NaVQA: Mitigating Silent Failures in Question Answering over Virtual Knowledge Graph

Guohui Xiao[1,2], **Haohan Xue**[1,2], **Lin Ren**[1,2], **Yishuai Geng**[1,2], **Guilin Qi**[1,2], **Shenyu Zhang**[1,2], **Dehao Guo**[1,2], **Marco Di Panfilo**[3], **Davide Lanti**[3], **Linfang Ding**[4,*]
[1]School of Computer Science and Engineering, Southeast University, Nanjing, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
[3]Free University of Bozen-Bolzano, Italy
[4]Jiangsu Key Laboratory of Soil and Water Processes in Watershed, College of Geography and Remote Sensing, Hohai University, Nanjing, China
{guohui.xiao, thex1ay, renlin, ysgeng, gqi, shenyu.zhang, guodehao}@seu.edu.cn, {mdipanfilo, davide.lanti}@unibz.it, linfang.ding@hhu.edu.cn

Abstract

Virtual Knowledge Graphs (VKGs) provide unified access to legacy relational data sources through a high-level ontology modeling a domain of interest. The content of the ontology elements (classes and properties) is virtually mapped to underlying data sources through declarative mappings. The standard approach to interacting with a VKG system is to use SPARQL as the query language. However, the complexity of SPARQL creates a high entry barrier for normal users. In this paper, we study the VKG-QA task, which enables users to interact with the VKGs through a natural language (NL) interface by translating their questions into SPARQL queries. One critical challenge in such translation is *silent failures*, where a syntactically correct SPARQL query returns empty results because it involves ontology elements that lack mappings to the underlying data. To address this, we propose NaVQA (Navigation-based VKG Question Answering), a framework leveraging Large Language Models that (1) identifies and indexes the *active* ontology elements based on the mapping specifications, and (2) iteratively constructs a query graph from the NL question using the active ontology elements and their relations to synthesize SPARQL. Experiments on three real-world VKGs show that NaVQA significantly mitigates silent failures and outperforms existing baselines.

1 Introduction

The Semantic Web vision promises a unified interface for accessing heterogeneous data. Virtual Knowledge Graphs (VKGs) realize this by mapping [**?**] legacy Relational Databases (RDBs) to high-level ontology [Lenzerini, 2011; Xiao *et al.*, 2019]. By exploiting data virtualization, VKGs avoid physical materialization to integrate data silos. However, since VKG systems use SPARQL as the standard interface for data retrieval [Rodriguez-Muro *et al.*, 2013], the complexity of such formal languages creates a high barrier, preventing users from directly utilizing these systems. Consequently, Question Answering over Virtual Knowledge Graphs (VKG-QA) has emerged as a critical task, enabling users to access VKGs through natural language.

	Schema Level		*Data Level*	
Task	Ontology	Mapping	Database	Triple
Text-to-SQL			✓	
KGQA	✓			✓
VKG-QA	✓	✓		

Table 1: Comparison of component availability across Text-to-SQL, KGQA, and VKG-QA.

However, implementing robust VKG-QA is hindered by the unique characteristics of the virtual environment. As demonstrated in Table 1, related paradigms depend heavily on components at the data level (i.e., database and triples) [Sequeda *et al.*, 2024]. Text-to-SQL approaches cannot be applied in VKG-QA, moreover, direct SQL access to the underlying data source is forbidden in the VKG setting. Similarly, applying traditional Knowledge Graph Question Answering (KGQA) paradigms to VKG settings presents distinct challenges. Retrieval-based approaches typically navigate materialized knowledge graphs to identify answer subgraphs [Sun *et al.*, 2024a; Luo *et al.*, 2024]. However, these methods are fundamentally incompatible with VKG settings because VKGs lack materialized instances. Without physical triples to traverse, graph-based retrieval algorithms simply cannot function. Consequently, semantic parsing approaches (specifically Text-to-SPARQL [Luz and Finger, 2018; Banerjee *et al.*, 2023]) become the viable alternative. By translating natural language directly into executable SPARQL queries, this

*Corresponding author: linfang.ding@hhu.edu.cn.

paradigm bypasses the need for instance traversal and aligns well with the query-based nature of VKG systems. Recently, Large Language Models (LLMs) have demonstrated exceptional performance in Text-to-SPARQL tasks [D'Abramo *et al.*, 2025; Meloni *et al.*, 2025; Pan *et al.*, 2025]. However, applying them to VKGs poses a critical challenge: the mismatch between ontology coverage and actual mappings to the underlying databases. In practice, when creating VKGs, ontologies are often built by importing standard ontologies with domain-specific additions [Ding *et al.*, 2020]. Hence, typically only a small fraction of the ontology elements are populated via mappings by the underlying databases. Unaware of mappings, LLMs tend to formulate queries using the unmapped elements based on their semantic relevance. This leads to *silent failures*—where queries are syntactically correct but return empty results, deceptively implying that no data exists. In this work, we propose *NaVQA*, an end-to-end framework that leverages LLMs for VKG-QA aimed at overcoming the aforementioned shortcomings. The framework operates through two key stages: *(1)* identifies *active* ontology elements based on the mappings and constructs efficient indices using LLMs for their retrieval; *(2)* iteratively builds a graph structure based on the NL question using the indexed active ontology elements and their relations to synthesize the final SPARQL query. In summary, our main contributions include: *(1)* we propose NaVQA, an end-to-end framework for VKG-QA that leverages LLMs and mapping specifications while effectively mitigating silent failures; *(2)* we curate three high-quality datasets derived from real-world VKGs, providing diverse and challenging scenarios to evaluate; *(3)* we show through extensive evaluation that NaVQA significantly outperforms existing baselines. All code and data are available at https://github.com/TheX1ay/NaVQA.

2 Preliminaries

2.1 Virtual Knowledge Graphs

A *VKG specification* is a triple $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \Sigma)$, where $\mathcal{T}$ denotes an *ontology*, $\mathcal{M}$ a set of *mappings*, and Σ a *database schema* [Xiao *et al.*, 2018].

Database schema (Σ). 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. We use the notation $Q(x)$ to denote a SQL query Q of *answer attributes* x. Given a *database instance* $\mathcal{D}$ of Σ, the *evaluation* $Q(x)^{\mathcal{D}}$ *of* $Q(x)$ *over* $\mathcal{D}$ is a *set* (ignoring duplicates) of *answers* $(x \mapsto o)$ mapping attributes in x to values in $\mathcal{D}$.

Ontology ($\mathcal{T}$). The ontology $\mathcal{T}$ provides a high-level conceptual view of the domain. An ontology $\mathcal{T}$ is a set of axioms in some Description Logic (DL) language (usually DL-Lite$_{\mathcal{R}}$ [**?**], 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 contain only class and property names that actually occur in $\mathcal{T}$.

Mappings ($\mathcal{M}$). The mapping set $\mathcal{M}$ bridges the semantic gap between the database and the ontology. A mapping $m \in \mathcal{M}$ is a pair of the form:

$$m = (s{:}Q(\mathbf{x}),\ t{:}\mathbf{L})$$

where the *source* $s{:}Q(\mathbf{x})$ is a SQL query over Σ, and the *target* $t{:}\mathbf{L}$ specifies a set $\mathbf{L}$ of atom templates. These templates populate specific ontology elements using values from the database. Atoms in $\mathbf{L}$ take the form of $C(t_1(\mathbf{x}))$, $p(t_1(\mathbf{x}), t_2(\mathbf{x}))$, or $d(t_1(\mathbf{x_1}), t_2(\mathbf{x_2}))$, where $C \in$ NC, $p \in$ NP, $d \in$ ND, and $t_i(\mathbf{x_i})$ are terms constructed from answer attributes [Sequeda *et al.*, 2011].

Virtual ABox ($\mathcal{M}(\mathcal{D})$). Classically, a Knowledge Base (KB) is defined as a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{A}$ (the ABox) is a finite set of materialized assertions (triples). These triples induce a graph representation where entities are nodes and relations are edges, creating a Knowledge Graph. In the VKG setting, however, the ABox is not materialized but *virtual*. Given a database instance $\mathcal{D}$, the *virtual ABox* $\mathcal{M}(\mathcal{D})$ is the set of RDF triples generated by applying the mappings $\mathcal{M}$ to $\mathcal{D}$:

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

where $L[\mathbf{x} \mapsto \mathbf{o}]$ denotes the ground atoms obtained by replacing attributes in $\mathbf{x}$ with the corresponding values $\mathbf{o}$ retrieved from the database. Therefore, querying a VKG is equivalent to reasoning over the virtual KB $\mathcal{K} = (\mathcal{T}, \mathcal{M}(\mathcal{D}))$.

SPARQL Query. SPARQL is the standard language for querying RDF graphs. A SPARQL query is primarily composed of a *graph pattern*. The fundamental unit of the pattern is a *triple pattern* (s, p, o), where any position can be a variable. Complex patterns are constructed by combining triple patterns using logical operators such as `AND` (conjunction), `UNION` (disjunction), `OPTIONAL`, and `FILTER`. Evaluating a SPARQL query y in VKG is performed by first rewriting it using $\mathcal{P}$ to a SQL query Y over the database schema Σ, and evaluating it over $\mathcal{D}$.

2.2 Task Definition

Given a VKG specification $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \Sigma)$, the objective of VKG-QA is to construct a generation model $f_{\mathcal{P}}$ that takes a natural language question q as input, and generates a SPARQL query $y = f_{\mathcal{P}}(q)$ that corresponds to q. The generated query y is considered valid if its execution yields the expected result set of q. This is a *schema-only* setting: $f_{\mathcal{P}}$ can access only the intensional specification $(\mathcal{T}, \mathcal{M})$, not the database instance $\mathcal{D}$ or materialized triples.

3 Methodology

We propose *NaVQA* (Navigation-based VKG Question Answering), an end-to-end framework to convert an NL question to a SPARQL query for a given VKG system. As illustrated in Figure 1, the pipeline operates in two sequential stages: Stage 1: Active Ontology Indexing (offline) takes the ontology $\mathcal{T}$ and mappings $\mathcal{M}$ as input, prunes unmapped elements, and outputs vector indices ($\mathcal{I}_C$, $\mathcal{I}_P$). Stage 2: SPARQL Query Generation (online) takes a natural language

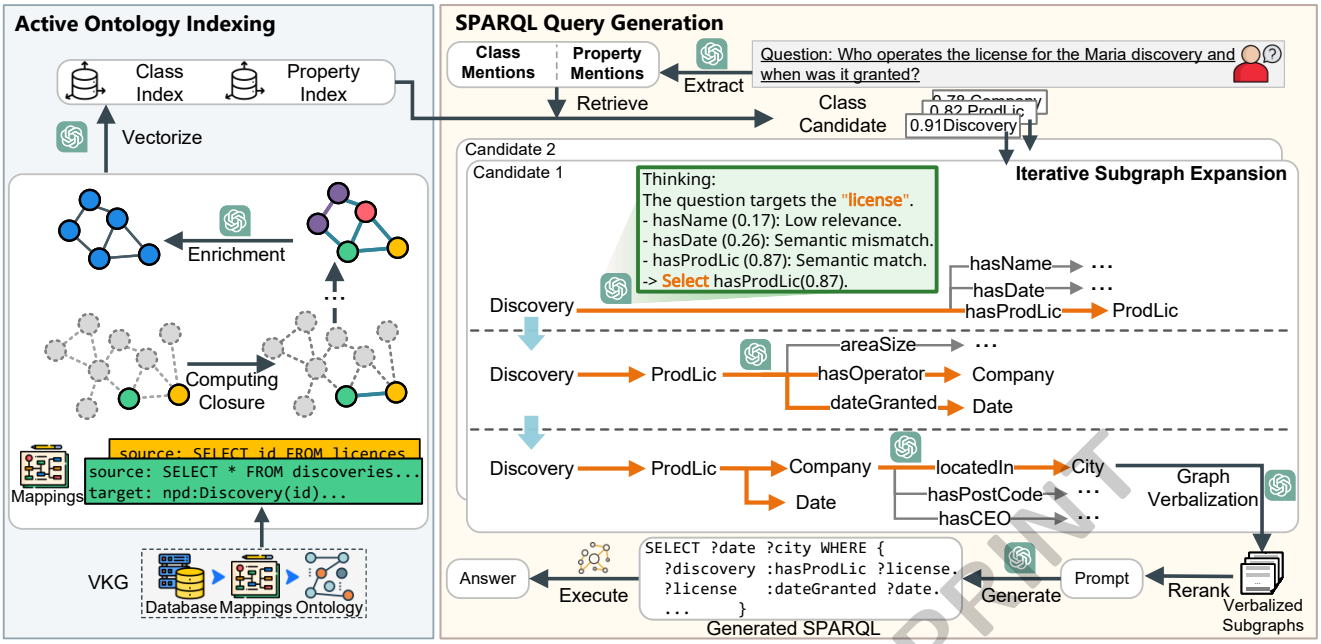

Figure 1: Overview of the *NaVQA* framework.

question as input, retrieves candidate elements, iteratively expands a subgraph, synthesizes a SPARQL query, and executes it against the VKG endpoint to retrieve answers.

3.1 Active Ontology Indexing

Active Ontology Construction

First, the set of *seed elements* $\mathcal{E}_0$, which are classes or properties explicitly referenced by the target templates $\mathbf{L}$ within $\mathcal{M}$ is extracted. Formally,

$$\mathcal{E}_0 = \{e \mid (s{:}Q(x), t{:}\mathbf{L}) \in \mathcal{M}, e \in \text{sig}(\mathbf{L})\}$$

Then we close the elements in $\mathcal{E}_0$ under directly stated ontological dependencies. Formally, for a class e occurring in $\mathcal{T}$, $\text{Dep}(e)$ denotes the direct superclasses of e explicitly stated in $\mathcal{T}$; for a property e occurring in $\mathcal{T}$, $\text{Dep}(e)$ includes its directly stated super-properties and the classes referenced in its domain and range axioms. Since one step only retrieves the immediate context of the current signature, the closure is derived through an iterative sequence initialized with $\mathcal{E}_0$. The set at step $k+1$ expands $\mathcal{E}_k$ with these dependencies:

$$\mathcal{E}_{k+1} = \mathcal{E}_k \cup \bigcup_{e \in \mathcal{E}_k} \text{Dep}(e)$$

The iteration terminates upon reaching a fixed point where $\mathcal{E}_{k_0+1} = \mathcal{E}_{k_0}$. Finally, the active ontology $\mathcal{T}_a$ is obtained by restricting $\mathcal{T}$ to the signature $\mathcal{E}_{k_0}$.

Index Construction

This phase transforms the signature of $\mathcal{T}_a$ into semantic vector indices. For every ontology element e in $\mathcal{T}_a$ a semantic description $\text{desc}(e)$ is synthesized by an LLM based on aggregated metadata (e.g., their IRIs and the values of properties `rdfs:label`, `rdfs:comment`) retrieved from the ontology. Subsequently, each description is encoded into a dense vector representation $\mathbf{v}_e$ via an embedding model [Zhang *et al.*, 2025], denoted as Vectorize:

$$\mathbf{v}_e = \text{Vectorize}(\text{desc}(e))$$

These vectors are organized into two specialized indices: a *Class Index* ($\mathcal{I}_C$) for identifying classes, and a *Property Index* ($\mathcal{I}_P$) for property matching.

3.2 SPARQL Query Generation

Leveraging the active ontology $\mathcal{T}_a$, this stage constructs a question-guided graph over its elements to support SPARQL synthesis. This use of $\mathcal{T}_a$ is a closed-world search heuristic over mappings: although $\mathcal{T}$ keeps its open-world semantics, unmapped ontology elements cannot produce triples in $\mathcal{M}(\mathcal{D})$; consequently, paths involving such elements are pruned before generation.

Elements Linking and Initialization

To align the user question q with the ontology, we first need to extract its semantic components and link them to the elements in the ontology. For a NL question q, let W_C and W_P denote the sets of *entity mentions* and *property mentions* extracted from q via an LLM.

For an ontology element e with embedding $\mathbf{v}_e \in \mathcal{I}_C \cup \mathcal{I}_P$ and a set of mentions W, we define the relevance score $\mathcal{S}(e, W)$ as the maximum cosine similarity between $\mathbf{v}_e$ and any mention embedding $w \in W$:

$$\mathcal{S}(e, W) = \max_{w \in W} \left(\frac{\mathbf{v}_e \cdot \text{Vectorize}(w)}{\|\mathbf{v}_e\| \|\text{Vectorize}(w)\|} \right)$$

This score is used to retrieve the most relevant elements with respect to the NL question q. In particular, we employ

the Top_k operator, which selects the subset of k items with the highest scores, to specify a set C_{cand} of *candidate classes*:

$$C_{\text{cand}} = \mathrm{Top}_k\left(\mathcal{I}_C, \mathcal{S}(\cdot, W_C)\right)$$

Iterative Subgraph Expansion

The goal of this stage is to iteratively construct a subgraph $\mathcal{G}_{sub}$ of the active ontology $\mathcal{T}_a$, viewed as a labeled graph $\mathcal{G}_a$. The graph $\mathcal{G}_a$ consists of triples $\langle c_1, p, c_2\rangle$, where c_1 and c_2 are class names and p is a property name such that $\mathcal{T}_a$ entails that the domain (resp., range) of p is c_1 (resp., c_2). Subgraph $\mathcal{G}_{sub}$, which will later serve as the basis to the process of SPARQL query generation, is obtained by traversing $\mathcal{G}_a$ starting from the set C_{cand} of candidate classes.

We formalize the traversal as a generic search algorithm with a frontier $\mathcal{B}$, initially set to a node in C_{cand}. Expansion at each node $node_{curr}$ proceeds according to Steps 1-2 below.

Step 1: Dynamic Property Selection. We first identify the set of outgoing properties P_{out} connected to $node_{curr}$ in $\mathcal{G}_a$. To guide the expansion, we leverage an LLM as a judge that implements a decision function $\mathrm{LLM}_{judge}(p, q, \mathcal{S}) \in \{0, 1\}$, evaluating whether property p is required to answer the user's question q, taking into account the relevance score $\mathcal{S}$. This decision function is used to create a set P_{sel} of selected properties, defined as:

$$P_{sel} = \{p \in P_{out} \mid \mathrm{LLM}_{judge}(p, q, \mathcal{S}(p, W_P)) = 1\}.$$

This approach aims to restrict the expansion to paths aligned with the user question.

Step 2: Path Expansion. For each selected property $p \in P_{sel}$, we retrieve the triples of the kind $\tau = \langle node_{curr}, p, node_{next}\rangle$ in $\mathcal{G}_a$. Let $\mathrm{Depth}(e)$ denote the shortest path distance from node e to the starting node. Then, the update rules for $\mathcal{G}_{sub}$ and frontier $\mathcal{B}$ are defined as follows:

$$\mathcal{G}_{sub} \leftarrow \mathcal{G}_{sub} \cup \{\tau\},$$
$$\mathcal{B} \leftarrow \mathcal{B} \cup \{node_{next} \mid \mathrm{Depth}(node_{curr}) + 1 \leq L_{max}\}.$$

Note that, to guarantee termination, we bound the expansion from each node to a maximum depth L_{max}, thereby preventing infinite exploration loops.

Step 3: Graph Verbalization. The resulting subgraph $\mathcal{G}_{sub}$ is then used as the knowledge context for SPARQL query generation by an LLM. Following common practice, $\mathcal{G}_{sub}$ is first converted into a linearized textual representation of its triples (see Figure 1), yielding a *verbalized subgraph*.

Query Synthesis and Execution

Subgraph Reranking. We use the procedure described to produce a verbalized subgraph for each class in C_{cand}. Subsequently, a reranker (instantiated with Qwen3-Reranker-8B [Zhang *et al.*, 2025]) is employed to evaluate the alignment between each verbalized subgraph and the user question q. Ultimately, the single subgraph with the highest semantic alignment score is selected as the optimal context, with the aim of filtering out noise from less relevant candidates.

Query Generation and Execution. Conditioned on the selected optimal context and the user question, an LLM generates the final SPARQL query. Answers to the original NL question q are retrieved by executing this query against the Virtual Knowledge Graph (VKG) endpoint.

Dataset	*Ontology ($\mathcal{T}$)*			*Database (Σ)*			*Quest.*
	Cls.	*Obj.*	*Dat.*	*Tabs.*	*Cols.*	*Rows*	
NPD	342	142	238	70	962	257K	134
Bgee	32	23	6	6	29	97K	42
ATKIS	147	12	139	147	1,607	2.7M	144

Table 2: VKG dataset statistics. *Quest.* denotes the number of annotated question-SPARQL pairs.

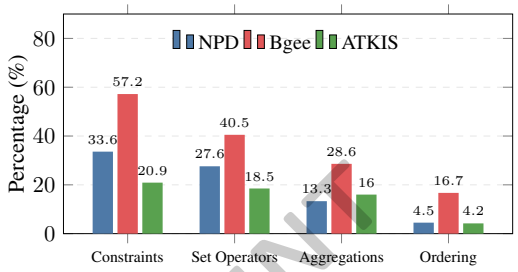

Figure 2: Distribution of SPARQL query structural features. *Constraints*: `FILTER`; *Set Operators*: `UNION`, `MINUS`, or `OPTIONAL`; *Aggregations*: `GROUP BY` or `COUNT`; *Ordering*: `ORDER BY` or `LIMIT`.

4 Experiments

4.1 Datasets and Setup

Our evaluation is conducted on three real-world VKGs representing diverse domains: *NPD* (Oil Exploration) [Lanti *et al.*, 2015], *Bgee* (Gene Expression) [Calvanese *et al.*, 2021], and *ATKIS* (Geospatial Data) [Xiao *et al.*, 2025]. As summarized in Table 2, these datasets pose distinct challenges: *NPD* is characterized by dense domain vocabulary and uneven mapping coverage, where many semantically plausible ontology elements lack mappings and may trigger silent failures; *Bgee* features a deep hierarchical structure requiring rigorous reasoning over transitive closures; and *ATKIS* represents a data-intensive scenario with over 2.7 million rows and 1,607 columns across the schema.

Given the absence of public QA datasets for these VKGs, we manually curate a dataset of 320 natural language questions paired with ground-truth SPARQL queries. The construction process follows a strict *generation-verification* protocol: (1) experts formulated questions covering diverse logical intents based on the ontology; (2) corresponding SPARQL queries were authored and executed against the Ontop system [Calvanese *et al.*, 2016] to verify their correctness; and (3) a cross-validation phase was conducted to filter out ambiguous or structurally invalid queries.

Figure 2 categorizes the query structural features into four dimensions. Notably, *Bgee* presents the highest complexity, particularly in *Constraints* (57.2%) and *Set Operators* (40.5%), reflecting the numerical filtering inherent to gene expression data.

4.2 Evaluation Metrics

Following established protocols in semantic parsing [Gu *et al.*, 2021], we employ *execution-based evaluation* to assess the model's performance.

	NPD			*Bgee*			*ATKIS*			*Average*		
Methods	*F1*	*Acc*	*NR*	*F1*	*Acc*	*NR*	*F1*	*Acc*	*NR*	*F1*	*Acc*	*NR*
Qwen-3-30B												
CoT-SPARQL	20.48	19.85	52.67	24.32	24.32	51.35	14.01	14.01	56.57	19.60	19.39	53.53
SPARQLGEN	15.62	15.62	51.04	23.53	23.53	29.41	32.64	26.62	74.27	23.93	21.92	51.57
NaVQA (Ours)	**32.17**	**29.69**	**67.21**	**48.39**	**41.46**	**92.68**	**50.79**	**48.54**	**91.08**	**43.78**	**39.90**	**83.66**
GPT-5												
CoT-SPARQL	27.68	26.51	73.48	33.33	33.33	74.35	32.21	30.85	68.14	31.07	30.23	71.99
SPARQLGEN	15.87	15.65	53.91	20.51	20.51	40.00	47.15	43.48	71.82	27.84	26.55	55.24
NaVQA (Ours)	**39.12**	**38.98**	**88.70**	**54.54**	**48.78**	**97.62**	**65.16**	**62.05**	**87.71**	**52.94**	**49.94**	**91.34**

Table 3: Performance comparison on the NPD, Bgee, and ATKIS datasets. All metrics (*F1*, *Acc*, *NR*) are expressed as percentages. Best results are highlighted in **bold**.

To facilitate consistent evaluation, execution results are normalized into a canonical format. Specifically, we treat each result row as a *multiset* of RDF terms, thereby ignoring variable ordering (e.g., treating `SELECT ?s ?o` as equivalent to `SELECT ?o ?s`). The canonicalized execution result is constructed as a *multiset* of these canonicalized rows, preserving duplicate occurrences to capture the exact cardinality of the execution result. For question i ($1 \leq i \leq N$), let $\mathcal{A}^{(i)}_{ref}$ and $\mathcal{A}^{(i)}_{gen}$ denote the result multisets for the reference answers and the generated query, respectively.

Execution F1. This metric assesses the overlap between the generated and reference answers. To account for duplicate results accurately, we employ the *multiset intersection* operator, denoted as $\cap_{\text{bag}}$. Formally, the multiplicity of a row r in the intersection $\mathcal{A}^{(i)}_{gen} \cap_{\text{bag}} \mathcal{A}^{(i)}_{ref}$ is defined as the minimum of its occurrences in the two multisets: $\min(\text{count}(r, \mathcal{A}^{(i)}_{gen}), \text{count}(r, \mathcal{A}^{(i)}_{ref}))$. Precision ($P$) and Recall ($R$) are then computed as:

$$P^{(i)} = \frac{|\mathcal{A}^{(i)}_{gen} \cap_{\text{bag}} \mathcal{A}^{(i)}_{ref}|}{|\mathcal{A}^{(i)}_{gen}|}, \quad R^{(i)} = \frac{|\mathcal{A}^{(i)}_{gen} \cap_{\text{bag}} \mathcal{A}^{(i)}_{ref}|}{|\mathcal{A}^{(i)}_{ref}|}$$

where $|\cdot|$ denotes the cardinality (total element count) of the multiset. The Execution F1 is the harmonic mean of precision and recall: $F1^{(i)} = 2 \cdot (P^{(i)} \cdot R^{(i)}) / (P^{(i)} + R^{(i)})$. The average F1 is $\frac{1}{N} \sum_{i=1}^{N} F1^{(i)}$.

Execution Accuracy (Acc). This metric measures the proportion of queries that retrieve the exact answer. A generated query is considered correct if and only if its execution result is multiset-equivalent to the reference:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\mathcal{A}^{(i)}_{gen} \equiv \mathcal{A}^{(i)}_{ref})$$

where $\mathbb{I}(\cdot)$ is the indicator function, and $\equiv$ denotes multiset equivalence. Note that this $\mathcal{A}^{(i)}_{gen} \equiv \mathcal{A}^{(i)}_{ref}$ corresponds to achieving an Execution F1 score of 1.0.

Non-Empty Answer Rate (NR). This metric quantifies the severity of silent failures, by measuring the percentage of queries that yield at least one result, regardless of correctness:

$$\text{NR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(|\mathcal{A}^{(i)}_{gen}| > 0)$$

A significant gap between NR and Acc indicates that the model is generating syntactically valid but semantically incorrect queries. Our goal is to maximize NR while maintaining high F1 and Acc.

4.3 Baselines

We evaluate NaVQA against schema-level paradigms using Qwen-3-30B [Yang *et al.*, 2025] and GPT-5 [OpenAI, 2025]. Due to the information-access constraints inherent to the VKG setting (recall Sec. 2.2 and Sec. 1), instance-based KGQA and Graph-RAG methods are not directly applicable, as they typically require materialized triples, graph traversal, or reliable execution feedback. This makes execution-guided SPARQL generation methods such as GRASP [Walter and Bast, 2025] unsuitable as direct baselines: they rely on probing the RDF graph, whereas empty results in VKGs may indicate either an incorrect query or an unmapped ontology element. Therefore, we select: (1) *CoT-SPARQL* [Zahera *et al.*, 2024], representing a *Prompting-based* approach, where an LLM generates SPARQL queries using a Chain-of-Thought [Wei *et al.*, 2022] prompt over the full ontology; and (2) *SPARQLGEN* [Kovriguina *et al.*, 2023], representing *Retrieval-Augmented* generation, which retrieves ontology elements relevant to the user question and utilizes one-shot prompting to guide SPARQL query generation.

4.4 Main Results

Table 3 presents the comparative performance of NaVQA against representative baselines across different backbone models.

Overall Performance Comparison. As indicated in Table 3, NaVQA establishes a significant performance advantage across all metrics. In terms of Average F1, NaVQA (with GPT-5) reaches *52.94%*, outperforming the CoT-SPARQL baseline by a margin of *+21.87%*. This superiority is consistent across datasets with varying characteristics. Notably, on *ATKIS*, which features the largest data scale, NaVQA achieves an F1 score of 65.16%, doubling the performance of CoT-SPARQL. Similarly, on *NPD*, characterized by dense domain vocabulary and uneven mapping coverage, NaVQA maintains a distinct lead, demonstrating robust adaptability to both data-intensive and knowledge-intensive scenarios.

Mitigation of silent failures. The NR metric quantifies silent failures, that is, syntactically valid queries yielding no

Variant	*Qwen-3-30B*		*GPT-5*	
	F1	*NR*	*F1*	*NR*
Full NaVQA	*43.78*	*83.66*	*52.94*	*91.34*
w/o AOC	34.77	75.63	45.61	82.57
w/o ISE	38.14	79.27	40.27	84.62
w/o AOC+ISE (CoT-SPARQL)	19.60	53.53	31.07	71.99

Table 4: Ablation study showing average performance (in percentages) across all datasets. *AOC*: Active Ontology Construction; *ISE*: Iterative Subgraph Expansion.

results. We observe a significant difference between NaVQA and the baselines. Standard approaches exhibit low NR scores (e.g., SPARQLGEN on Bgee is 29.41% with Qwen-3), suggesting a high frequency of silent failures. In contrast, NaVQA consistently maintains a high NR (averaging *91.34%* with GPT-5) while achieving the highest Acc scores. This indicates that the generated queries are structurally aligned with the active ontology, effectively minimizing empty-result executions.

Backbone and Grounding Effects. We do not interpret Table 3 as a general comparison between backbone models. The key observation is that both Qwen-3-30B and GPT-5 benefit from active-ontology grounding and iterative subgraph expansion, showing that mapping-aware guidance is crucial in the schema-only VKG setting. Absolute performance still varies with dataset structure, e.g., ATKIS is shallower whereas Bgee requires deeper hierarchical reasoning.

4.5 Ablation Study

To evaluate the isolated contribution of each module, we conducted an ablation study utilizing the average performance across all three datasets. Results are shown in Table 4.

Impact of Individual Modules. As shown in Table 4, removing either AOC or ISE leads to performance drops, but their specific impact varies across metrics. Removing AOC results in a clear decline in NR (e.g., from 91.34% to 82.57% on GPT-5). This confirms that AOC effectively filters out unmapped ontology elements, which is essential for generating executable queries. In contrast, removing ISE has a significant impact on the F1 score, especially on GPT-5, where it drops from 52.94% to 40.27%. This indicates that without the step-by-step verification in ISE, the system struggles to construct valid reasoning paths, leading to structural errors even when the correct ontology elements are selected.

Synergistic Effect. The most significant performance drop occurs when both modules are removed (*w/o AOC+ISE*). In this setting, the framework is equivalent to the *CoT-SPARQL* baseline. As shown in the results, the F1 score decreases substantially to 31.07% on GPT-5 and 19.60% on Qwen-3-30B. This demonstrates that AOC and ISE are complementary and rely on each other to achieve high performance. AOC defines the valid boundaries of the search space, while ISE validates the subgraph structure within these boundaries. Without the guidance from these two modules, the LLM relies entirely on probabilistic generation based on parametric knowledge, which frequently leads to structural hallucinations.

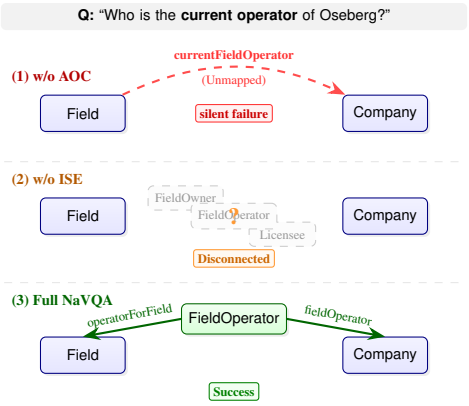

Figure 3: Case study on NPD under different ablation settings.

Dataset	*Search Space (Ontology)*			*Latency (s)*		
	Original	*Pruned*	*Red. (%)*	*w/o AOC*	*w/ AOC*	*Speedup*
NPD	13,449	6,688	50.3%	114.14	32.91	*3.47×*
Bgee	882	664	24.7%	92.46	11.30	*8.18×*
ATKIS	671	641	4.5%	20.93	11.87	*1.76×*

Table 5: Efficiency analysis using GPT-5. AOC reduces the search space and latency across datasets.

Case Study. Figure 3 illustrates the question *"Who is the current operator of Oseberg?"* (1) Without AOC, the LLM selects the semantically intuitive but unmapped property `npd:currentFieldOperator`, leading to a silent failure. (2) Without ISE, it sees valid but fragmented elements, e.g., `npd:FieldOwner` and `npd:FieldOperator`, and may confuse *Owner* with *Operator* or hallucinate a direct edge from `npd:Field` to `npd:Company`. (3) The full model combines both modules: AOC removes unmapped shortcuts, while ISE recovers the mapped path through `operatorForField`, `npd:FieldOperator`, and `fieldOperator`.

4.6 Further Analysis

Efficiency Analysis. Table 5 reports the impact of AOC on end-to-end latency, which measures the total time required to transform an NL question into a SPARQL query. While multi-stage reasoning frameworks typically incur additional computational overhead, NaVQA partially mitigates this issue by reducing the query search space. For instance, on the Bgee dataset, AOC reduces the ontology scale by 24.7%, resulting in an *8.18×* speedup (decreasing from 92.46s to 11.30s). This significant acceleration is attributed to the precise pruning of the ontology: by filtering out irrelevant ontology elements, the system eliminates potential distractors that induce noise during generation, and thus avoids exploring invalid query paths. Consequently, our approach achieves superior accuracy and significantly reduces the total processing time compared to the unpruned baseline.

Robustness to Query Complexity. Table 6 shows the performance relative to the number of hops in the SPARQL queries. The performance of the baseline *SPARQLGEN* de-

	Non-Empty Returns		*Silent Failures*		*Syntax Errors*	
Query Complexity	**N**	**S**	**N**	**S**	**N**	**S**
1-Hop	96	78	1	7	3	15
2-Hop	84	42	3	38	13	20
3-Hop+	70	15	6	60	24	25

Table 6: Distribution of query outcomes (in percentages) across varying query complexities. **N** denotes NaVQA, and **S** denotes the SPARQLGEN baseline.

clines significantly as query hop increases, with *silent failures* rising from 7% (1-hop) to 60% (3-hop+). This indicates that longer query paths significantly increase the probability of traversing unmapped ontology elements, which leads to empty results in the baseline. In contrast, NaVQA exhibits strong robustness to this query complexity. By guiding the generation with the active ontology, it maintains a 70% non-empty return rate on 3-hop+ queries while keeping *silent failures* under 6%, effectively handling complex query paths where mapping coverage is sparse.

5 Related Work

In this section, we review the Text-to-SQL and KGQA paradigms and discuss their applicability to the VKG setting.

5.1 Text-to-SQL

Text-to-SQL aims to translate natural language questions into executable SQL queries. Early approaches primarily relied on sequence-to-sequence models or slot-filling techniques [Qin *et al.*, 2022]. However, with the rapid advancement of LLMs, the recent research has shifted towards leveraging them to address the task [Liu *et al.*, 2025b]. Consequently, we focus our review on these recent LLM-based methodologies.

Prompt Engineering. DAIL-SQL [Gao *et al.*, 2024] enhances performance by selecting few-shot examples via structural similarity. For handling *complex queries*, DIN-SQL [Pourreza and Rafiei, 2023] and C3 [Dong *et al.*, 2023] introduce strategy decomposition and context calibration, enabling models to break down hard queries into manageable steps.

Instruction Tuning. SQL-PaLM [Sun *et al.*, 2024c] fine-tunes proprietary models with execution feedback, while CodeS [Li *et al.*, 2024] offers an open-source alternative with parameters ranging from 1B to 15B, pre-trained on specifically curated SQL-centric corpora.

Agent-based Frameworks. To improve reliability, systems like MAC-SQL [Wang *et al.*, 2025] adopt multi-agent collaboration. By employing specialized roles (e.g., planner or refiner) for iterative validation, these frameworks effectively mitigate hallucinations found in single-pass generation.

5.2 KGQA

KGQA aims to retrieve answers from structured data represented as a graph of entities and relations. We categorize existing approaches based on how they utilize the graph structure during the answer generation process.

Graph-Augmented Generation. Methods like KGP [Wang *et al.*, 2024] and PathRAG [Chen *et al.*, 2025a] verbalize retrieved subgraphs to provide coherent context to an LLM, while KG-Adapter [Tian *et al.*, 2024] employs a learnable adaptation layer to capture graph features. Entity linking and knowledge selection are also closely relevant to VKG-QA. For example, MAPS [Chen *et al.*, 2025b] improves zero-shot entity linking, and Huang et al. [Huang *et al.*, 2025] select question-relevant knowledge via iterative encoding and re-ranking.

Iterative and Agentic Reasoning. These approaches integrate LLMs with dynamic graph traversal or tool use. ToG [Sun *et al.*, 2024a] and RoG [Luo *et al.*, 2024] iteratively explore the graph using intermediate results to guide generation. Agentic frameworks like KG-Agent [Jiang *et al.*, 2025] and ODA [Sun *et al.*, 2024b] further advance this idea by selecting tools or adjusting trajectories based on feedback. More recently, GRASP [Walter and Bast, 2025] uses LLM-guided search and SPARQL execution to construct queries over RDF graphs.

Ontology-Guided Generation. These approaches exploit ontology semantics to improve SPARQL generation or validation. GAIL [Zhang *et al.*, 2024] fine-tunes models on synthesized data to improve logical validity, while a recent study [Liu *et al.*, 2025a] proposes an ontology-guided reverse thinking strategy to enhance generalization. One study [Allemang and Sequeda, 2024] uses ontology-based checking and LLM repair to fix structurally invalid SPARQL queries.

None of the discussed approaches are directly applicable to VKG-QA. Text-to-SQL rely on database schemas, which are typically very different from the high-level ontologies in VKGs, while KGQA requires materialized triples. Moreover, existing ontology-based methods typically assume that all ontology elements are populated. Contrarily to our method, they fail to filter out unmapped ontology elements, often leading to empty results.

6 Conclusion and Future Work

In this paper, we study the problem of silent failures in VKG-QA, which arise from the semantic mismatch between the high-level ontology and the actual data mappings of a VKG setting. We proposed NaVQA, a framework that mitigates this issue by identifying active ontology elements and employing an iterative subgraph expansion to guide SPARQL query generation. Experiments on real-world VKGs confirm that our approach effectively mitigates silent failures and significantly outperforms existing baselines. In future work, we plan to extend NaVQA in two directions: (1) enhancing the capabilities to support complex analytical queries involving nested aggregations and comparative reasoning; and (2) exploring automated mapping enrichment, where the system identifies frequently queried but unmapped ontology elements to suggest new mappings to the VKG maintainers.

E.g., queries for which schema mapping is non-trivial or those using multiple joins or having a nested structure.

Acknowledgments

This work is partially supported by National Nature Science Foundation of China under No. 62476058. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations in this paper. This research has been partially supported by the 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, and by MUR through the PRIN project 2022XERWK9 S-PIC4CHU. 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. Linfang Ding is supported by the Start-up Funding from Hohai University (B250201131) and the Jiangsu Specially Appointed Professors Program.

References

[Allemang and Sequeda, 2024] Dean Allemang and Juan Sequeda. Increasing the Accuracy of LLM Question-Answering Systems with Ontologies. In *Proc. of ISWC*, page 324–339, 2024.

[Banerjee *et al.*, 2023] Debayan Banerjee, Pranav Nair, et al. The role of output vocabulary in T2T LMs for SPARQL semantic parsing. In *Findings of ACL*, pages 12219–12228, 2023.

[Calvanese *et al.*, 2016] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3):471–487, 2016.

[Calvanese *et al.*, 2021] Diego Calvanese, Davide Lanti, Tarcisio Mendes De Farias, Alessandro Mosca, and Guohui Xiao. Accessing scientific data through knowledge graphs with Ontop. *Patterns*, 2(10), 2021.

[Chen *et al.*, 2025a] Boyu Chen, Zirui Guo, et al. PathRAG: Pruning Graph-based Retrieval Augmented Generation with Relational Paths. *arXiv preprint arXiv:2502.14902*, 2025.

[Chen *et al.*, 2025b] Chao Chen, Pengfei Luo, Changkai Feng, et al. MAPS: A Multi-task Framework with Anchor Point Sampling for Zero-shot Entity Linking. *Data Intelligence*, 7(4):1085–1107, 2025.

[Ding *et al.*, 2020] Linfang Ding, Guohui Xiao, Diego Calvanese, and Liqiu Meng. A framework uniting ontology-based geodata integration and geovisual analytics. *ISPRS Int. J. Geo-Inf.*, 9(8):474, 2020.

[Dong *et al.*, 2023] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, et al. C3: Zero-shot Text-to-SQL with ChatGPT. *arXiv preprint arXiv:2307.07306*, 2023.

[D'Abramo *et al.*, 2025] Jacopo D'Abramo, Andrea Zugarini, and Paolo Torroni. Investigating Large Language Models for Text-to-SPARQL Generation. In *Proceedings of KALM*, pages 66–80, 2025.

[Gao *et al.*, 2024] Dawei Gao, Haibin Wang, Yaliang Li, Xiunei Sun, Yanyang Qian, Bolin Ding, and Jingren Zhou. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proceedings of the VLDB Endowment*, 17(11):3103–3116, 2024.

[Gu *et al.*, 2021] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases. In *Proceedings of the Web Conference 2021*. ACM, 2021.

[Huang *et al.*, 2025] Zhisheng Huang, Xudong Jia, Tao Chen, and Zhongwei Zhang. A General Scalable Approach for Knowledge Selection based on Iterative Hybrid Encoding and Re-ranking. *Data Intelligence*, 7(1):124–142, 2025.

[Jiang *et al.*, 2025] Jinhao Jiang, Kun Zhou, et al. KG-Agent: An Efficient Autonomous Agent Framework for Complex Reasoning over Knowledge Graph. 2025.

[Kovriguina *et al.*, 2023] Liubov Kovriguina, Roman Teucher, Daniil Radyush, and Dmitry Mouromtsev. SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation. In *SEMANTICS*, 2023.

[Lanti *et al.*, 2015] Davide Lanti, Martin Ignacio Rezk, Guohui Xiao, and Diego Calvanese. The NPD benchmark: Reality check for OBDA systems. In *Proc. of EDBT*, pages 617–628, 2015.

[Lenzerini, 2011] Maurizio Lenzerini. Ontology-based data management. In *Proc. of CIKM*, pages 5–6, 2011.

[Li *et al.*, 2024] Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, et al. CodeS: Towards Building Open-Source Language Models for Text-to-SQL. 2024.

[Liu *et al.*, 2025a] Runxuan Liu, Bei Luo, Jiaqi Li, Baoxin Wang, Ming Liu, et al. Ontology-Guided Reverse Thinking Makes Large Language Models Stronger on Knowledge Graph Question Answering. In *Proc. of ACL*, pages 15269–15284, July 2025.

[Liu *et al.*, 2025b] Xinyu Liu, Shuyu Shen, et al. A Survey of Text-to-SQL in the Era of LLMs: Where Are We, and Where Are We Going? *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[Luo *et al.*, 2024] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *Proc. of ICLR'2024*, 2024.

[Luz and Finger, 2018] Fabiano Ferreira Luz and Marcelo Finger. Semantic Parsing Natural Language into SPARQL: Improving Target Language Representation with Neural Attention. *arXiv preprint arXiv:1803.04329*, 2018.

[Meloni *et al.*, 2025] Antonello Meloni, Diego Reforgiato Recupero, et al. Exploring Large Language Models for Scientific Question Answering via Natural Language to SPARQL Translation. *ACM Transactions on Intelligent Systems and Technology*, 2025.

[Motik *et al.*, 2012] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, et al. OWL 2 Web Ontology Language:

Profiles. W3C Recommendation, World Wide Web Consortium, 2012.

[OpenAI, 2025] OpenAI. GPT-5 System Card. System card, OpenAI, 2025.

[Pan *et al.*, 2025] Xueli Pan, Victor de Boer, and Jacco van Ossenbruggen. FIRESPARQL: A LLM-based Framework for SPARQL Query Generation over Scholarly Knowledge Graphs. 2025.

[Pourreza and Rafiei, 2023] Mohammadreza Pourreza and Davood Rafiei. DIN-SQL: Decomposed In-Context learning of Text-to-SQL with Self-Correction. In *NeurIPS*, volume 36, 2023.

[Qin *et al.*, 2022] Bowen Qin, Binyuan Hui, et al. A survey on Text-to-SQL parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*, 2022.

[Rodriguez-Muro *et al.*, 2013] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of ISWC'2013*, pages 558–573. Springer, 2013.

[Sequeda *et al.*, 2011] Juan Sequeda, Syed Ahmed Tirmizi, Oscar Corcho, et al. Survey of directly mapping SQL databases to the semantic web. *The Knowledge Engineering Review*, 26(4):445–486, 2011.

[Sequeda *et al.*, 2024] Juan Sequeda, Dean Allemang, and Bryon Jacob. A Benchmark to Understand the Role of Knowledge Graphs on Large Language Model's Accuracy for Question Answering on Enterprise SQL Databases. In *Proc. of GRADES-NDA*, pages 1–12, 2024.

[Sun *et al.*, 2024a] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, et al. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In *Proc. of ICLR'2024*, 2024.

[Sun *et al.*, 2024b] Lei Sun, Zhengwei Tao, Youdi Li, and Hiroshi Arakawa. ODA: Observation-Driven Agent for integrating LLMs and Knowledge Graphs. In *Proc. of ACL*, pages 7417–7431, 2024.

[Sun *et al.*, 2024c] Ruoxi Sun, Sercan Omer Arik, Alexandre Muzio, Lesly Miculicich, et al. SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL. *Transactions on Machine Learning Research*, 2024.

[Tian *et al.*, 2024] Shiyu Tian, Yangyang Luo, et al. KG-Adapter: Enabling Knowledge Graph Integration in Large Language Models through Parameter-Efficient Fine-Tuning. In *ACL Findings*, pages 3813–3828, 2024.

[Walter and Bast, 2025] Sebastian Walter and Hannah Bast. GRASP: Generic Reasoning And SPARQL Generation across Knowledge Graphs. *ISWC (Posters, Demos & Industry Tracks)*, 2025.

[Wang *et al.*, 2024] Yu Wang, Nedim Lipka, Ryan Anthony Rossi, et al. Knowledge Graph Prompting for Multi-Document Question Answering. In *AAAI*, volume 38, pages 19206–19214, 2024.

[Wang *et al.*, 2025] Bing Wang, Changyu Ren, et al. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. In *Proc. of COLING*, pages 540–557, 2025.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, et al. Chain-of-thought Prompting Elicits Reasoning in Large Language Models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[Xiao *et al.*, 2018] Guohui Xiao, Diego Calvanese, Roman Kontchakov, et al. Ontology-Based Data Access: A Survey. In *Proc. of IJCAI'2018*, 2018.

[Xiao *et al.*, 2019] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. Virtual Knowledge Graphs: An Overview of Systems and Use Cases. *Data Intelligence*, 1(3):201–223, 2019.

[Xiao *et al.*, 2025] Guohui Xiao, Diluxan Sathalingam, Albulen Pano, Yu Feng, and Linfang Ding. ATKIS-DLM-KG: a unified knowledge graph framework for integrating and querying fragmented topographic-cartographic data in the German digital landscape model. *International Journal of Digital Earth*, 18(1):2528703, 2025.

[Yang *et al.*, 2025] An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report, 2025.

[Zahera *et al.*, 2024] Hamada Zahera, Manzoor Ali, Mohamed Ahmed Sherif, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. Generating SPARQL from Natural Language Using Chain-of-Thoughts Prompting. In *SEMANTICS*, pages 353–368, 2024.

[Zhang *et al.*, 2024] Zhiqiang Zhang, Liqiang Wen, and Wen Zhao. A GAIL Fine-Tuned LLM Enhanced Framework for Low-Resource Knowledge Graph Question Answering. In *Proc. of CIKM'2024*, pages 3300–3309, 2024.

[Zhang *et al.*, 2025] Yanzhao Zhang, Mingxin Li, et al. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176*, 2025.