LLM4VKG: Leveraging Large Language Models for Virtual Knowledge Graph Construction

Guohui Xiao[1,3,†,*], **Lin Ren**[1,3,†], **Guilin Qi**[1,3], **Haohan Xue**[1,3],
Marco Di Panfilo[2] and **Davide Lanti**[2]
[1]School of Computer Science and Engineering, Southeast University, Nanjing, China
[2]Free University of Bozen-Bolzano, Italy
[3]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
{guohui.xiao, renlin, gqi, thex1ay}@seu.edu.cn, marco.dipanfilo@student.unibz.it, lanti@inf.unibz.it

Abstract

Virtual Knowledge Graphs (VKGs) provide an effective solution for data integration but typically require significant expertise for their construction. This process, involving ontology development, schema analysis, and mapping creation, is often hindered by naming ambiguities and matching issues, which traditional rule-based methods struggle to address. Large language models (LLMs), with their ability to process and generate contextually relevant text, offer a potential solution. In this work, we introduce LLM4VKG, a novel framework that leverages LLMs to automatize VKG construction. Experimental evaluation on the RODI benchmark demonstrates that LLM4VKG surpasses state-of-the-art methods, achieving an average F1-score improvement of +17% and a peak gain of +39%. Moreover, LLM4VKG proves robust against incomplete ontologies and can handle complex mappings where current methods fail.

1 Introduction

Data integration and access to legacy data sources using end-user-oriented languages are increasingly challenging. Among various integration approaches, Virtual Knowledge Graphs (VKGs) have gained significant traction, particularly when integrating data from relational databases (DBs) [Xiao *et al.*, 2019]. VKGs replace the rigid structure of traditional tables with a flexible graph model that incorporates domain knowledge while ensuring real-time access to fresh, accurate information. Accessing the data through a VKG can significantly improve question answering accuracy on enterprise databases [Sequeda *et al.*, 2023]. A VKG specification comprises three main components [Calvanese *et al.*, 2023]: *(a)* data sources, storing the actual data; *(b)* a domain ontology, capturing the relevant concepts, relations, and constraints of the domain; *(c)* a set of mappings, linking the data sources to the domain ontology.

As a running example (see Figure 1), suppose that we are working with a database for supply chain management, including three interconnected tables, and aim to publish the data as a Knowledge Graph (KG) following the schema.org ontology. To achieve this, we first analyze whether the tables and columns in the database schema align with classes or properties defined in the ontology. The analysis reveals that tables *Provider* and *Product* can be matched to classes *sc:Vendor* and *sc:Product*. The table *Provider-Product* is matched to the object property *sc:seller*, and the columns *Provider.Tag* and *Product.Desc* are matched to the data property *sc:name*. Then, based on these alignments, we construct a set of mappings that define the transformation rules from the data source to a KG following the schema.org ontology. The developed VKG can be queried using the SPARQL language, and the triples can be materialized for further use.

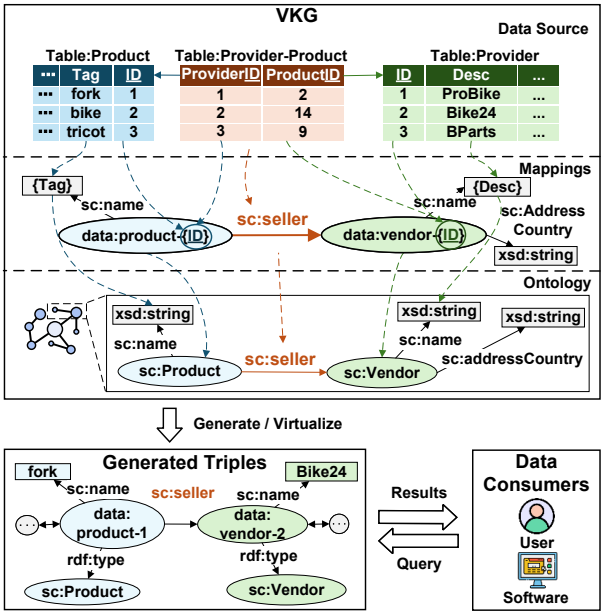

Figure 1: An example of VKG construction. "sc:" means the namespace "https://schema.org/".

Manually constructing mappings is a resource-intensive task that requires considerable human effort, time, domain expertise, and technical skills. Additionally, in real-world scenarios of VKG construction, the ontology is often incom-

plete and must be augmented based on the database schema and data. Recently, several works have been developed for automating or semi-automatic VKG mapping bootstrapping, such as IncMap [Pinkel *et al.*, 2013], BootOX [Jiménez-Ruiz *et al.*, 2015] and ontop [Calvanese *et al.*, 2017]. To evaluate the effect of VKG mapping bootstrapping approaches, Pinkel et al. [2018] introduced a benchmark, RODI, suggesting that those approaches can cope with relatively simple mapping challenges but perform poorly on advanced challenges related to real-world problems.

The emergence of large language models (LLMs) [OpenAI, 2024b; OpenAI, 2024a; Anthropic, 2024; DeepSeek-AI, 2024] represents a promising avenue for addressing these challenges [Chen, 2024; Azaria *et al.*, 2024]. LLMs, with their ability to process and generate contextually relevant text based on learned patterns in language, can assist in interpreting complex relationships between the ontology and the database schema, potentially improving the process of VKG construction.

In this paper, we propose LLM4VKG, a framework to leverage LLMs for VKG construction. The approach leverages LLMs to emulate human decision-making in tasks like attribute naming and concept matching and is guided by formally grounded mapping patterns [Calvanese *et al.*, 2023] to build robust, practical ontology structures. These patterns distill best practices for mapping and ontology design. Moreover, to streamline the evaluation process, we develop a system, based on RODI [Pinkel *et al.*, 2018], to create and evaluate the quality of the generated ontology and mappings.

Our experimental evaluation shows that LLM4VKG outperforms BootOX, the current SOTA method, across all scenarios with an average improvement of +17% in F1-score. Notably, in the "Conference - adjusted naming" scenario, LLM4VKG achieved an absolute improvement of +39%. Surprisingly, LLM4VKG closely matched the performance of the SOTA even with 25% of the ontology vocabulary removed, with an average F1-score of 0.46. These findings highlight the robustness and effectiveness of LLM4VKG in handling incomplete ontologies and complex mapping tasks.

In summary, *(1)* we propose LLM4VKG, the first framework for VKG construction leveraging LLMs and mapping patterns to complete ontologies and bootstrap mappings automatically; *(2)* in LLM4VKG, we design an alignment strategy that leverages the capabilities of LLMs for language understanding and uncertain decision-making to solve problems - attribute naming and concept matching - that are challenging for traditional rule-based methods; *(3)* we implement the proposed methodology and conduct experiments using variations of the RODI benchmark to validate the effectiveness of LLM4VKG. All code and datasets associated with this work are publicly available.[1]

2 Related Work

2.1 Mapping Bootstrapping Systems

Over the past two decades, numerous methods have been developed to create mappings from databases. These methods vary by purpose (e.g., VKGs, data integration, ontology learning, DB schema validation via reasoning), the languages used (e.g., OWL profiles, RDFS, R2RML, or custom mapping languages), and the degree of automation [Calvanese *et al.*, 2023; Mosca *et al.*, 2023; Pinkel *et al.*, 2018; Mogotlane and Dombeu, 2016; Sequeda and Miranker, 2015]. Here, we focus on actual implementations, specifically on the most relevant (semi-)automated tools for bootstrapping VKGs starting from a relational database.

Ontop [Calvanese *et al.*, 2017] automates the generation of R2RML [Das *et al.*, 2012] mappings, ensuring that the RDF graph largely conforms to the Direct Mapping recommendation (with minor exceptions, e.g., duplicate row handling [Calvanese *et al.*, 2023]). Similarly, D2RQ [Bizer and Seaborne, 2004] employs reverse engineering to identify many-to-many relationships, translates foreign keys into object properties, and utilizes a custom mapping language.

MIRROR [de Medeiros *et al.*, 2015] detects class hierarchies and many-to-many relationships, which are not directly expressible in relational schemas, enriching R2RML mappings. BootOX [Jiménez-Ruiz *et al.*, 2015] supports specifying OWL profiles, designing complex R2RML mappings with selection and join operators, recognizing class and property hierarchies, and aligning to target ontologies using LogMap [Jiménez-Ruiz and Grau, 2011].

COMA++ [Aumueller *et al.*, 2005] matches schemas in heterogeneous formats by unifying them into a common model and applying diverse matching techniques. IncMap [Pinkel *et al.*, 2013] represents ontologies and schemas as graphs, computes ranked correspondence using lexical and structural similarities, and generates semi-automatic mappings with user approval. Karma [Gupta *et al.*, 2012], a leading relational-to-ontology mapping tool, supports data extraction from various sources, cleaning, normalization, mapping to a target vocabulary, multi-source integration, and publishing in formats like RDF and CSV.

2.2 LLMs for Ontology Engineering

LLMs are transforming ontology engineering by enabling semantic-driven alignment and automated learning, significantly reducing manual curation while preserving domain fidelity across tasks.

Ontology Alignment Recent LLM-driven approaches demonstrate significant advances in ontology alignment. OLALA [Hertling and Paulheim, 2023] leverages open-source LLMs with zero-shot/few-shot prompting to achieve supervised-level performance on OAEI benchmarks using minimal training data. LLMs4OM [Giglou *et al.*, 2024b] employs multi-representation prompting strategies for cross-domain ontology matching, outperforming traditional systems through context-aware retrieval modules. While OWL2Vec4OA [Teymurova *et al.*, 2024] integrates LLM-augmented embeddings to refine structural alignment, all methods highlight LLMs' capability to reduce dependency on manual annotations and domain-specific constraints.

Ontology Learning LLM-based approaches are advancing ontology learning through diverse methodologies. LLMs4OL [Giglou *et al.*, 2024a] demonstrates LLMs' partial efficacy in concept extraction tasks but highlights the neces-

[1] https://github.com/HomuraT/LLM4VKG

sity of fine-tuning for complex reasoning scenarios. Kommineni *et al.* [2024] establishes a semi-automated LLM-RAG pipeline that reduces expert dependency in ontology construction while retaining human validation mechanisms. Specialized applications like Li *et al.* [2024] show LLMs' ability to generate educational ontologies from unstructured materials, enhancing downstream analytics through structured knowledge integration. For systematic learning, Lo *et al.* [2024] proposes an end-to-end LLM framework with adaptive regularization, outperforming subtask-based approaches in cross-domain ontology generation. These works collectively emphasize LLMs' role in balancing automation with domain-specific knowledge refinement.

3 Preliminaries

This section introduces the key concepts, definitions, and notations used in this work, including VKGs, mapping patterns, and the task definition.

3.1 Virtual Knowledge Graph

In this work, tuples are denoted by the **bold** font.

A *VKG specification* is a triple $(\mathcal{T}, \mathcal{M}, \Sigma)$, where $\mathcal{T}$ denotes an *ontology* (sometimes called *TBox*), $\mathcal{M}$ denotes a set of *mappings*, and Σ denotes a *DB schema*.

The schema Σ consists of a collection of table schemata in the form $T(x_1, \ldots, x_n)$, denoting a table T of attributes $x_1, \ldots, x_n$, and a set of database constraints, in this work consisting of primary and foreign keys. With a slight abuse of notation, we write $e \in \Sigma$ to indicate that e is a schema element, i.e., a table T or an attribute x_i. We use notation $Q(\mathbf{x})$ to denote a SQL query Q of *answer attributes* $\mathbf{x}$. Given a *database instance* $\mathcal{D}$ of Σ, the *evaluation* $Q(\boldsymbol{x})^{\mathcal{D}}$ *of* $Q(\boldsymbol{x})$ *over* $\mathcal{D}$ is a *set* (ignoring duplicates) of *answers* $(\mathbf{x} \mapsto \mathbf{o})$ mapping attributes in $\mathbf{x}$ to values in $\mathcal{D}$.

An ontology $\mathcal{T}$ is a set of axioms in some Description Logic (DL) language (usually DL-Lite$_{\mathcal{R}}$ [Poggi *et al.*, 2008], the mathematical underpinning of OWL 2 QL [Motik *et al.*, 2012]). Axioms are expressed in terms of three pairwise-disjoint sets NC, NP, and ND denoting respectively *class names*, *object property names*, and *data property names*. Without loss of generality, we assume these three sets to contain only class and property names actually occurring in $\mathcal{T}$.

A Knowledge Base (KB) is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is an ontology and $\mathcal{A}$ is a KG (sometimes called ABox). As custom in the DL literature, we assume $\mathcal{A}$ to be a set of unary and binary first-order atoms (intuitively, each atom corresponds to an RDF triple). In VKGs, $\mathcal{A}$ is not materialized, but provided indirectly by means of mappings.

A *mapping* $m \in \mathcal{M}$ is a pair of the form $(s{:}Q(\mathbf{x}), t{:}\mathbf{L})$ where the expression $s{:}Q(\mathbf{x})$, called *source*, specifies a SQL query $Q(\mathbf{x})$ over Σ and of answer attributes $\mathbf{x}$, and expression $t{:}\mathbf{L}$, called *target*, specifies a set $\mathbf{L}$ of first-order atoms. Atoms in $\mathbf{L}$ can be of the form $C(t_1(\mathbf{x}_1))$, $p(t_1(\mathbf{x}_1), t_2(\mathbf{x}_2))$, or $d(t_1(\mathbf{x}_1), t_2(\mathbf{x}_2))$, where $C \in$ NC, $p \in$ NP, $d \in$ ND, and $t_1(\mathbf{x}_1)$ and $t_2(\mathbf{x}_2)$ are first-order terms called *templates*.

For example, the following is a *mapping* using the (concrete) syntax of the Ontop VKG system [Calvanese *et al.*, 2017] for $C(t_1(\mathbf{x}_1))$. $Q(\mathbf{x})$ is "SELECT ID FROM Provider", $L(t(\mathbf{x}))$ is "data:vendor-{ID} a sc:Vendor", and $\mathbf{x}$ is (*ID*).

```
Example mapping 1:
source SELECT ID FROM Provider
target data:vendor-{ID} a sc:Vendor .
```

Moreover, the following is another example for $p(t_1(\mathbf{x}_1), t_2(\mathbf{x}_2))$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are (*ProviderID*) and (*ProductID*) respectively.

```
Example mapping 2:
source SELECT ProviderID, ProductID FROM Provider-Product
target data:product-{ProductID}
   sc:seller data:vendor-{ProviderID}.
```

Given a set $\mathcal{M}$ and a database instance $\mathcal{D}$, the *virtual ABox* $\mathcal{M}_{\mathcal{D}}$ *exposed by* $\mathcal{D}$ *through* $\mathcal{M}$ is the set:

$$\{L[\mathbf{x} \mapsto \mathbf{o}] \mid (\mathbf{x} \mapsto \mathbf{o}) \in Q(\mathbf{x})^{\mathcal{D}}, (s{:}Q(\mathbf{x}), t{:}\mathbf{L}) \in \mathcal{M}, L \in \mathbf{L}\}$$

where $L[\mathbf{x} \mapsto \mathbf{o}]$ denotes the *ground* atom obtained after replacing attributes in $\mathbf{x}$ with their corresponding values in $\mathbf{o}$. For instance, given an atom $C(t(x_1, x_2))$ and an answer $((x_1, x_2) \mapsto (o_1, o_2))$, $C(t(x_1, x_2))[(x_1, x_2) \mapsto (o_1, o_2)]$ denotes the ground atom $C(t(o_1, o_2))$.

Assuming the SQL query from *Example 1* retrieves the solution mappings $(ID \mapsto 1)$ and $(ID \mapsto 2)$, the resulting RDF graph will be as follows:

```
data:vendor-1 a sc:Vendor . data:vendor-2 a sc:Vendor .
```

3.2 Mapping Patterns

Calvanese *et al.* [2023] analyzed many common scenarios of VKGs, and summarized the results as *mapping patterns*. These mapping patterns encapsulate a distilled set of best practices for creating mappings and designing ontology structures. We adopt four of the most common *schema-driven* patterns: *(1)* Schema Entity (SE), *(2)* Schema Relationship (SR), *(3)* Schema Relationship with Merging (SRm), and *(4)* Schema Hierarchy (SH). The *schema components* of these mapping patterns are illustrated in Figure 2.

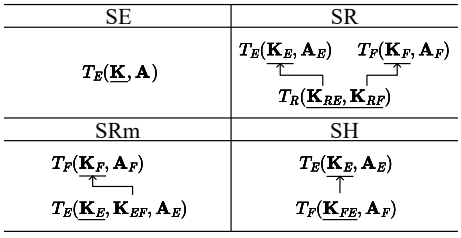

Figure 2: The DB schema structures of SE, SR, SRm, and SH. T_* denotes table names, $\underline{\mathbf{K}_*}$ denotes primary keys, $\mathbf{K}_*/\mathbf{A}_*$ denotes other attributes, and the arrow $\leftarrow$ denotes foreign key constraints.

For each schema structure in Figure 2, mapping patterns specify corresponding mapping and ontological axioms. For example, the mapping corresponding to SR is:

$$(s{:}T_R,\ \ t{:}p_R\left(t_{C_E}\left(\mathbf{K}_{RE}\right), t_{C_F}\left(\mathbf{K}_{RF}\right)\right)).$$

Example in Figure 1 can be seen as an instantiation of pattern SR, which is identical to Example mapping 2, where C_E is *sc:Vendor*, C_F is *sc:Product*, and p_R is *sc:seller*.

3.3 Task Definition

The VKG construction task can be seen as a function, which takes a DB schema Σ and an initial ontology $\mathcal{T}_0$ as inputs, and produces a VKG specification $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \Sigma)$ with $\mathcal{T}_0 \subseteq \mathcal{T}$, such that $\mathcal{T}$ captures the domain of interest, and $\mathcal{M}$ *correctly* establishes the relation between $\mathcal{T}$ and Σ. Typically, the quality of $\mathcal{P}$ needs to be assessed by either human experts or benchmarks, e.g., by verifying that queries answerable in the source database remain answerable in the VKG.

4 Methodology

In this work, we introduce LLM4VKG, an LLM-based framework designed to support VKG construction by automating tasks that typically require human judgment, such as concept matching and naming. Furthermore, we utilize several mapping patterns to refine and guide the VKG construction process, ensuring a well-structured output.

The overall framework of LLM4VKG is illustrated in Figure 3. The framework comprises two main steps: *(1) Mapping Pattern Recognition*, which extracts mapping pattern instances from the DB schema using SPARQL queries, and *(2) Ontology Completion & Mapping Bootstrapping*, which leverages LLMs and mapping pattern instances to complete the input ontology and generate mappings.

4.1 Mapping Pattern Recognition

To recognize the occurrences of mapping patterns in a given database schema Σ, we first convert Σ into a *DB graph* G_Σ in a straightforward way and then, for each mapping pattern, we design a dedicated SPARQL query to match the corresponding graph pattern in G_Σ.

Due to space constraints, we skip the details of the definition of G_Σ, which is provided in Appendix A. Instead, for our running example in Figure 1, we visualize the DB graph in Figure 4. Then, we utilize SPARQL queries to encode the structures depicted in Figure 2. An example query for the SR pattern is:

```
SELECT DISTINCT ?T_E ?K_E ?K_RE ?T_R ?K_RF ?K_F ?T_F {
  ?T_E :hasPrimaryKeyColumn ?K_E .
  ?T_F :hasPrimaryKeyColumn ?K_F .
  ?T_R :hasPrimaryKeyColumn ?K_RE, ?K_RF .
  ?K_RE :foreignKeyReferences ?K_E .
  ?K_RF :foreignKeyReferences ?K_F .
}
```

Note that this query deals with the case in which both T_E and T_F have primary keys of single columns, but can be easily generalized to keys with multiple columns. Finally, these SPARQL queries are executed on G_Σ to derive the recognized mapping pattern instances.

4.2 LLM Modules

In LLM4VKG, as depicted in Figure 3, the key component is "Element Level DB-to-Ontology Alignment", which attempts to match a given DB element to an ontology element. If no suitable match is found, it generates a new ontology element representing the DB element. It consists of three sequential modules[2] to leverage (possibly different) LLMs:

- *Retriever* takes an element of the schema (e.g., *Provider* for a table name, *Provider.Tag* for a column name) and a set of ontology elements $\mathbf{E}_{on} \subseteq \text{NC} \cup \text{NP} \cup \text{ND}$ as inputs. This module uses a *pre-trained sentence similarity language model*[3] *MR* to retrieve top-n candidate elements $\mathbf{E}'_{on}$, where n is a hyperparameter. The process is expressed as: $\mathbf{E}'_{on} = \text{Retriever}^n_{MR}(e_{db}, \mathbf{E}_{on})$.
- Similar to Retriever, *Matcher* takes a DB schema element e_{db} and a set of ontology elements $\mathbf{E}_{on}$ as inputs. However, Matcher uses a *generative language model*[4] *MM* to generate a pair (d, e^m_{on}) where d is a label denoting the matching degree (with possible values High, Medium, or Low) and e^m_{on} is an element from $\mathbf{E}_{on}$ matched to e_{db}. When d is Low, e^m_{on} is a null value, indicating that no element in $\mathbf{E}_{on}$ can be matched. The process is expressed as: $(d, e^m_{on}) = \text{Matcher}_{MM}(e_{db}, \mathbf{E}_{on})$.
- *Namer* takes as inputs an element $e_1 \in \Sigma$ and optionally $e_2 \in \Sigma \cup \text{NC} \cup \text{NP} \cup \text{ND}$, and uses a generative language model *MN* to generate a class or property name according to the inputs. The generation uses the ontology as the context to make sure the names are meaningful. When only e_1 is provided, Namer generates a fresh class or property name in the ontology based on the name of e_1. If two elements are provided and $e_2 \in \text{NC} \cup \text{NP} \cup \text{ND}$, Namer generates a fresh sub class or property of e_2, and names it based on the name of e_1; if $e_2 \in \Sigma$, Namer generates a fresh *object property* and names it according to the pair (e_1, e_2). The process is expressed as: $\text{Namer}_{MN}(e_1, e_2)$.

Element Level DB-to-Ontology Alignment Given a DB schema element e_{db} and a set of ontology elements $\mathbf{E}_{on}$, we first try to match e_{db} to $\mathbf{E}_{on}$ directly, using Retriever and Matcher: $(d, e^m_{on}) = \text{Matcher}_{MM}(e_{db}, \text{Retriever}^n_{MR}(e_{db}, \mathbf{E}_{on}))$. If d is High, e^m_{on} will be the matched ontology element. Otherwise, Namer will be used to generate a new element (a class or property) of the ontology and name it based on e_{db}:

$$\text{DB2Ont}(e_{db}, \mathbf{E}_{on}) = \begin{cases} e^m_{on} & d = \text{High} \\ \text{Namer}_{MN}(e_{db}, e^m_{on}) & d = \text{Medium} \\ \text{Namer}_{MN}(e_{db}) & d = \text{Low}, \\ & \mathbf{E}_{on} \subseteq \text{NC} \cup \text{ND} \\ \text{Namer}_{MN}(e^1_{db}, e^2_{db}) & d = \text{Low}, \\ & \mathbf{E}_{on} \subseteq \text{NP} \end{cases}$$

In the last case, Namer generates a name as an object property based on the pair (e^1_{db}, e^2_{db}), which will be used to deal with the mapping patterns SR ($e_{db} = T_R$) and SRm ($e_{db} = \mathbf{K}_{EF}$), and for both patterns $e^1_{db} = T_E, e^2_{db} = T_F$. For example, in Figure 1, e_{db} is *Provider-Product*, e^1_{db} is *Provider*, and e^2_{db} is *Product*.

4.3 Ontology Completion & Mapping Bootstrapping

Now we present how to perform ontology completion and mapping bootstrapping using the above LLM modules over

[2]The detailed prompts for the modules are in Appendix B.

[3]See sentence similarity models in this URL: https://huggingface.co/models?pipeline_tag=sentence-similarity

[4]See generative models in this URL: https://huggingface.co/models?pipeline_tag=text-generation

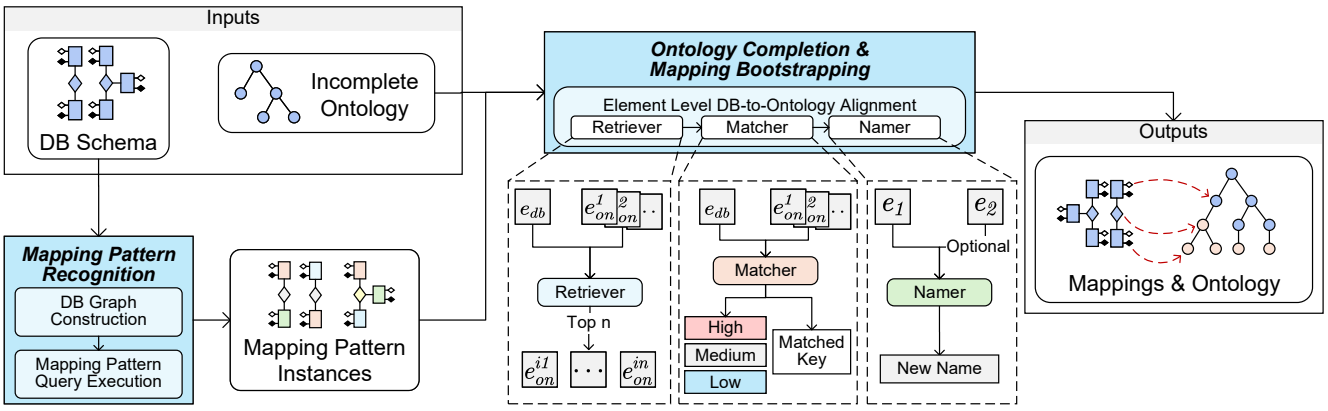

Figure 3: An overview of the LLM4VKG framework. The azure background represents the submodules of the LLM4VKG.

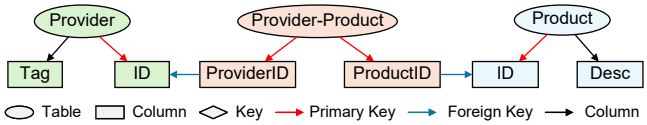

Figure 4: An example of the DB graph.

the mapping pattern instances. To avoid potential overlaps in mapping patterns, we process them in the specific sequence of **SH**, **SR**, **SRm**, and **SE**.

SH This mapping pattern describes a hierarchical relationship between two classes. Following the pattern, T_E and T_F are matched to class names:

$$C_E = \text{DB2Ont}(T_E, \text{NC}) \qquad C_F = \text{DB2Ont}(T_F, \text{NC})$$

and a sub-class axiom $C_F \sqsubseteq C_E$ is added to the ontology. Note that no mappings are generated for this pattern.

SR This mapping pattern describes an object property from T_E to T_F, via T_R. Following the pattern, T_R is matched to an object property name:

$$p_R = \text{DB2Ont}(T_R, \text{NP})$$

and mappings are generated as follows:

$$(s{:}T_R, t{:}p_R\left(t_{C_E}\left(\mathbf{K}_{RE}\right), t_{C_F}\left(\mathbf{K}_{RF}\right)\right))$$

where the name of the topmost ancestor of C_E or C_F is used as the template for t_{C_E} or t_{C_F}. For example, if C_E is *Student* and its topmost ancestor is *Person*, $\mathbf{K}_{RE}$ is *sid*, then t_{C_E} takes the form *:Person/*{*sid*}.

SRm This mapping pattern describes an object property from T_E to T_F, via $\mathbf{K}_{EF}$. Therefore, $\mathbf{K}_{EF}$ is matched to an object property name:

$$p_{EF} = \text{DB2Ont}(\mathbf{K}_{EF}, \text{NP})$$

If $\mathbf{K}_{EF}$ consists of multiple columns, the names of these columns are concatenated using the word "and" (e.g., ID and Tag) to preserve their joint semantic significance. This unified string will serve as one DB schema element during the matching process. Mappings are generated as follows:

$$(s{:}T_E, t{:}p_{EF}\left(t_{C_E}\left(\mathbf{K}_E\right), t_{C_F}\left(\mathbf{K}_{EF}\right)\right))$$

SE This mapping pattern specifies the correspondence between a table representing an *entity*[5], characterized by a primary identifier and its attributes, and a class in the ontology with associated data properties.

If T_E or an attribute $a \in \mathbf{K} \cup \mathbf{A}$ corresponds to an object property, the corresponding table and column are excluded from the process. Let $\mathbf{A}_D$ denote the set of columns that do not match the object properties. Then, T_E is matched to a class name, and each $a \in \mathbf{A}_D$ is matched to a data property name:

$$d_a = \text{DB2Ont}(a, \text{ND}) \qquad C_E = \text{DB2Ont}(T_E, \text{NC})$$

Mappings are generated as follows:

$$(s{:}T_E\,, t{:}C_E(t_E(\mathbf{K})), \{d_a(t_E(\mathbf{K}), a)\}_{a\in\mathbf{A}_D})$$

5 Experimental Setups

Datasets We evaluate LLM4VKG on RODI [Pinkel *et al.*, 2018] and RODI-T (x%), a variant of RODI in which x% of the ontology vocabulary is removed from the ontology starting from the leaf nodes. A RODI sample comprises three main elements: a database schema, a golden ontology, and a set of query pairs. Each query pair includes an SQL query and a SPARQL query, both of which are formulated to answer the same question. The output of the SQL query is treated as the label, whereas the output of the SPARQL query, based on the golden ontology, is regarded as the prediction. The domain and statistical details of the RODI dataset are presented in Table 1.

Domain	# Samp	# Tbl	# Col	# FK	# QP	# NC	# NP	# ND
Conference	26	38.3	93.4	51.2	29.8	55.5	39.3	17.0
Geodata	5	154.6	290.8	64.4	49.4	23.0	44.0	27.0
Oil & Gas	2	70.0	962.0	78.0	228.0	378.5	148.0	237.0

Table 1: Statistics of the RODI dataset across three domains, showing the average number of samples, tables, columns, foreign keys, query pairs, classes, object properties, and data properties.

[5]As in *Entity-Relationship* [Chen, 1976] diagrams.

Evaluation Process Some mapping generation systems produce mappings and an ontology based solely on the DB schema, leading to a vocabulary that differs from the golden ontology in RODI. To evaluate the quality of the VKG, ontology matching between the generated ontology and the RODI golden ontology is necessary. The evaluation includes the following steps: *(1)* Match the ontology generated by the VKG construction system to the golden ontology in the sample using LogMap [Jiménez-Ruiz and Grau, 2011], and add the matching names (i.e., names only, without structure) from the golden ontology as equivalent classes or properties in the generated ontology. *(2)* Construct the VKG based on the database schema, the generated ontology, and mappings. *(3)* Compare the results of each query pair of the SQL query and the corresponding SPARQL query.

Metrics To measure the quality of generated mappings automatically, we follow Pinkel *et al.* [2018] and use the F1 metric computed by comparing the reference set (by SQL query) and test set (by SPARQL query). Let *ref* be the results from the SQL query, and *res* be the results from the SPARQL query, then the precision $P = 1 - \frac{|res \setminus ref|}{|res|}$, the recall $R = 1 - \frac{|ref \setminus res|}{|ref|}$ and $F1 = \frac{2 \times \mathrm{P} \times \mathrm{R}}{\mathrm{P}+\mathrm{R}}$, where $ref \setminus res$ denotes the elements in *ref* but not in *res* (and vice-versa).

Comparative Baselines We consider baselines from two categories: *(1)* Traditional mapping generation systems, including state-of-the-art systems like BootOX and IncMap, as well as those used in the RODI paper [Pinkel *et al.*, 2018] (Tradition); and *(2)* direct application of LLMs for mapping generation, provided the ontology and database schema (Vanilla LLM).

- Tradition: (a) COMA [Aumueller *et al.*, 2005], IncMap [Pinkel *et al.*, 2013], and BootOX [Jiménez-Ruiz *et al.*, 2015] leverage both DB schema and Ontology. (b) D2RQ [Bizer and Seaborne, 2004], MIRROR [de Medeiros *et al.*, 2015], and ontop [Calvanese *et al.*, 2017] rely solely on DB schema.
- Vanilla LLM: *o1* [OpenAI, 2024b], *GPT-4o* [OpenAI, 2024a], and *DeepSeek-V3* [DeepSeek-AI, 2024]. Note that the generated mappings often contain errors that we fix through manual verification and correction (See Appendix C for details).

Implementation Details In this study, we leverage the VKG system Ontop [Calvanese *et al.*, 2017][6] to initialize the generated VKG based on a database connection, an ontology, and a set of mappings. For Retriever, we use *bge-m3* [Chen *et al.*, 2024] as the backbone model. For Matcher and Namer, we incorporate *GPT-4o*, *GPT-4o-mini* [OpenAI, 2024a], and *Qwen2.5-7b* [Qwen, 2024] as backbone models, representing various levels of performance across LLMs.

6 Results & Analysis

Results on RODI To evaluate the effectiveness of LLM4VKG, we conduct experiments using the scenarios of RODI, where the model takes the golden ontology as input, treats it as incomplete, and actively completes it. The results are summarized in Table 2 and Table 3.

Table 2 reports that LLM4VKG outperforms traditional methods in all scenarios, achieving SOTA performance. The most significant improvement is observed in the scenario "Conference - adjusted naming", which achieves an absolute increase of +39%.

Furthermore, the results presented in Table 3 indicate that, modulo manual verification and correction, both *o1* and *DeepSeek-V3* achieve performance levels comparable to the SOTA mapping generation systems BootOX and IncMap. These findings highlight the potential of using LLMs directly to produce effective mappings.

Finally, LLM4VKG achieves substantial improvements over baseline methods, even when employing *Qwen2.7-7b*, a small-scale open-source LLM, as its backbone model. Specifically, compared to BootOX, LLM4VKG achieves improvements of +17%, +14%, and +8% in average F1 scores on *GPT-4o*, *GPT-4o-mini*, and *Qwen2.5-7b*, respectively.

Scenario	D2RQ	MIRR.	ontop	COMA	IncM.	B.OX	**ours**
Conference domain, adjusted naming							
CMT	0.31	0.28	0.28	0.48	0.45	0.76	**0.86**
Conference	0.26	0.27	0.26	0.36	0.53	0.51	**0.92**
SIGKDD	0.38	0.30	0.38	0.66	0.76	0.86	**0.93**
Conference domain, restructured							
CMT	0.14	0.17	0.14	0.38	0.44	0.41	**0.55**
Conference	0.21	0.23	0.13	0.31	0.41	0.41	**0.59**
SIGKDD	0.28	0.11	0.21	0.41	0.38	0.52	**0.71**
Conference domain, combined case							
SIGKDD	0.21	0.11	0.21	0.28	0.38	0.48	**0.72**
Conference domain, missing FKs							
Conference	0.18	0.17	-	0.21	0.41	0.33	**0.51**
Conference domain, denormalized							
CMT	0.20	0.22	0.20	-	0.40	0.44	**0.52**
Geodata domain							
Classic Rel.	0.06	-	-	-	0.08	0.13	**0.18**
Oil & gas domain							
Atomic	0.08	0.00	0.10	0.02	0.12	0.14	**0.19**

Table 2: Fine-grained results of RODI using ***GPT-4o*** as the LLM for the Matcher and Namer, with F1-scores reported . For each scenario, the best number is emphasized in **bold**, and the second-best number is distinguished by underlines.

Results on RODI-T(x%) Figure 5 shows that LLM4VKG achieves an average F1 score of 0.46 even with 25% of the vocabulary of the ontology removed, closely matching the performance of BootOX. This highlights the ability of LLM4VKG to infer naming rules and reconstruct missing

	Tradition		Vanilla LLM			LLM4VKG(ours)		
	B.OX	IncM.	DS-V3	o1	G4o	G4o	G4o-m	Q-7b
AN	0.71	0.58	0.53	0.37	0.09	**0.91**	0.85	0.87
Res	0.45	0.41	0.43	0.46	0.12	**0.61**	**0.61**	0.50
CC	0.48	0.38	0.31	0.51	0.14	**0.72**	0.69	0.68
MF	0.33	0.41	0.43	0.38	0.15	**0.51**	0.44	0.43
Den	0.44	0.40	0.44	0.48	0.16	**0.52**	0.52	0.34
avg.	0.48	0.44	0.43	0.44	0.13	**0.65**	0.62	0.56

Table 3: Main results: Average F1 scores of RODI across various scenarios. Abbreviations: AN = Adjusted Naming, Res = Restructured, CC = Combined Case, MF = Missing Foreign Keys, Den = Denormalized, avg. = Average, G4o = *GPT-4o*, G4o-m = *GPT-4o-mini*, DS-V3 = *DeepSeek-V3*, Q-7b=*Qwen2.5-7b*.

[6]https://ontop-vkg.org/

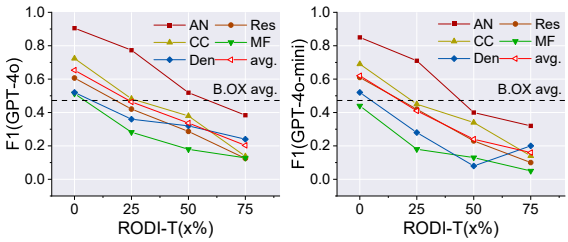

Figure 5: Results of RODI-T(x%) for LLM4VKG.

ontology parts from existing elements. As more vocabulary is removed, performance declines partly because ontology alignment during evaluation may incorrectly flag correctly structured outputs if they don't precisely match the reference ontology. Consequently, the final performance also depends on the alignment method used. Nonetheless, LLM4VKG remains robust with partial vocabularies, demonstrating its effectiveness in handling incomplete ontologies.

Ablation Experiments To evaluate the contribution of individual components to the performance of LLM4VKG, we perform ablation experiments and present the results in Table 4: (1) *-Matcher* disables the Matcher component and directly selects the top-1 result from the Retriever as the matched key, assigning a matching degree of High as the matched output; (2) *-SH*, *-SR*, *-SRm*, and *-SE* indicate the removal of the corresponding mapping pattern handling processes during the mapping generation process. The results indicate that removing the Matcher component (*-Matcher*) decreases the average F1 score from 0.65 to 0.53. This reduction reflects that relying only on text similarity is insufficient for accurately mapping the database to the ontology, emphasizing the need for LLMs' reasoning capabilities. Similarly, eliminating the SE component (*-SE*) results in a significant drop to an average F1 score of 0.16, highlighting its essential role in the mapping process. Other components, such as SH, SR, and SRm, also contribute to performance, with their removal leading to moderate decreases in the average F1 score. These ablation results underscore the necessity of each component in maintaining the robustness and performance of LLM4VKG.

Methods	AN	Res	CC	MF	Den	avg
ours	0.91	0.61	0.72	0.51	0.52	0.65
- Matcher	0.76	0.53	0.66	0.35	0.33	0.53
- SH	0.64	0.51	0.62	0.46	0.40	0.52
- SR	0.84	0.49	0.48	0.49	0.52	0.56
- SRm	0.84	0.53	0.65	0.49	0.52	0.61
- SE	0.19	0.26	0.28	0.00	0.08	0.16

Table 4: F1 scores of ablation experiments for the components of LLM4VKG.

Fine-Grained Analysis for Query Types Table 5 presents the F1 score of LLM4VKG and baseline models across different query types. Overall, LLM4VKG consistently achieves a higher F1 score than BootOX across all query types, highlighting its effectiveness in managing diverse and intricate ontology queries. Among the three query types (C, D, O), object property queries (O) are the most challenging because multiple tables and columns are involved. LLM4VKG significantly outperforms BootOX, the best system, in this category (0.59 vs. 0.05), demonstrating its superior ability to handle better on relationships between different classes.

Additionally, while BootOX performs well on 1:1 mappings (0.82) but fails on n:1 mappings (0.00), LLM4VKG achieves 0.93 for 1:1 and 0.36 for n:1, showcasing its capability to manage more complex mappings.

Scnario	LLM4VKG			BootOX
	GPT-4o	GPT-4o-mini	Qwen2.5-7b	
C	0.71	0.70	0.47	0.53
D	0.69	0.63	0.47	0.50
O	0.59	0.61	0.46	0.05
1:1	0.93	0.92	0.80	0.82
n:1	0.36	0.33	0.26	0.00

Table 5: F1 scores of the queries on different match types. "C" represents queries on classes, "D" on data properties, and "O" on object properties. "1:1" and "n:1" stands for queries involving 1:1 or n:1 mappings among classes and tables, respectively.

RODI-T Error Analysis Figure 6 shows that as more vocabulary is removed in RODI-T(x%), the F1 score for classes (C), data properties (D), and object properties (O) decreases. The decline is smallest for classes, followed by data properties, with object properties experiencing the most significant drop. This is because classes and data properties involve fewer elements, making it easier to generate names that are similar to those in the golden ontology. In contrast, object properties are more complex, involving multiple tables and columns, which makes it harder to reproduce the exact properties in the golden ontology and leads to more errors.

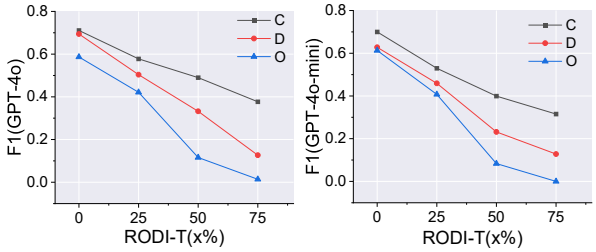

Figure 6: Fine-grained results of RODI-T(x%) for LLM4VKG.

7 Conclusion & Future Work

In this work, we propose LLM4VKG, a framework that leverages LLMs for VKG construction. LLM4VKG automatically builds mappings and enriches the initial ontology based on mapping patterns extracted from the DB schema. Additionally, we developed an automated evaluation framework to simplify the assessment process. Our experiments demonstrate that LLM4VKG is robust and efficient in handling incomplete ontologies and complex mapping tasks. In future work, we aim to *(1)* expand the framework with additional mapping patterns; *(2)* in addition to the DB schema, utilize the statistics of the data in the DB to enhance ontology construction; and *(3)* develop a more comprehensive dataset based on RODI to further improve and validate our approach.

Acknowledgments

This work is partially supported by National Nature Science Foundation of China under No. 62476058, by HEU project CyclOps (GA n. 101135513), by the Province of Bolzano and FWF through project OnTeGra (DOI 10.55776/PIN8884924), by the Province of Bolzano and EU through projects ERDF-FESR 1078 CRIMA, and ERDF-FESR 1047 AI-Lab, by MUR through PRIN project 2022XERWK9 S-PIC4CHU, and by the EU and MUR through PNRR project PE0000013-FAIR. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations in this paper. This work has been carried out while Marco Di Panfilo was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome in collaboration with Free University of Bozen-Bolzano.

Contribution Statement

†Guohui Xiao and Lin Ren contribute equally to this work. *Corresponding author: guohui.xiao@seu.edu.cn.

References

[Anthropic, 2024] Anthropic. Claude 3.5 sonnet. *Anthropic blog*, 2024.

[Aumueller *et al.*, 2005] D. Aumueller, H. Hai Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. In *Proc. of SIGMOD'2005*, pages 906–908, 2005.

[Azaria *et al.*, 2024] A. Azaria, R. Azoulay, and S. Reches. Chatgpt is a remarkable tool—for experts. *Data Intelligence*, 6(1):240–296, 2024.

[Bizer and Seaborne, 2004] C. Bizer and A. Seaborne. D2RQ-treating non-RDF databases as virtual RDF graphs. In *Proc. of ISWC'2004*, volume 2004. Springer Hiroshima, 2004.

[Calvanese *et al.*, 2017] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3):471–487, 2017.

[Calvanese *et al.*, 2023] D. Calvanese, A. Gal, D. Lanti, M. Montali, A. Mosca, and R. Shraga. Conceptually-grounded mapping patterns for virtual knowledge graphs. *Data & Knowledge Engineering*, 145:102157, 2023.

[Chen *et al.*, 2024] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu. BGE M3-embedding: Multilingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

[Chen, 1976] Peter P. Chen. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, March 1976.

[Chen, 2024] H. Chen. Large knowledge model: Perspectives and challenges. *Data Intelligence*, 6(3):587–620, 2024.

[Das *et al.*, 2012] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. W3C Recommendation, World Wide Web Consortium, September 2012. Available at http://www.w3.org/TR/r2rml/.

[de Medeiros *et al.*, 2015] L. F. de Medeiros, F. Priyatna, and Ó. Corcho. MIRROR: automatic R2RML mapping generation from relational databases. In *Proc. of ICWE'2015*, volume 9114, pages 326–343. Springer, 2015.

[DeepSeek-AI, 2024] DeepSeek-AI. Deepseek-V3 technical report, 2024.

[Giglou *et al.*, 2024a] H. B. Giglou, J. D'Souza, and S. Auer, editors. *LLMs4OL 2024: The 1st Large Language Models for Ontology Learning Challenge*, volume 4 of *Open Conference Proceedings*. TIB Open Publishing, 2024.

[Giglou *et al.*, 2024b] H. B. Giglou, J. D'Souza, F. Engel, and S. Auer. LLMs4OM: Matching ontologies with large language models. *CoRR*, abs/2404.10317, 2024.

[Gupta *et al.*, 2012] S. Gupta, P. Szekely, C. A. Knoblock, A. Goel, M. Taheriyan, and M. Muslea. Karma: A system for mapping structured sources into the semantic web. In *Proc. of the 9th Extended Semantic Web Conf. (ESWC)*, pages 430–434. Springer, 2012.

[Hertling and Paulheim, 2023] S. Hertling and H. Paulheim. Olala: Ontology matching with large language models. In *Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP 2023, Pensacola, FL, USA, December 5-7, 2023*, pages 131–139. ACM, 2023.

[Jiménez-Ruiz and Grau, 2011] E. Jiménez-Ruiz and B. Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[Jiménez-Ruiz *et al.*, 2015] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, C. Pinkel, M. G. Skjæveland, E. Thorstensen, and J. Mora. BootOX: Practical mapping of RDBs to OWL 2. In *Proc. of ISWC'2015*, pages 113–132. Springer, 2015.

[Kommineni *et al.*, 2024] V. K. Kommineni, B. König-Ries, and S. Samuel. From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. *CoRR*, abs/2403.08345, 2024.

[Li *et al.*, 2024] G. Li, C. Tang, L. Chen, D. Deguchi, T. Yamashita, and A. Shimada. LLM-Driven ontology learning to augment student performance analysis in higher education. In *Knowledge Science, Engineering and Management - 17th International Conference, KSEM 2024, Birmingham, UK, August 16-18, 2024, Proceedings, Part III*, volume 14886 of *Lecture Notes in Computer Science*, pages 57–68. Springer, 2024.

[Lo *et al.*, 2024] A. Lo, A. Q. Jiang, W. Li, and M. Jamnik. End-to-End ontology learning with large language models. *CoRR*, abs/2410.23584, 2024.

[Mogotlane and Dombeu, 2016] K. Mogotlane and J. V. Fonou Dombeu. Automatic conversion of relational databases into ontologies : A comparative analysis

of protege plug-ins performances. *International journal of Web & Semantic Technology*, 7:21–40, 10 2016.

[Mosca *et al.*, 2023] R. Mosca, M. De Santo, and R. Gaeta. Ontology learning from relational database: a review. *J. Ambient Intell. Humaniz. Comput.*, 14(12):16841–16851, 2023.

[Motik *et al.*, 2012] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 web ontology language: Profiles. W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/owl2-profiles/, 2012.

[OpenAI, 2024a] OpenAI. Hello gpt-4o. *OpenAI blog*, 2024.

[OpenAI, 2024b] OpenAI. Introducing openai o1-preview. *OpenAI blog*, 2024.

[Pinkel *et al.*, 2013] C. Pinkel, C. Binnig, E. Kharlamov, and P. Haase. IncMap: pay as you go matching of relational schemata to OWL ontologies. In *OM*, pages 37–48, 2013.

[Pinkel *et al.*, 2018] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, E. Kharlamov, W. May, A. Nikolov, A. S. Bastinos, M. G. Skjæveland, A. Solimando, M. Taheriyan, C. Heupel, and I. Horrocks. RODI: Benchmarking relational-to-ontology mapping generation quality. *Semantic Web*, 9(1):25–52, 2018.

[Poggi *et al.*, 2008] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, 10:133–173, 2008.

[Qwen, 2024] Team Qwen. Qwen2.5: A party of foundation models, September 2024.

[Sequeda and Miranker, 2015] Juan F. Sequeda and Daniel P. Miranker. Ultrawrap mapper: A semi-automatic relational database to RDF (RDB2RDF) mapping tool. In Serena Villata, Jeff Z. Pan, and Mauro Dragoni, editors, *ISWC (Posters & Demos)*, volume 1486 of *CEUR Electronic Workshop Proceedings*. CEUR-WS.org, 2015.

[Sequeda *et al.*, 2023] J. Sequeda, D. Allemang, and B. Jacob. A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise sql databases, 2023.

[Teymurova *et al.*, 2024] S. Teymurova, E. Jiménez-Ruiz, T. Weyde, and J. Chen. OWL2Vec4OA: Tailoring knowledge graph embeddings for ontology alignment. *CoRR*, abs/2408.06310, 2024.

[Xiao *et al.*, 2019] G. Xiao, L. Ding, B. Cogrel, and D. Calvanese. Virtual knowledge graphs: An overview of systems and use cases. *Data Intelligence*, 1(3):201–223, 2019.