whether the query language (SPARQL) itself provides any advantage over SQL.

Condition 3: Ontology-Grounded RDF+SKOS (Proposed). The LLM generates SPARQL queries against an OxiGraph RDF triple store ($\sim$51,000 triples) augmented with the ClinSKOS-ICU ontology. The triple store schema (namespace prefixes, entity types, and relationship predicates) is injected into the system prompt, and few-shot examples demonstrate the `altLabel` → `prefLabel` traversal pattern. The LLM does not have direct access to the triple store; it generates SPARQL strings that are executed by the evaluation harness against OxiGraph via its HTTP SPARQL endpoint. This system provides formal, W3C-standardized synonym resolution independent of the LLM's parametric knowledge.

Condition 4: SQL + Synonym Dictionary. The question is preprocessed through a Python dictionary that maps colloquial terms to formal diagnoses (derived from the same synonym mappings as ClinSKOS-ICU). The resolved question is then passed to the LLM for SQL generation. This condition tests whether synonym resolution can be implemented *outside* the graph layer—simulating what an LPG system with custom synonym nodes would achieve.

Conditions 1–2 pass the raw query directly to the LLM, relying on parametric knowledge. Condition 3 inserts a SKOS ontology traversal (`altLabel` → `prefLabel` → `eicu:problem`) within the SPARQL query. Condition 4 preprocesses the input through a synonym dictionary *before* LLM generation.

4. ClinNLU Benchmark and Experimental Design

The ClinNLU benchmark evaluates two dimensions of clinical Natural Language Understanding: Lexical Grounding and Semantic Reasoning, representing core capabilities required for effective clinical query generation.

4.1. Lexical Grounding

The lexical grounding task evaluates whether the system can resolve clinical slang and abbreviations to correct formal concepts. We test 102 questions phrased in colloquial language, each mapping to a specific formal diagnosis. The questions span 10 clinical categories (cardiovascular, respiratory, neurologic, renal, infectious, GI/hepatic, metabolic, hematologic, trauma/toxicology, pain/cardiac) with explicit `slang_term` → `expected_problem` ground truth (Table 2).

Table 2: Representative ClinNLU lexical grounding examples.

Colloquial Term	Expected Formal Concept
sugar disease	diabetes mellitus
code blue	cardiac arrest
pump failure	cardiogenic shock
irregular heartbeat	atrial fibrillation
high blood pressure	hypertension

We evaluate four conditions under few-shot prompting (2–3 worked examples): (1) ungrounded SQL, (2) SPARQL without SKOS (ablation), (3) ontology-grounded RDF+SKOS, and (4) SQL with synonym dictionary preprocessing (simulating LPG with custom synonym nodes). McNemar's test is used for paired binary (pass/fail) comparisons between conditions.

4.2. Semantic Reasoning

The semantic reasoning task evaluates whether the system can perform semantic traversal beyond flat lexical substitution. We designed a 150-query **Semantic Reasoning** benchmark across three 50-query tiers, with representative examples:

1. **Lexical** (`skos:altLabel`): Resolving colloquial slang to formal terms.

Example: "Count the number of patients who suffered from the flu." → extract *"the flu"* → resolve via `altLabel` to "influenza."

2. **Hierarchical** (`skos:broader`): Subsumption queries mapping specific diagnoses up to broad organ systems.
   Example: "How many patients have conditions related to the cardiovascular system?" → extract *"cardiovascular"* → traverse `skos:broader` to the organ system node.

3. **Cross-Standard** (`skos:exactMatch`): Mapping internal terminology to external standards like ICD-9/ICD-10.
   Example: "How many patients were diagnosed with ICD-9 code C61?" → extract *"C61"* → match via `skos:exactMatch`.

These represent the core semantic relationships required for clinical Natural Language Understanding. We evaluate Mistral 7B and LLaMA 3.1 8B on this benchmark to compare a **Flat Dictionary** approach against the W3C **RDF+SKOS** architecture.

4.3. Substring Matching Failure Modes

Conditions 1–2 rely on substring matching, allowing us to characterize systematic failure modes independent of query language.

4.4. Models and Conditions

Primary experiments for the $N = 102$ text-to-SQL benchmark use the Gemini 2.0 Flash efficiency tier, accessed via the `langchain_google_genai` Python library. While future work should assess whether heavier reasoning models (e.g., the 'Pro' tier) can overcome the multi-hop SPARQL syntax barrier, strict PHI regulations legally prohibit routing patient data to external cloud APIs of any scale. Therefore, resolving the syntax failures of local LLMs via Architectural Decomposition remains the most pragmatic clinical imperative. Conversely, the 150-query semantic reasoning extraction benchmark (Section 5.4) is evaluated under strictly zero-shot conditions across all models to test structural reliance without prompt bias.

We additionally evaluate Mistral 7B (`mistral:latest`) and LLaMA 3.1 8B (`llama3.1:8b`) locally via the Ollama inference server (HTTP API at `localhost:11434`) on an Apple Silicon M4 with 16 GB unified RAM, using default 4-bit quantization (Q4_0) as shipped by Ollama. All local model calls use **temperature 0.0** (deterministic decoding), **`num_predict` = 64** (maximum output tokens), and **`format` = "json"** to enforce structured JSON output. Each prompt includes a multi-shot preamble (6–8 worked examples per tier) followed by the target question. Because local LLMs are susceptible to hallucination with full database schemas, we evaluate them using **Extractive Schema Linking** (entity-first retrieval with minimal schema injection) and **Grammar-Constrained Decoding (GCD)** via strict JSON output formatting.

5. Results

We report results across both ClinNLU benchmarks: the lexical grounding evaluation ($N = 102$, Sections 5.1–5.3), the semantic reasoning benchmark ($N = 150$, Section 5.4), and a paradigm-aware analysis that distinguishes ontology deferral from semantic bypass (Section 5.5).

5.1. Four-Condition Comparison

Table 3 consolidates results across all four query conditions. Unless otherwise noted, Sections 5.1–5.3 report results using Gemini 2.0 Flash.

Table 3: Results across four query conditions on the ClinNLU benchmark (few-shot, $N = 102$). Best results in **bold**.

Metric	SQL	SPARQL$_{\neg SKOS}$	RDF+SKOS	SQL+syn
Success rate	42.2%	23.5%	72.5%	**94.1%**

Lexical grounding ($N = 102$). RDF+SKOS achieves 72.5% (74/102), a 30.3pp improvement over ungrounded SQL at 42.2% (McNemar's $\chi^2 = 15.79$, $p < 0.001$). Removing SKOS drops SPARQL to 23.5%—*worse* than SQL—confirming the ontology drives the improvement ($\chi^2 = 48.02$, $p < 0.001$). The SPARQL-without-SKOS condition underperforms even ungrounded SQL, likely because LLM pretraining corpora contain substantially more SQL than SPARQL (reflecting the prevalence of SQL on forums such as Stack Overflow), making LLMs less proficient at generating valid SPARQL syntax. SQL+synonym dictionary achieves the highest rate: 94.1% (96/102, $\chi^2 = 18.38$, $p < 0.001$ vs. RDF+SKOS), demonstrating that synonym infrastructure is the critical capability regardless of implementation mechanism.

5.2. Few-Shot Learning Impact

The most striking pattern across all three experiments is the dramatic impact of few-shot prompting on the RDF+SKOS system (Table 4). LLMs already understand clinical language; what they lack is schema pattern knowledge. Just 2–3 worked examples bridge this gap.

Table 4: Impact of few-shot prompting on RDF+SKOS success rate (Gemini 2.0 Flash, $N = 102$).

Metric	Zero-Shot	Few-Shot	Δ
Success rate	3%	72.5%	+69.5pp

5.3. Ablation: Isolating the SKOS Contribution

Table 5 isolates each component's contribution.

Table 5: Ablation study isolating synonym infrastructure contribution ($N = 102$). $^{**}p<0.01$; $^{***}p<0.001$.

	Condition	Synonym	Success	χ^2 vs. SQL
1	SQL (ungrounded)	None	42.2%	—
2	SPARQL (no SKOS)	None	23.5%	7.20**
3	RDF + SKOS	SKOS ontology	72.5%	15.79***
4	SQL + synonym dict	Python dict	**94.1%**	44.33***

5.4. Semantic Reasoning Results

Table 6 presents the zero-shot 150-query semantic reasoning benchmark. While the flat dictionary matches RDF+SKOS on pure lexical substitution (Tier 1), it collapses entirely (0%) on hierarchical subsumption and cross-standard traversals. Under zero-shot conditions, even the RDF+SKOS architecture achieves low success on complex tiers (Gemini: 10% T2/T3; LLaMA: 8%/2%).

However, these zero-shot floors dramatically understate the architecture's potential. Table 7 shows that adding 6–8 worked examples per tier unlocks latent capacity: Tier 2 jumps from 8–20% to 78–90% across all models (+70–78 pp), isolating *prompt engineering*—not model capacity—as the bottleneck. This parallels the lexical few-shot effect (Table 4) and confirms that the ontology provides the structural pathway; models simply need schema-pattern demonstrations to use it. Tier 3 (cross-standard mapping) remains resistant to prompting (10–22%), suggesting that `skos:exactMatch` traversal with external code sets presents a qualitatively harder generation challenge.

Table 6: Semantic Reasoning benchmark ($N = 150$). Success rates for Dictionary (Dict) vs. RDF+SKOS across Gemini 2.0 Flash, LLaMA 3.1 8B, and Mistral 7B.

Model	Tier	Dict	RDF+SKOS
Gemini 2.0 Flash	1: Lexical	100.0%	100.0%
Gemini 2.0 Flash	2: Hierarchical	0%	10.0%
Gemini 2.0 Flash	3: Cross-Standard	0%	10.0%
LLaMA 3.1	1: Lexical	100.0%	100.0%
LLaMA 3.1	2: Hierarchical	0%	8.0%
LLaMA 3.1	3: Cross-Standard	0%	2.0%
Mistral 7B	1: Lexical	94.0%	98.0%
Mistral 7B	2: Hierarchical	0%	20.0%
Mistral 7B	3: Cross-Standard	0%	6.0%

Table 7: Multi-shot semantic reasoning ($N = 150$, 6–8 examples per tier). Δ shows improvement over zero-shot (Table 6).

Model	Tier	0-shot	Multi-shot	Δ
Gemini 2.0 Flash	1: Lexical	100%	100%	—
Gemini 2.0 Flash	2: Hierarchical	10%	90%	+80 pp
Gemini 2.0 Flash	3: Cross-Standard	10%	22%	+12 pp
LLaMA 3.1 8B	1: Lexical	100%	100%	—
LLaMA 3.1 8B	2: Hierarchical	8%	78%	+70 pp
LLaMA 3.1 8B	3: Cross-Standard	2%	10%	+8 pp
Mistral 7B	1: Lexical	98%	100%	+2 pp
Mistral 7B	2: Hierarchical	20%	90%	+70 pp
Mistral 7B	3: Cross-Standard	6%	10%	+4 pp

5.5. Paradigm-Aware Evaluation: Ontology Deferral vs. Semantic Bypass

The preceding evaluation measures whether the *correct diagnosis* appears somewhere in the generated query. However, for a grounded system (RDF+SKOS), this metric conflates two fundamentally different behaviors: (a) the LLM *defers* to the ontology by using the slang term to search `altLabel`, letting SKOS resolve it; or (b) the LLM *bypasses* the ontology by hardcoding the formal term directly, rendering the SKOS layer useless.

We introduce a **paradigm-aware** grading scheme:

- **Ungrounded (SQL):** Success = formal term present in query (LLM must parametrically translate).
- **Grounded (SPARQL+SKOS): Ontology Deferral** = slang term present in query AND formal term *not* hardcoded. **Semantic Bypass** = formal term hardcoded (ontology rendered useless).

To evaluate Semantic Bypass (Table 8), all models, including local LLMs, were provided with identical few-shot examples demonstrating `skos:altLabel` traversal. Table 8 reports results under this corrected evaluation.

Table 8: Paradigm-aware evaluation ($N = 102$). For SQL, success requires the formal term in the query. For SPARQL+SKOS, we distinguish ontology deferral (correct) from semantic bypass (architecture neutralized).

Model	Architecture	✓ Deferred	△ Bypass	✗ Fail
Gemini 2.0 Flash	SQL (ungrounded)	—	—	40.2%
Gemini 2.0 Flash	SPARQL+SKOS	**90.2%**	4.9%	4.9%
LLaMA 3.1 8B	SQL (ungrounded)	—	—	0%
LLaMA 3.1 8B	SPARQL+SKOS	0%	**100%**	0%
Mistral 7B	SQL (ungrounded)	—	—	0%
Mistral 7B	SPARQL+SKOS	2.0%	**95.1%**	2.9%

Gemini defers; local LLMs bypass. Gemini 2.0 Flash correctly defers to the SKOS ontology **90.2%** of the time—it generates `FILTER(CONTAINS(LCASE(?alt), "sugar disease"))` and traverses `altLabel` → `prefLabel`, allowing the expert-curated graph to handle the clinical translation. The 5 "bypasses" are cases where the slang term *is* the formal term (e.g., "septic shock" → "septic shock").

In contrast, both local LLMs exhibit **high rates of semantic bypass**: LLaMA 3.1 8B bypasses the ontology on 100% of SPARQL queries, and Mistral 7B on 95.1%. Rather than using the slang term to search `altLabel`, both models tend to hardcode the formal diagnosis term directly into the SPARQL query—bypassing the intended safety checks of the SKOS architecture.

6. Discussion: Three Acts of Clinical NLU

Our paradigm-aware evaluation reveals a three-act narrative with direct implications for deploying LLM-powered clinical query systems.

6.1. Act 1: Why Ontologies Are Necessary

Even a highly capable cloud model (Gemini 2.0 Flash) can only parametrically resolve colloquial clinical terms to the correct formal diagnosis **59.8%** of the time when ungrounded (our paradigm-aware metric; the strict execution success rate in Table 3 is 42.2%). Consequently, relying on parametric memory alone is insufficient for robust terminology translation. An external synonym resource remains a critical requirement, a point reinforced by our ablation where SPARQL without SKOS (23.5%) performs worse than ungrounded SQL.

6.2. Act 2: The Necessity of Taxonomy and Limits of SPARQL

When given the RDF+SKOS architecture, Gemini understands the assignment. It correctly defers to the `altLabel` → `prefLabel` traversal pattern **90.2%** of the time (92/102 queries), allowing the deterministic, expert-curated ontology to handle the clinical translation rather than attempting parametric guessing. The architecture is sound: the LLM generates SPARQL that searches for the *slang term* via `skos:altLabel`, and the graph resolves it to the correct formal concept.

The 5 apparent "bypasses" (4.9%) are cases where the slang term is identical to the formal term (e.g., "septic shock," "hepatic failure")—these are not architectural failures but benchmark edge cases where no translation is needed. The 5 true failures (4.9%) reflect ontology coverage gaps for terms not yet registered as `altLabel` entries.

Augmenting SQL with a Python synonym dictionary achieves an even higher raw success rate (94.1%) on 1:1 lexical substitution. However, this success is parasitic on the underlying ontology: dictionaries do not author themselves. Building a comprehensive map of clinical slang requires the rigorous curation processes formalized by SKOS. Furthermore, while flat dictionaries *could* in principle be extended to encode hierarchical or cross-standard relationships, doing so would effectively re-implement the ontology without its formal governance guarantees (versioning, transitivity, W3C interoperability). Consequently, while a dictionary artificially inflates text-to-SQL benchmark scores on isolated queries, it presents an intractable maintenance burden at scale and fundamentally fails in real-world use cases requiring compositional semantic traversal.

6.3. Act 3: The Deployment Crisis

Hospital environments cannot legally transmit Protected Health Information (PHI) to external cloud APIs. Clinical deployments are therefore mandated to use air-gapped local LLMs such as LLaMA 3.1 8B and Mistral 7B, running on local hardware within the institution's security perimeter.

As demonstrated by our ungrounded ablation (Table 12), local LLMs completely fail baseline SQL query generation due to strict string-matching brittleness. However, even when provided with the complete RDF+SKOS architecture and identical W3C-compliant semantic traversal examples (Table 8), our paradigm-aware evaluation reveals that local LLMs **frequently bypass the intended safety mechanisms**. Rather than deferring to the ontology, both local models exhibit a strong **Semantic Bypass** bias:

- **LLaMA 3.1 8B**: 100% semantic bypass (0/102 ontology deferrals)
- **Mistral 7B**: 95.1% semantic bypass (2/102 ontology deferrals)

Both models hardcode the formal diagnosis term directly into the SPARQL query, bypassing the `altLabel` traversal entirely. The SKOS ontology—carefully curated to provide safe, deterministic synonym resolution—is rendered invisible to the local model. This is not a knowledge failure (the local LLMs correctly identify the formal term) but a *behavioral* failure: the models default to the parametric shortcut of hardcoding the answer rather than following the `altLabel` $\rightarrow$ `prefLabel` pattern, even when provided with few-shot examples of the traversal.

Architectural Decomposition: Rescuing Local LLM Grounding. Having identified the Semantic Bypass vulnerability in end-to-end SPARQL generation (where local LLMs bypassed the ontology 100% of the time), we evaluated an Architectural Decomposition pipeline. We restricted the LLaMA 3.1 8B model to a strict JSON extraction task via Grammar-Constrained Decoding, isolating entity extraction from query syntax generation. The extracted entities were then deterministically passed to the SKOS ontology for semantic resolution.

Table 9 summarizes the outcomes of Architectural Decomposition on the $N = 102$ lexical benchmark.

Table 9: Architectural Decomposition outcomes on the lexical benchmark ($N = 102$).

Outcome	Rate	Count
Semantic Bypass	0.0%	0/102
Ontology Deferral	80.4%	82/102
Extraction failure / missing keyword	14.7%	15/102
Pass-through (colloquial = formal)	4.9%	5/102

This establishes that while end-to-end local LLM query generation creates an illusion of grounding, decoupling entity extraction from graph traversal structurally enforces W3C semantic compliance on privacy-preserving local hardware.

7. Limitations

The ClinNLU benchmark size ($N = 102$) remains relatively small; scaling is needed. The ClinSKOS-ICU ontology has coverage gaps for clinical abbreviations (DIC, PE, MVA, AKI) and symptom-as-diagnosis terms, explaining its 72.5% vs. the dictionary's 94.1%. All primary experiments use the Gemini 2.0 Flash efficiency tier. While future work should assess whether heavier reasoning models (e.g., the 'Pro' tier) can overcome the multi-hop SPARQL syntax barrier, strict PHI regulations legally prohibit routing patient data to external cloud APIs of any scale. Therefore, resolving the syntax failures of local LLMs via Architectural Decomposition remains the most pragmatic clinical imperative. The SQL+synonym condition uses the same synonym mappings as SKOS, making the comparison controlled but potentially overfitting. Future work should also include same-family model comparisons (e.g., generic vs. domain-specific variants within the same architecture) to isolate the effect of domain pretraining on ontology compliance. All experiments use English data; extending the approach to morphologically rich or inflectional languages would require more sophisticated synonym matching (e.g., lemmatization before `altLabel` lookup) and is left to future work. Fine-tuning local LLMs on SPARQL generation with `altLabel` traversal patterns could reduce Semantic Bypass without requiring Architectural Decomposition; this is a promising direction for future investigation. No clinician-in-the-loop evaluation has been conducted.

8. Conclusion

We contribute two reusable language resources: **ClinSKOS-ICU**, a 421-concept SKOS clinical ontology with 60+ synonym mappings, and **ClinNLU**, an evaluation benchmark (102 lexical probes). We further introduce a **paradigm-aware evaluation methodology** that distinguishes genuine ontology deferral from semantic bypass in grounded systems.

Our analysis reveals a critical deployment gap in clinical NLU:

1. **Ontologies are necessary**: Even highly capable cloud models achieve only 59.8% ungrounded SQL success on clinical slang, indicating that parametric medical knowledge alone is insufficient. Crucially, while flat dictionaries initially appear capable for 1:1 lexical grounding (94.1%), they are artifacts of ontological modeling rather than replacements for it. Without a standardized ontological framework (like SKOS) to govern and version these mappings, maintaining ad-hoc dictionaries across multi-institutional networks is intractable. Furthermore, our 150-query semantic benchmark demonstrated that the synonym dictionary—designed for 1:1 lexical substitution—collapses to 0.0% accuracy when queries require hierarchical subsumption or cross-standard mapping. The RDF+SKOS architecture provides a structural pathway to resolve these interactions: multi-shot prompting unlocks 78–90% Tier 2 success across all models (up from 8–20% zero-shot), confirming that the ontology provides the necessary structure while prompt engineering remains the primary bottleneck. Tier 3 cross-standard

mapping (10–22% even with prompting) remains an open challenge.

2. **The architecture works**: Gemini defers to the SKOS ontology 90.2% of the time when given the RDF+SKOS pipeline, validating the architectural design.

3. **Local LLMs bypass the safety layer**: LLaMA 3.1 8B (100%) and Mistral 7B (95.1%) exhibit high rates of semantic bypass, hardcoding formal terms directly. Given that hospital PHI constraints often necessitate local LLM deployment (Thirunavukarasu et al., 2023), mitigating this bypass is a key engineering challenge for clinical NLU.

We propose **Architectural Decomposition** as a highly effective solution. Rather than attempting to prompt an 8B-parameter model into structural compliance, the local LLM's role can be restricted to Grammar-Constrained feature extraction (e.g., rigid JSON output). By moving SPARQL synthesis back into deterministic code, semantic bypass is structurally prevented, preserving the SKOS design even under air-gapped deployment conditions. A decision framework unifying these architectural trade-offs is provided in Table 13.

Acknowledgements

I would like to thank Dr. David Leake and Dr. Damir Cavar for their guidance throughout this work. I gratefully acknowledge Indiana University Bloomington's Luddy School of Informatics, Computing, and Engineering for providing computational resources, and MIT for making the clinical datasets available. I thank my parents, my mother for her unwavering support, and my late father, may he rest in peace. Above all, gratitude is owed to God and whatever supreme power there may be.

9. Bibliographical References

Renzo Angles and Claudio Gutierrez. 2018. An introduction to graph data management. *Graph Data Management: Fundamental Issues and Recent Developments*, pages 1–32.

Zihui Gu, Ju Fan, Nan Tang, Lei Cao, and Guoliang Li. 2023. Few-shot text-to-SQL translation using structure and content prompt learning. In *Proceedings of the ACM on Management of Data*, volume 1, pages 1–28.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, Jose Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. In *ACM Computing Surveys*, volume 54, pages 1–37. ACM.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Liubov Kovriguina, Roman Teucher, and Ricardo Usbeck. 2023. SPARQL generation with pretrained language models. In *Proceedings of the International Semantic Web Conference*, pages 238–254. Springer.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature Medicine*, 29(8):1930–1940.

10. Language Resource References

Kevin Donnelly. 2006. SNOMED-CT: The advanced terminology and coding system for eHealth. *Studies in Health Technology and Informatics*, 121:279–290.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. In *Scientific Data*, volume 3, pages 1–9.

Alistair Miles and Sean Bechhofer. 2009. SKOS Simple Knowledge Organization System Reference. Technical report, W3C. W3C Recommendation, https://www.w3.org/TR/skos-reference/.

Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific Data*, 5(1):1–13.

A. Appendix: Qualitative Query Examples

Table 10 shows cases where synonym infrastructure succeeds and substring matching fails. Table 11 categorizes failure modes from the $N = 102$ evaluation.

Table 10: Success examples ($N = 102$): synonym infrastructure resolves clinical slang that literal matching cannot.

Slang Term	Formal Term	Slang Type	SQL	SPARQL$_{\neg SKOS}$	RDF+SKOS	SQL+syn
code blue	cardiac arrest	Hospital code	✗	✗	✓	✓
pump failure	cardiogenic shock	Clinical jargon	✗	✗	✓	✓
brain attack	stroke	Patient-facing	✗	✗	✓	✓
low blood pressure	hypotension	Lay description	✗	✗	✓	✓
NSTEMI	myocardial infarction	Abbreviation	✗	✗	✓	✓

Table 11: Failure taxonomy from the $N = 102$ evaluation, showing where each system fails and why.

Failure Type	Example	Explanation	RDF+SKOS	SQL+syn	SQL
Ontology gap	"wheezing" → asthma	Not registered as altLabel; synonym dict has it	✗	✓	✗
Abbreviation gap	"DIC" → DIC syndrome	Short abbreviation not in altLabel map	✗	✓	✓
Over-correction	"bleeding" → hemorrhagic shock	Dict maps to overly specific term; raw "bleeding" is in DB	✓	✗	✓
SPARQL syntax	Mortality by bed cat.	LLM uses `xsd:boolean` without prefix declaration	✗	N/A	✓
No synonym exists	"blood infection" → sepsis	Not in any synonym map; requires parametric knowledge	✗	✗	✗

Table 12: Baseline ungrounded ablation: local LLM failure modes ($N = 102$). Models received entity-grounded prompts without few-shot examples or SKOS access.

Model	Failure Type	Example	SQL	SPARQL
LLaMA 3.1 8B	Exact equality	`WHERE diagnosisstring = 'myocardial infarction'` fails against pipe-delimited paths	97.1% fail	—
LLaMA 3.1 8B	Bare graph pattern	`?p eicu:hasDiagnosis ?dx .` without `SELECT WHERE` wrapper	—	24.5% fail
Mistral 7B	Schema halluc.	`patient.patientunitstaysid` (fabricated column name)	100% fail	—
Mistral 7B	Parse errors	Malformed SPARQL: list output instead of string, nested JSON	—	52.9% fail
Contrast: Gemini 2.0 Flash uses `ILIKE '%term%'` *for SQL and SKOS traversal for SPARQL*			42.2% pass	72.5% pass

Table 13: Decision framework for synonym infrastructure in clinical NLU.

Criterion	Recommendation
Synonyms needed, single deployment	Python Dict (requires ontology for curation)
Standardized governance	RDF+SKOS provides native W3C versioning
Multi-system interoperability	RDF+SKOS; flat dictionaries not portable
LLM query generation	SQL+dictionary avoids SPARQL syntax errors

B. Prompt Templates

Below we reproduce the two core prompt templates used in our evaluation pipeline (abridged for space; full versions are included in the replication repository).

RDF+SKOS SPARQL Generation Prompt (Condition 3). The system prompt includes the full RDF schema and three few-shot examples demonstrating the `altLabel` → `prefLabel` traversal pattern:

```
You are an expert SPARQL query generator
for an eICU RDF Knowledge Graph.

=== RULES ===
- The SKOS Ontology Lookup Pattern is
  CRITICAL for resolving clinical terms:
  1. Match keyword against skos:altLabel
  2. Get skos:prefLabel from concept
  3. Use prefLabel to match eicu:problem

=== FEW-SHOT EXAMPLE ===
Q: "Find patients with sugar disease"
PREFIX eicu: <http://eicu.mit.edu/
  ontology/>
PREFIX skos: <...skos/core#>
SELECT DISTINCT ?p ?prob WHERE {
```

```
  ?concept a skos:Concept ;
    skos:altLabel ?alt .
  FILTER(CONTAINS(LCASE(?alt),
    "sugar disease"))
  ?concept skos:prefLabel ?pref .
  ?dx eicu:problem ?pref .
  ?p a eicu:Patient ;
    eicu:hasDiagnosis ?dx .
  ?dx eicu:problem ?prob .
}
```

Architectural Decomposition Extraction Prompt. The local LLM receives *only* an extraction task with Grammar-Constrained Decoding (`format="json"`):

```
You are a clinical data extraction tool.
Your ONLY job is to extract the literal
clinical symptoms, diagnoses, and
demographics from the user's description.

=== STRATEGY ===
1. Extract demographics (age, gender).
   Leave null if unknown.
2. Extract clinical keywords from
   symptoms (e.g., "sugar disease").
3. DO NOT translate or formalize terms.
   Extract the EXACT words or phrases.

=== OUTPUT FORMAT ===
Output STRICT JSON:
{"age": 65, "gender": "Male",
 "keywords": ["sepsis", "shock"]}
```

The extracted keywords are then passed to deterministic Python code that constructs the SPARQL query programmatically, ensuring `altLabel` traversal without relying on the LLM for query syntax.