基于模式感知混合语义-结构检索的自适应 Text-to-Cypher 生成方法

郭震雨[1]，赵海喻[2]，张春红[3]，胡铮[4]，詹智强[5*]

[1] 北京邮电大学，信息与通信工程学院，北京 100876，中国

摘要： 由于需要在图数据库中进行精确的结构推理和模式（Schema）对齐，从自然语言生成可执行的 Cypher 查询面临着巨大的挑战。现有的方法通常依赖于特定数据集的微调或固定的提示模板，导致泛化能力较差，且对新模式的适应能力有限。在这项工作中，我们提出了一种自适应的、即插即用（plug-and-play）的推理框架，该框架在无需更新模型的情况下仍能保持较高的准确率。我们的方法通过系统地采样其子图和查询路径，构建了一个模式驱动的思维链（Chain-of-Thought）样例库，从而同时捕捉语义意图和结构推理。在推理阶段，模型通过混合语义-结构匹配来检索高相似度的样例，从而在不重新训练的前提下实现上下文感知的、分步的 Cypher 生成。在三个公共基准测试上的实验表明，该方法在执行准确率和结构保真度方面均取得了一致的提升，证实了我们的框架有效地弥合了自然语言意图与图结构推理之间的差距——在没有任何模型修改的情况下，实现了跨不同领域的强大适应性。

关键词： 人工智能，Text-to-Cypher，大语言模型

中图分类号： TP181

Towards Adaptive Text-to-Cypher Generation through Schema-Aware Hybrid Semantic-Structural Retrieval

Zhenyu Guo[1], Haiyu Zhao[2], Chunhong Zhang[3], Zheng Hu[4], Zhiqiang Zhan[5*]

[1] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: Generating executable Cypher queries from natural language is particularly challenging due to the need for precise **structural reasoning and schema alignment** in graph databases. Existing methods often depend on dataset-specific fine-tuning or fixed prompt templates, leading to poor generalization and limited adaptability to new schemas. In this work, we present an **adaptive, plug-and-play inference framework** that eliminates the need for model updates while maintaining high accuracy. Our method constructs a

Foundations: This research received no external funding.

Author Introduction: Zhenyu Guo (2001-), male, Ph.D. candidate, major research direction: Large Language Models, Graph Databases, Knowledge Graphs.Correspondence author: Zhiqiang Zhan (1976-), male, associate professor, major research direction: artificial intelligence–based multi-agent engineering practice training, and intelligent evaluation and benchmarking of engineering competencies.

schema-driven library of Chain-of-Thought exemplars by systematically sampling subgraphs and query paths, capturing both semantic intent and structural reasoning. During inference, the model retrieves high-similarity exemplars via **hybrid semantic-structural matching**, enabling context-aware, stepwise Cypher generation without retraining. Experiments across three public benchmarks demonstrate consistent gains in execution accuracy and structural fidelity, confirming that our framework effectively bridges the gap between natural language intent and graph-structured reasoning—achieving strong adaptability across diverse domains without any model modification.

Key words: Artificial Intelligence, Text-to-Cypher, Large Language Model

0 Introduction

Graph databases have become indispensable for modeling and querying highly interconnected data, powering applications from financial fraud detection and social network analysis to biomedical research [Pan et al.(2024), Hogan et al.(2021)]. The Cypher query language, with its expressive pattern-matching syntax, serves as the de facto standard for navigating these complex structures. However, mastering Cypher remains a significant barrier for domain experts and non-technical users, motivating the need for intuitive Natural Language Interfaces (NLIs) [Tsampos and Marakakis(2025), Sharma et al.(2025)].

Large Language Models (LLMs) have demonstrated impressive capabilities in bridging natural language and structured data, particularly in Text-to-SQL tasks [Brown et al.(2020), Chen et al.(2021), Gao et al.(2023), Chowdhery et al.(2022)]. Yet, extending this success to graph databases—the **Text-to-Cypher** task [Hornsteiner et al.(2024), Zhong et al.(2024)] [Tiwari et al.(2025)]—remains far more challenging. Unlike tabular SQL, graph queries demand **structural reasoning** over relationships, paths, and constraints among heterogeneous entities. Models must not only recognize semantic intent but also map it precisely onto graph topologies, often involving multi-hop reasoning and schema-dependent traversal logic. This structural dependence makes Cypher generation uniquely sensitive to database schemas, which are frequently large, dynamic, and domain-specific [Farías et al.(2025)].

Existing in-context learning approaches, though effective in Text-to-SQL, struggle to generalize in Text-to-Cypher scenarios. Prompt-based methods typically rely on static exemplars or fixed Chain-of-Thought (CoT) templates, which cannot adapt to diverse graph schemas or complex query structures. This results in a persistent **semantic–structural gap**: examples may be semantically related to the input question but structurally mismatched, offering limited guidance for graph reasoning or multi-hop traversal. As a result, current in-context learning pipelines fail to produce syntactically correct and semantically aligned queries when confronted with schema variations or unseen structural patterns [Chang and Fosler-Lussier(2023)].

Complementary research efforts have also explored **fine-tuning-based adaptation** [Zhong et al.(2024), Tiwari et al.(2025)], training LLMs on large domain-specific (NL, Cypher) pairs to improve in-domain accuracy. While such models can perform well within a fixed schema, they suffer from poor scalability, high annotation costs, and limited adaptability to evolving databases—issues that are especially pronounced in graph settings where schemas frequently change.

To address these limitations, we propose a **schema-aware, example-augmented framework** that dynamically retrieves high-similarity Chain-of-Thought exemplars based on both semantic intent and structural alignment. As illustrated in the right panel of Figure 1, our approach equips the LLM with contextually relevant and structurally consistent demonstrations, enabling accurate and adaptive Cypher generation without any fine-tuning. Specifically, our framework comprises: (1) **Schema-driven CoT Example Construction**, which systematically samples subgraphs and query paths to build a diverse example library; (2) **Hybrid Semantic–Structural Retrieval**, selecting top-k exemplars that jointly maximize semantic relevance and structural compatibility; and (3) **Context-Rich Prompting**, integrating these exemplars to guide step-by-step reasoning during Cypher generation.

Comprehensive experiments on three public benchmarks demonstrate that our method substantially improves execution accuracy and robustness, particularly under schema shifts and partially observed environments. In summary, our work reveals that structure-aware retrieval is the key to unlocking adaptive in-context learning for Text-to-Cypher generation.

1 Related Work

1.1 Natural Language to Structured Query Language

The task of translating natural language into structured query languages, commonly known as Text-to-SQL, has been a central and long-standing challenge in natural language processing and database research. Early approaches relied on symbolic or rule-based systems, which were often brittle and difficult to scale. The advent of deep learning, particularly sequence-to-sequence models, marked a major step forward, but these models still struggled with generalizing to unseen database schemas—a problem known as cross-domain Text-to-SQL [Chang and Fosler-Lussier(2023), Arora et al.(2023), Patil et al.(2023)].

The emergence of Large Language Models (LLMs) has fundamentally reshaped this landscape [Brown et al.(2020), Chowdhery et al.(2022)]. LLMs, endowed with powerful in-context learning capabilities, have made few-shot and even zero-shot Text-to-SQL generation feasible without task-specific fine-tuning [Chen et al.(2021)]. Recent research has emphasized prompt design, Chain-of-Thought (CoT) reasoning, and retrieval-augmented strategies to improve generalization [Gao et al.(2023)]. These studies confirm that providing contextually relevant ex-

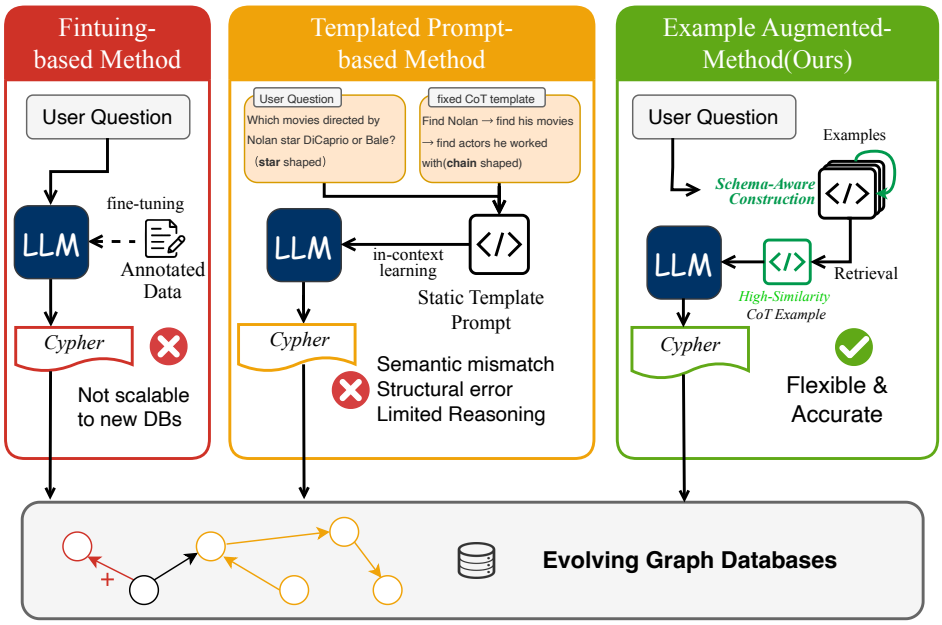

Figure 1: Mainstream Text-to-Cypher paradigms: finetuning-based (left), prompt-based (middle), and our schema-aware example-augmented method (right).

amples or schema information is critical for accurate query generation. However, due to the inherent structural complexity of graphs, insights from Text-to-SQL cannot be directly applied to Text-to-Cypher, motivating dedicated approaches.

1.2 Text-to-Cypher

Recent progress on Text-to-Cypher has followed two dominant paradigms: **fine-tuning-based adaptation** and **prompt-based in-context learning**.

Fine-tuning Approaches. Several works fine-tune LLMs on domain-specific (NL, Cypher) pairs, achieving strong in-domain performance when abundant annotated data is available [Liang et al.(2025), Mandilara et al.(2025)]. To mitigate annotation costs, methods such as SyntheT2C [Zhong et al.(2024)] and AutoCypher [Tiwari et al.(2025)] leverage synthetic data generation or generate–verify frameworks. Despite reducing manual effort, these models remain tightly coupled to specific graph schemas and lack cross-schema generalization, a limitation also observed in Text-to-SQL systems under schema or domain shifts [Chang and Fosler-Lussier(2023), Arora et al.(2023), Patil et al.(2023)]. Consequently, fine-tuning approaches scale poorly in dynamic, heterogeneous environments where graph structures evolve over time.

Prompt-based and In-context Learning. Leveraging the few-shot capabilities of

LLMs, prompt-based methods offer a flexible and tuning-free alternative [Hornsteiner et al.(2024), Ozsoy et al.(2025)]. These models typically incorporate a few demonstration examples in the prompt to guide query generation. However, the use of *static* or hand-crafted Chain-of-Thought (CoT) templates often fails to capture the diverse reasoning structures inherent to Cypher, which involves multi-hop traversals, optional matches, and nested filtering logic. Recent evaluations such as CypherBench [Sun et al.(2024), Munir et al.(2024)] highlight the weaknesses of existing prompt-based systems under schema variation and unseen query types. These findings collectively reveal a critical research gap—the need for mechanisms that can *dynamically* retrieve contextually relevant examples that are not only semantically similar but also structurally aligned with user intent. Our work directly addresses this challenge by introducing a schema- and structure-aware retrieval strategy for in-context Text-to-Cypher generation.

2 Method

Our method comprises three components: (1) schema-aware CoT-based example construction, (2) hybrid similarity-based example retrieval, and (3) in-context Cypher generation guided by CoT reasoning. The overall pipeline is shown in Figure 2.

2.1 Schema-Aware CoT Example Construction

To systematically construct a high-quality library of reasoning exemplars, we design a **schema-aware generate–verify–refine framework**, formally described in **Algorithm 1**. The framework operates as a closed-loop process that incrementally distills schema-level information into validated Chain-of-Thought (CoT) examples through iterative generation, execution, and feedback refinement.

Stage I: Schema-Guided Template Generation. The process begins by feeding the database schema $\mathcal{S}$ into an LLM with structure-specific prompts (e.g., chain, star, tree). This produces diverse Cypher query templates that ensure structural coverage. For instance, given the schema `(Actor)-[:ACTED\_IN]->(Movie)-[:DIRECTED\_BY]->(Director)`, a "chain" prompt such as "*Based on the schema, write a query to find all actors who have worked with a specific director.*" yields a parameterized Cypher template: `MATCH (a:Actor)-[:ACTED\_IN]->(:Movie)-[:` ↪ `DIRECTED\_BY]->(d:Director {name:$director_name$})RETURN a.name`. The generated templates are executed to gather representative entities, attributes, and relationship paths that serve as structural backbones.

Stage II: Back-Translation for Semantic Alignment. Each generated Cypher query is then converted into a natural language question via a **back-translation mechanism**

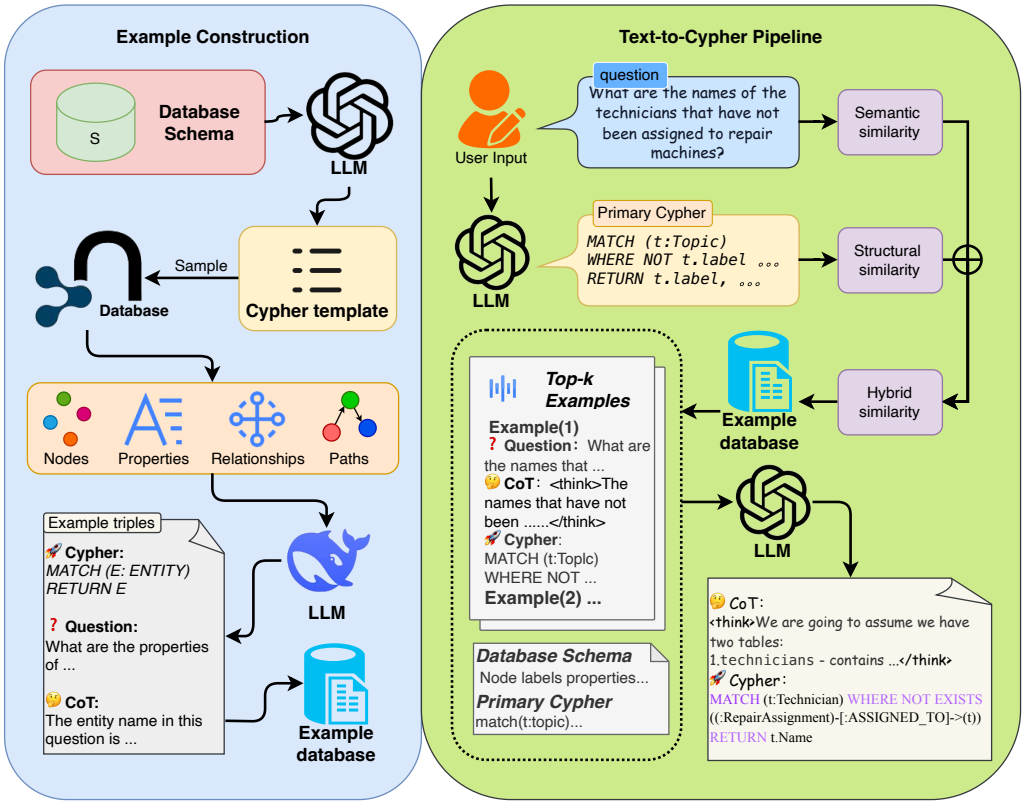

Figure 2: Overview of our framework. Left: schema-driven sampling and CoT-based explanation build the example library. Right: Cypher generation uses hybrid similarity retrieval and CoT-guided refinement for accurate query construction.

[Guo et al.(2018), Li et al.(2023), Nie et al.(2019)], enhancing linguistic diversity and semantic consistency. For example, the instantiated query `MATCH (a:Actor)-[:ACTED\_IN]->(:Movie)-[:DIRECTED\_BY]->(d:Director {name:'Christopher␣Nolan'})RETURN a.name` is translated to "*Which actors have worked with director Christopher Nolan?*" , yielding a well-aligned (NL, Cypher) pair.

Stage III: Execution-Guided Refinement. For each question–query pair $\langle Q, C_t \rangle$, we initialize a prompt P_0 and iteratively refine the reasoning process using a **Monte Carlo-style prompting strategy** [Kocsis and Szepesvári(2006), Li et al.(2025)]. At each iteration i, the model generates a reasoning trace CoT_i, reconstructs a candidate query C_i, and executes it to obtain result $\mathcal{R}_i$. All these elements form the feedback tuple $\mathcal{F}_i = \{P_i, \text{CoT}_i, C_i, \mathcal{R}_i\}$, which is analyzed to update the next prompt $P_{i+1} = \text{UpdatePrompt}(P_i, \mathcal{F}_i)$.

Algorithm 1: Execution-Guided Generate–Verify–Refine Framework for Schema-Aware CoT Construction

Input: Schema $\mathcal{S}$, question Q, target query C_t, max iterations N
Output: Valid CoT example $\langle Q, C_t, \text{CoT} \rangle$ or $\varnothing$
Initialize prompt $P \leftarrow P_0$; iteration counter $i \leftarrow 0$;
while $i < N$ **do**
  Generate reasoning trace CoT_i using LLM with (P, Q);
  Generate query C_i from CoT_i and Q;
  Execute C_i on database and obtain result $\mathcal{R}_i$;
  if $\mathcal{R}_i \equiv \textit{Expected}(C_t)$ **then**
    Store $\langle Q, C_t, \text{CoT}_i \rangle$ in library;
    return success ; `// Functionally equivalent result`
  Construct feedback tuple $\mathcal{F}_i \leftarrow \{C_i, \mathcal{R}_i, \text{CoT}_i, P_i\}$;
  Update prompt $P \leftarrow \text{UpdatePrompt}(P, \mathcal{F}_i)$; `// Execution-guided refinement`
  $i \leftarrow i + 1$;
return $\varnothing$; `// Discard invalid example after` N `iterations`

Execution-Guided Self-Refinement. During refinement, the framework dynamically updates prompts according to the feedback type:

1. **Successful Execution:** If C_i executes successfully and $\mathcal{R}_i \equiv \text{Expected}(C_t)$, the triplet $\langle Q, C_t, \text{CoT}_i \rangle$ is stored as a validated exemplar.

2. **Syntax Error:** If execution fails, error messages are incorporated into the next prompt for syntax correction.

3. **Semantic Misalignment:** If execution succeeds but results are incorrect, the model is prompted to adjust schema-level reasoning and property constraints.

4. **Reasoning–Query Inconsistency:** If mismatches occur between CoT_i and C_i, the model is instructed to reconcile reasoning with generated structure.

If no valid result is obtained after N iterations, the example is discarded. Through this iterative feedback mechanism, Algorithm 1 continuously distills high-quality reasoning examples that capture both semantic and structural regularities, forming the foundation for the hybrid retrieval strategy introduced in Section 2.2.

2.2 Hybrid Similarity-based Example Retrieval

Existing retrieval-based approaches for Text-to-Cypher primarily rely on semantic similarity, selecting examples whose natural-language questions are lexically or semantically close to the input [Gao et al.(2022), Hornsteiner et al.(2024), Zhang et al.(2023)]. However, such methods often overlook the graph-structural characteristics of Cypher queries, leading to a semantic–structural mismatch: retrieved examples may appear topically related but differ in relational topology, hop length, or constraint structure. This misalignment hampers the model' s ability to infer correct graph reasoning patterns and generalize to complex query forms.

To address these challenges, we propose a hybrid similarity-based example retrieval framework that integrates both semantic and structural signals for demonstration selection. By jointly considering linguistic relevance and graph-structural compatibility, our method ensures that in-context examples are not only aligned with the user's intent but also mirror the underlying graph operations required by the query. This dual perspective enables more robust in-context learning and leads to greater execution accuracy and structural fidelity in generated Cypher queries.

Semantic Similarity. We compute semantic similarity between the input question and candidate examples using a sentence encoder $f_{\text{sem}}(\cdot)$. Given a test question Q_{test} and a candidate example Q_i, their embeddings are obtained as $\mathbf{q}_{\text{test}} = f_{\text{sem}}(Q_{\text{test}})$ and $\mathbf{q}_i = f_{\text{sem}}(Q_i)$, respectively. The semantic similarity score is then calculated using cosine similarity:

$$\text{sim}_{\text{sem}}(Q_{\text{test}}, Q_i) = \frac{\mathbf{q}_{\text{test}}^{\top}\mathbf{q}_i}{\|\mathbf{q}_{\text{test}}\|\|\mathbf{q}_i\|}.$$

This measure captures linguistic alignment between questions, ensuring that retrieved exemplars are semantically consistent with the user intent while remaining independent of model fine-tuning or language domain.

Structural Similarity. While semantic similarity captures linguistic intent, it cannot reflect how a query interacts with the underlying graph structure. To model this structural dimension, we design a rule-based **structural encoder** $f_{\text{struct}}(\cdot)$ that maps each Cypher query into a multi-dimensional feature vector representing its operational blueprint. This encoding captures both the *logical composition* of the query and the *topological shape* of the graph traversal, following the sequential query encoding paradigm [Bai et al.(2023)].

Specifically, for a Cypher query C, the encoder extracts a vector $\mathbf{z} = f_{\text{struct}}(C)$ composed of three hierarchical feature categories:

1. **Query Clause and Keyword Features.** These binary indicators capture the query' s high-level logical structure, identifying whether specific clauses (`OPTIONAL MATCH`, `WITH`,

`ORDER BY`, etc.) or logical operators (`AND`, `OR`, `NOT`) are present. This component reflects the *reasoning logic* embedded in the query—e.g., whether it performs filtering, projection, aggregation, or path expansion.

2. **Query Complexity and Aggregation Features.** This group of numerical features quantifies the query' s structural scale and analytical depth. It includes counts of node and relationship patterns (`num_nodes`, `num_rels`), filter conditions (`num_conditions`), and aggregation operators (`COUNT`, `AVG`, etc.). Together, these features capture the *computational workload* and *compositional complexity* of the query.

3. **Graph Pattern Topology Features.** This component encodes the literal "shape" of the `MATCH` pattern, which determines how entities are connected. We define five canonical topologies—`chain`, `star`, `tree`, `dag`, and `cyclic_or_dense`—each represented as a one-hot vector. This captures global structural alignment, ensuring that queries with similar traversal structures (e.g., multi-hop chains or branching joins) are recognized as structurally related.

The final feature vector $\mathbf{z}$ thus integrates symbolic, quantitative, and topological aspects of query structure.

Given a test query Q_{test} and a candidate example Q_i, their corresponding structural vectors are represented as $\mathbf{z}_{\text{test}} = f_{\text{struct}}(Q_{\text{test}})$ and $\mathbf{z}_i = f_{\text{struct}}(Q_i)$. The structural similarity is computed as:

$$\text{sim}_{\text{struct}}(Q_{\text{test}}, Q_i) = \frac{\mathbf{z}_{\text{test}}^{\top}\mathbf{z}_i}{\|\mathbf{z}_{\text{test}}\|\|\mathbf{z}_i\|}.$$

A detailed specification of all 15 extracted features, including their categories and data types, is provided in Table 1.

Hybrid Similarity Scoring. To harness the complementary strengths of both semantic and structural views, we define the final hybrid similarity as a weighted combination of the two scores. While structural similarity captures the query's operational blueprint, semantic similarity is vital for capturing the user's specific intent and the entities involved (e.g., distinguishing a query about "movies" from one about "actors"). The final score is computed as:

$$\text{sim}_{\text{hybrid}}(Q_{\text{test}}, Q_i) = \alpha\,\text{sim}_{\text{sem}} + (1-\alpha)\,\text{sim}_{\text{struct}}$$

where the hyperparameter $\alpha \in [0, 1]$ balances the two components. This parameter allows for control over the retrieval strategy: a higher α prioritizes topical relevance, which is useful for queries with common structures but specific entities, while a lower α emphasizes structural isomorphism, which is crucial for generating queries with rare or complex patterns. Based on empirical validation (see Section 4), we use $\alpha = 0.5$ as a robust default. The top-k candidates

Table 1: Comprehensive taxonomy of structural features extracted by the rule-based structural encoder (f_{struct}). These features capture query-level logical composition, operational complexity, and graph traversal topology, forming the structural basis for similarity computation.

Category	Feature Name	Description	Type
Query Clause & Keyword	`has_optional_match`	Indicates whether the query includes the `OPTIONAL MATCH` clause.	Binary
	`has_AND` / `has_OR` / `has_NOT`	Detects the presence of logical operators `AND`, `OR`, and `NOT`.	Binary
	`has_order`	Identifies whether the `ORDER BY` clause is used for result sorting.	Binary
	`has_limit`	Checks for the use of the `LIMIT` clause to restrict result size.	Binary
	`has_unwind`	Indicates whether the query contains the `UNWIND` clause for list expansion.	Binary
Query Complexity	`path_length`	Counts the number of explicit path traversals (e.g., `()-[]->()`).	Count
	`num_nodes`	Total number of node patterns (`()` occurrences).	Count
	`num_rels`	Total number of relationship patterns (`[]` occurrences).	Count
	`num_conditions`	Counts the number of filtering conditions (`=`, `<`, `CONTAINS`, etc.).	Count
	`num_with`	Counts the number of `WITH` clauses used for data projection.	Count
	`num_return_fields`	Number of fields specified in the `RETURN` clause.	Count
Aggregation	`has_aggregation`	Indicates whether the query applies any aggregation function (e.g., `COUNT`, `SUM`).	Binary
	`agg_type_count`	Counts distinct types of aggregation functions used in the query.	Count
Path Pattern	`has_path_pattern`	Detects the use of variable-length path patterns (e.g., `[*..]`).	Binary
Graph Topology	`pattern_shape`	One-hot encoded vector representing the topological structure of the `MATCH` pattern, classified as one of: `chain`, `star`, `tree`, `dag`, or `cyclic_or_dense`.	One-Hot

with the highest hybrid similarity are then selected as the in-context example set E for the subsequent generation stage.

2.3 CoT-Guided Cypher Generation via In-Context Learning

The final query generation process is a two-step approach that leverages our hybrid retrieval mechanism to construct a rich in-context learning prompt. This process utilizes the user query Q, the database schema $\mathcal{S}$, and the retrieved example set E.

Primary Cypher Generation for Retrieval. To enable our hybrid retrieval, we first generate a preliminary Cypher query, C_0, under zero-shot conditions. This initial query's primary role is to serve as a **structural proxy** for the user's intent, providing the necessary input for the structural similarity calculation described in Section 2.2. Based on the resulting hybrid similarity score, we then select the top-k most relevant examples E from our library, ensuring they are both semantically and structurally aligned with the input query.

CoT-Guided Final Query Generation. In the final step, we construct a comprehensive prompt to generate the definitive query, C^*. This prompt integrates all essential components: the user's question (Q), the full database schema ($\mathcal{S}$), and the high-similarity example set (E) retrieved in the previous step. Guided by the Chain-of-Thought reasoning patterns and structural templates provided by these examples, the language model is instructed to perform a step-by-step analysis and produce the final, executable Cypher query. This approach leverages the curated examples to significantly improve the accuracy and robustness of the final output.

3 Experiment

3.1 Experimental Setup

Datasets. We evaluate our method on three representative benchmarks: (1) **SpCQL** [Guo et al.(2022)], a Chinese NL–Cypher dataset featuring diverse and complex query structures; (2) **Text2Cypher-2024v1** [Ozsoy et al.(2025)], an English Neo4j dataset. Following [Ozsoy et al.(2025)], we evaluate on the 3,951-question executable subset with valid `database_reference` fields; (3) **CypherBench** [Feng et al.(2025)], a recent benchmark designed for robust Cypher generation across 11 realistic knowledge graphs, each containing executable NL–Cypher test pairs.

Setup. We use **Qwen-7B** [Team(2023b)] as the base model for Cypher generation, and **DeepSeek-R1** [DeepSeek-AI et al.(2025)] to generate CoT reasoning traces. For retrieval, we adopt `bge-base-zh` [Team(2023a)] for Chinese and `allmpnetbasev2` [Reimers and Gurevych(2019)]

for English sentence embeddings. By default, we retrieve k=3 in-context examples using a hybrid similarity score with α=0.5. All models are accessed via API without additional fine-tuning.

Baseline Models. For fair comparison, we adopt the baselines and evaluation protocols from Text2Cypher [Ozsoy et al.(2025)], CypherBench [Feng et al.(2025)], and the recent reinforcement learning approach Refining Text2Cypher [Tran et al.(2025)]. Specifically, for the Text2Cypher-2024v1 and CypherBench datasets, we include both fine-tuned models (e.g., LLaMA3-8B-Instruct), open-weight foundational models (e.g., Gemma-7B, CodeLlama-7B), and state-of-the-art API models (e.g., GPT-4o, Gemini 1.5). We also integrate the schema filtering method of Nan et al. [Nan et al.(2023)] and the reinforcement learning-based refinement method proposed by Tran et al. [Tran et al.(2025)] for direct comparison with other schema-aware or self-improving Text-to-Cypher systems. For the SpCQL dataset, we follow the same selection of open-source and fine-tuned models to ensure evaluation consistency.

Evaluation Metrics. We report **Google-BLEU** and **execution accuracy (Exact Match)** on SpCQL and Text2Cypher-2024v1 to measure query string similarity and execution correctness, respectively. For CypherBench, we follow the original paper [Feng et al.(2025)] and report three key metrics: **execution accuracy (EX)**; **PSJS (Provenance Subgraph Jaccard Similarity)**, which assesses the structural correctness of a query by measuring the Jaccard similarity of nodes and relationships in the predicted and ground-truth query subgraphs; and **executable percentage (Exec.)**, which indicates the ratio of successfully executed queries.

3.2 Main Results

The performance of our proposed method is summarized in Tables 2 and 3. The results unequivocally demonstrate the effectiveness and robustness of our hybrid retrieval framework across diverse and challenging benchmarks.

On SpCQL and Text2Cypher-2024v1 (Table 2), our method achieves state-of-the-art performance. For the Chinese dataset SpCQL, our approach significantly outperforms strong open-source baselines like CodeLLaMA-13B, improving the BLEU score by over 20 points and Exact Match (EM) by nearly 9 points. The improvements on Text2Cypher-2024v1 are even more pronounced. Notably, while the BLEU score sees a modest gain over the top-performing baseline (0.8117 vs. 0.8017), the EM score experiences a massive leap from 32.50% to **56.76%**. This disparity is critical: it indicates that while lexical similarity (measured by BLEU) is approaching a ceiling, our structure-aware retrieval provides a substantial advantage in generating *functionally correct* queries, which is the ultimate goal. Our method surpasses all other baselines, including heavily fine-tuned proprietary models from Neo4j.

On CypherBench (Table 3), which is designed to test generalization across 11 different knowledge graphs, our method further solidifies its superiority. It achieves an execution accuracy of **73.98%**, dramatically outperforming all open-source models and even surpassing the strongest proprietary models like Claude 3.5 Sonnet by a remarkable margin of over 12 points. This outstanding performance on a multi-domain benchmark is the strongest evidence for the robustness and adaptability of our tuning-free approach, directly addressing the key limitations of fine-tuning methods. The high PSJS score (84.15%) further confirms that our generated queries are not only executable but also structurally aligned with the ground truth.

Table 2: Main results on SpCQL and Text2Cypher-2024v1 datasets using Google-BLEU and execution accuracy (Exact Match).

Dataset	**Model**	BLEU	EM
SpCQL	DeepSeek-v3-0324	0.2985	0.2213
	DeepSeek-R1-0528	0.4425	0.2915
	LLaMA3-8B	0.4856	0.3714
	CodeLLaMA:13B	<u>0.5565</u>	<u>0.4093</u>
	Ours (hybrid)	**0.7858**	**0.4982**
Text2Cypher-2024v1	LLaKos-Instruct-13B (FT)	0.3767	0.1522
	NeeAi-GPT3.5 (FT)	0.4552	0.1489
	Tomasnjo-Text2Cypher (FT)	0.5534	0.2987
	CodeLlama-7B (Instruct)	0.3986	0.0943
	Gemini-1.5 Pro-001	0.5845	0.2307
	GPT-4o	0.6293	0.3173
	Neo4j-Gemma-2B (FT)	0.6470	0.2104
	Neo4j-LLaMA3-8B (FT)	0.5560	0.2299
	Neo4j-Gemini-1.5 Flash (FT)	0.7780	0.2768
	Neo4j-GPT-4o (FT)	<u>0.8017</u>	0.3250
	Schema Filtering [Nan et al.(2023)]	0.4914	0.1438
	Qwen2.5-3B (RL) [Tran et al.(2025)]	0.7701	<u>0.5623</u>
	Ours (hybrid)	**0.8117**	**0.5676**

Table 3: Execution accuracy (EX), provenance subgraph jaccard similarity (PSJS) and executable percentage (Exec.) on the CypherBench test set.

Model	EX (%)	PSJS (%)	Exec. (%)
Open-source LLMs (<10B)			
llama3.2–3b	11.20	17.33	86.46
llama3.1–8b	18.82	30.98	90.67
gemma2–9b	18.61	30.67	68.57
Open-source LLMs (10–100B)			
mixtral–8x7b	19.21	37.01	59.33
qwen2.5–72b	41.87	56.39	86.84
llama3.1–70b	38.84	54.79	92.25
Proprietary LLMs			
gemini1.5-pro-001	39.95	57.70	86.03
gpt-4o-20240806	60.18	76.87	94.93
claude3.5-sonnet-20240620	<u>61.58</u>	<u>80.85</u>	**96.34**
Ours (top-k retrieval)	**73.98**	**84.15**	<u>95.55</u>

4 Analysis

4.1 Effectiveness of Retrieved Examples

The main challenge in Text-to-Cypher generation is bridging the gap between natural language intent and complex graph query structures. As LLMs lack access to database content, they must rely entirely on in-context examples for both semantic mapping and structural pattern inference. Thus, the quality and relevance of retrieved demonstrations are crucial for accurate generation.

On the Text2Cypher-2024v1 dataset, we compare three strategies: *0-shot*, *random-shot*, and our *top-3 hybrid-shot*. This experiment, with results illustrated in Figure 3, provides a clear validation for our structure-aware retrieval methodology. The performance leap from **0-shot** (36.14% EM) to **random-shot** (48.97% EM) is highly instructive, demonstrating that even topically irrelevant examples can provide a powerful **syntactic scaffolding** to guide the model in learning the basic grammar and format of the Cypher language.

However, the subsequent, even larger leap to our **hybrid-shot** method (56.76% EM) confirms that high-level syntactic guidance alone is insufficient. While random examples may teach

the LLM the *language*, our hybrid retrieval method provides a relevant **reasoning blueprint** that teaches it how to *solve the specific problem*. By retrieving examples that are aligned in both semantic intent and structural complexity, our method enables the model to move beyond simple pattern mimicry towards a more effective, targeted form of analogical reasoning. This result confirms that our sophisticated mechanism for selecting schema- and structure-aware examples is essential for achieving accurate Text-to-Cypher generation.

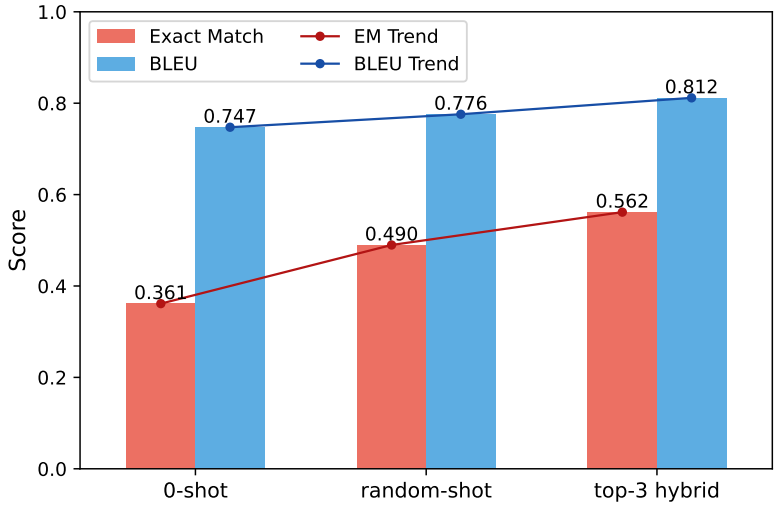

Figure 3: Impact of example retrieval strategy on BLEU and Exact Match (EM) scores for the Text2Cypher-2024v1 dataset. The hybrid top-3 retrieval method consistently outperforms the 0-shot and random-shot baselines.

4.2 Impact of Retrieved Example Size (k)

The number of retrieved in-context examples (k) plays a critical role in balancing information sufficiency and prompt efficiency. Our ablation results highlight a fundamental trade-off in in-context learning for structured query generation: providing too few examples risks insufficient structural and semantic coverage, while including too many can introduce noise, redundancy, and even context truncation due to limited input length.

Trade-off Analysis. When k is small (e.g., $k = 1$), the model becomes highly sensitive to the particular structure of the retrieved exemplar. If this top-ranked example is semantically relevant but structurally mismatched, it may mislead the model' s reasoning process and cause incorrect query patterns. As k increases, the model benefits from richer structural diversity, allowing it to abstract over multiple exemplars and infer a more generalizable mapping between natural language and Cypher operations. However, this advantage plateaus and eventually

reverses once redundant or conflicting patterns dominate the input context, leading to *attention dilution* [Zhao et al.(2021)]—a phenomenon where excessive contextual content reduces focus on task-relevant cues.

Task Complexity Sensitivity. The optimal value of k also varies across query types. For simple single-hop or attribute-retrieval questions, a small k suffices, as additional examples may provide little incremental benefit. In contrast, multi-hop or compositional queries require greater exposure to structurally analogous examples to facilitate correct reasoning paths and relationship chaining. Future work could explore adaptive selection strategies that dynamically adjust k based on query complexity or contextual uncertainty, similar to dynamic demonstration methods in in-context learning.

Key Insight. Empirically, our experiments show that performance consistently improves up to $k = 3$ and degrades beyond that point, suggesting an effective balance between semantic variety and context compactness. This finding supports the notion that small, well-aligned example sets can yield stronger inductive biases than larger but noisier contexts—an observation that aligns with prior studies in retrieval-augmented reasoning [Zhu et al.(2025)].

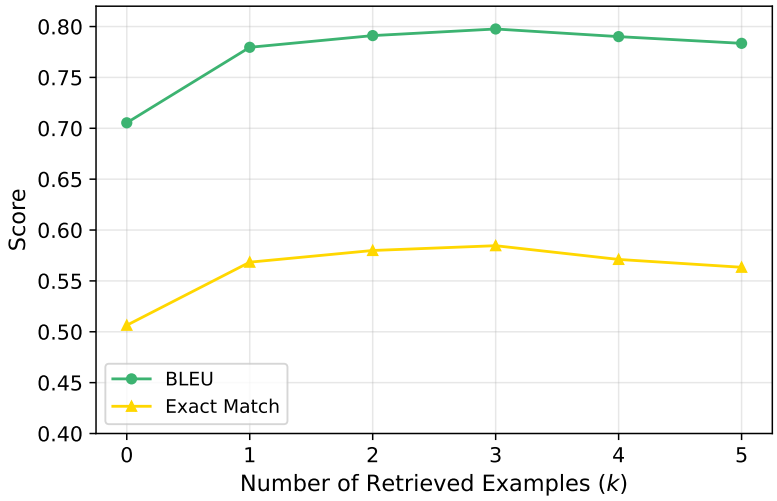

Figure 4: Effect of the number of retrieved examples (k) on BLEU and EM scores. Performance improves up to $k = 3$, then declines due to redundancy and attention dilution.

4.3 Influence of Semantic–Structural Fusion Weight (α)

To analyze the relative contributions of semantic and structural similarity in retrieval, we conduct an ablation study on the neo4j_functional_cypher subset—the largest partition of the

Text2Cypher-2024v1 benchmark. We fix $k = 3$ and vary the fusion weight α from 0 (structure only) to 1 (semantic only), as shown in Figure 5.

Complementary Roles. Semantic similarity captures linguistic intent alignment between the user query and candidate examples, while structural similarity captures the latent relational topology—i.e., node connectivity patterns, path lengths, and operation types. Our results show that either signal alone is insufficient: $\alpha = 0$ or 1 both underperform mid-range values. As α increases from 0 to 0.5, both BLEU and EM scores improve, reflecting that the inclusion of structural awareness helps the model identify relevant subgraph traversal patterns and filtering constraints while maintaining semantic consistency.

Overweighting Semantics. When $\alpha > 0.5$, overemphasizing semantic similarity introduces structurally inconsistent examples that misalign with the target graph operations. This imbalance reduces the model' s logical precision and increases spurious reasoning. In essence, the retrieval distribution becomes semantically dense but structurally sparse, leading to degraded compositional accuracy—a phenomenon similar to overfitting to surface-level textual similarity [Gao et al.(2022)].

Adaptive Fusion. Overall, semantic and structural signals are complementary and their optimal balance ($\alpha = 0.5$) suggests a roughly equal contribution from intent alignment and schema consistency. Future directions could include learning-based fusion strategies, where α becomes a dynamic parameter estimated by a lightweight retrieval controller conditioned on query complexity or uncertainty. Such adaptive retrieval weighting has shown promise in recent hybrid retrieval frameworks [Wan et al.(2025)] and could further enhance Text-to-Cypher generalization across diverse database schemas.

4.4 Bad Case Analysis and Limitations

To better understand the failure modes of our method, we analyzed all bad cases (i.e., failed executions or incorrect results) on three representative datasets: CypherBench, SpCQL, and neo4j_functional
_cypher. We report the distribution of five typical error types—low semantic similarity, low structural similarity, reasoning failure, understanding deficiency, and syntax error—across bad cases in each dataset, as shown in Figure 6. Similar trends are observed across all three datasets.

Low-quality Retrieved Examples. Further analysis reveals two main sources of low-quality retrieval: (1) insufficient coverage in the example library, leading to queries for which no relevant examples can be retrieved, and (2) failure of the retrieval mechanism to select highly

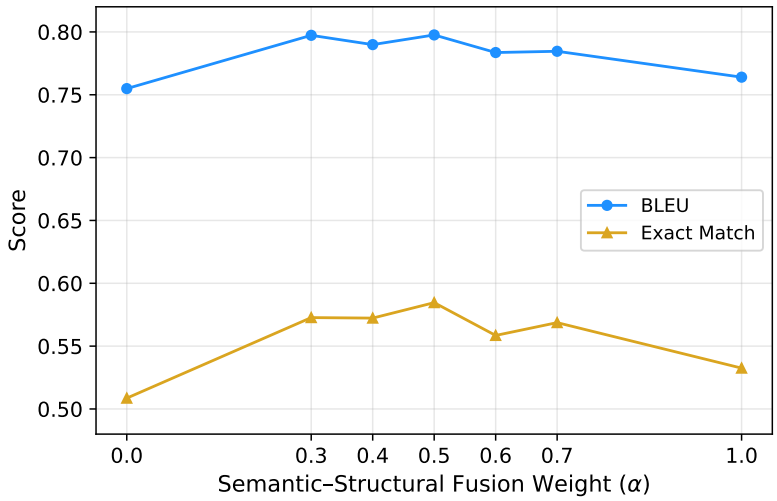

Figure 5: Performance with varying semantic–structural fusion weight α on the `neo4j_functional_cypher` subset. BLEU and EM peak at $\alpha = 0.5$.

relevant examples that actually exist in the library. The latter is often due to misalignment between the primary Cypher structure and the intended query logic, or to suboptimal similarity weighting. This suggests that for structurally clear queries, greater emphasis should be placed on structural similarity, while for queries with less explicit logic, semantic similarity should be weighted more heavily. Future work could explore adaptive retrieval weighting strategies to further mitigate such errors.

High Proportion of Reasoning and Understanding Failures. We identify two major error types: *reasoning failure*, where the model fails to construct logically correct queries despite relevant retrieval, often due to limited multi-step reasoning or generalization capacity; and *understanding deficiency*, where misinterpretation of entities, relations, or attributes leads to queries that are structurally correct but semantically off-target, often yielding empty or irrelevant results. These findings suggest that, even with strong example selection, current LLMs still face challenges in both logical composition and precise entity-relation alignment. Further improvements may come from stronger in-context reasoning abilities and the incorporation of feedback, entity linking, or normalization modules.

Low Syntax Error Rate Syntax errors are relatively rare across all datasets (typically below 10%), suggesting that LLMs have largely internalized Cypher syntax through pretraining. This implies that the main bottleneck is not in surface-level query generation, but in reasoning logic and the precision of retrieval-based guidance.

Figure 6: Distribution of bad case error types across three datasets: CypherBench, SpCQL, and Text2Cypher-2024v1.

5 Conclusion

We present a structure-aware in-context learning framework for Text-to-Cypher generation, which retrieves both semantically and structurally relevant examples to guide LLM query generation. By integrating a structural feature encoder with semantic similarity, our method improves example selection and downstream performance.Experiments on three datasets show that our approach outperforms strong baselines in exact match and BLEU across varied databases. Error analysis further highlights that, while LLMs have largely mastered Cypher syntax, logical reasoning and precise understanding remain challenging when retrieval is suboptimal.Our results underscore the value of structure-guided retrieval for robust in-context learning on graph queries, including domain-specific knowledge graphs. Future directions include structure-aware reranking, relation-aware similarity, and adaptive demonstration composition to further enhance retrieval and generation robustness.

参考文献（References）

[Arora et al.(2023)] Aryaman Arora, Vivek Gupta, Peter Hase, and Hinrich Schütze. 2023. Adapt and Decompose: Efficient Generalization of Text-to-SQL via Domain-Adapted

Least-to-Most Prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 8461–8474. https://arxiv.org/abs/2308.02582

[Bai et al.(2023)] Jiaxin Bai, Tianshi Zheng, and Yangqiu Song. 2023. *Sequential Query Encoding For Complex Query Answering on Knowledge Graphs*. arxiv. arXiv:2302.13114 [cs] doi:10.48550/arXiv.2302.13114

[Brown et al.(2020)] Tom B. Brown, Benjamin Mann, Nick Ryder, and et al. 2020. *Language Models Are Few-Shot Learners*. arxiv. arXiv:2005.14165 [cs] doi:10.48550/arXiv.2005.14165

[Chang and Fosler-Lussier(2023)] Shuaichen Chang and Eric Fosler-Lussier. 2023. Selective Demonstrations for Cross-domain Text-to-SQL. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 14174–14189. doi:10.18653/v1/2023.findings-emnlp.944

[Chen et al.(2021)] Mark Chen, Jerry Tworek, Heewoo Jun, and et al. 2021. *Evaluating Large Language Models Trained on Code*. arxiv. arXiv:2107.03374 [cs] doi:10.48550/arXiv.2107.03374

[Chowdhery et al.(2022)] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, and et al. 2022. *PaLM: Scaling Language Modeling with Pathways*. arxiv. arXiv:2204.02311 [cs] doi:10.48550/arXiv.2204.02311

[DeepSeek-AI et al.(2025)] DeepSeek-AI, Daya Guo, Dejian Yang, and et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs] doi:10.48550/arXiv.2501.12948

[Farías et al.(2025)] Benjamín Farías, Wim Martens, Carlos Rojas, and Domagoj Vrgoč. 2025. *PathFinder: A Unified Approach for Handling Paths in Graph Query Languages*. arXiv:2306.02194 [cs] doi:10.48550/arXiv.2306.02194

[Feng et al.(2025)] Yanlin Feng, Simone Papicchio, and Sajjadur Rahman. 2025. *CypherBench: Towards Precise Retrieval over Full-scale Modern Knowledge Graphs in the LLM Era*. arxiv. arXiv:2412.18702 [cs] doi:10.48550/arXiv.2412.18702

[Gao et al.(2023)] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. *Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation*. arXiv. arXiv:2308.15363 [cs] doi:10.48550/arXiv.2308.15363

[Gao et al.(2022)] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. *SimCSE: Simple Contrastive Learning of Sentence Embeddings.* arxiv. arXiv:2104.08821 [cs] doi:10.48550/arXiv.2104.08821

[Guo et al.(2022)] Aibo Guo, Xinyi Li, Guanchen Xiao, Zhen Tan, and Xiang Zhao. 2022. SpCQL: A Semantic Parsing Dataset for Converting Natural Language into Cypher. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta GA USA, 2022-10-17). ACM, Atlanta, GA, USA, 3973–3977. doi:10.1145/3511808.3557703

[Guo et al.(2018)] Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question Generation from SQL Queries Improves Neural Semantic Parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, 2018). Association for Computational Linguistics, Brussels, Belgium, 1597–1607. doi:10.18653/v1/D18-1188

[Hogan et al.(2021)] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio L Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. 2021. Knowledge graphs. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–37.

[Hornsteiner et al.(2024)] Markus Hornsteiner, Michael Kreussel, Christoph Steindl, Fabian Ebner, Philip Empl, and Stefan Schönig. 2024. Real-Time Text-to-Cypher Query Generation with Large Language Models for Graph Databases. *Future Internet* 16, 12 (2024), 438. doi:10.3390/fi16120438

[Kocsis and Szepesvári(2006)] Levente Kocsis and Csaba Szepesvári. 2006. Bandit Based Monte-Carlo Planning. 282–293 pages. doi:10.1007/11871842_29

[Li et al.(2025)] Boyan Li, Jiayi Zhang, Ju Fan, Yanwei Xu, Chong Chen, Nan Tang, and Yuyu Luo. 2025. *Alpha-SQL: Zero-Shot Text-to-SQL Using Monte Carlo Tree Search.* arxiv. arXiv:2502.17248 [cs] doi:10.48550/arXiv.2502.17248

[Li et al.(2023)] Jiahui Li, Yuwen Zhang, Yujie Hu, Haoyu Zhang, Jicheng Chen, Yuzhuo Wu, and Lei Cui. 2023. OmniSQL: Synthesizing High-quality Text-to-SQL Datasets via Large Language Models. https://arxiv.org/abs/2312.00734. Preprint, arXiv:2312.00734.

[Liang et al.(2025)] Yuanyuan Liang, Keren Tan, Tingyu Xie, Wenbiao Tao, Siyuan Wang, Yunshi Lan, and Weining Qian. 2025. *Aligning Large Language Models to a Domain-specific Graph Database for NL2GQL.* arxiv. arXiv:2402.16567 [cs] doi:10.48550/arXiv.2402.16567

[Mandilara et al.(2025)] Ioanna Mandilara, Christina Maria Androna, Eleni Fotopoulou, Anastasios Zafeiropoulos, and Symeon Papavassiliou. 2025. Decoding the Mystery: How Can LLMs Turn Text Into Cypher in Complex Knowledge Graphs? *IEEE Access* 13 (2025), 80981–81001. doi:10.1109/ACCESS.2025.3567759

[Munir et al.(2024)] Muhammad Munir, Waqas Ahmad, Asad Ashraf, and Murtaza Malik. 2024. Towards Evaluating Large Language Models for Graph-to-Cypher Query Generation. arXiv:2411.08449 [cs.DB] https://arxiv.org/abs/2411.08449 arXiv preprint.

[Nan et al.(2023)] Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. *Enhancing Few-shot Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies.* arxiv. arXiv:2305.12586 [cs] doi:10.48550/arXiv.2305.12586

[Nie et al.(2019)] Allen Nie, Erin Bennett, and Noah Goodman. 2019. DisSent: Learning Sentence Representations from Explicit Discourse Relations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019). Association for Computational Linguistics, Florence, Italy, 4497–4510. doi:10.18653/v1/P19-1442

[Ozsoy et al.(2025)] Makbule Gulcin Ozsoy, Leila Messallem, Jon Besga, and Gianandrea Minneci. 2025. *Text2Cypher: Bridging Natural Language and Graph Databases.* arxiv. arXiv:2412.10064 [cs] doi:10.48550/arXiv.2412.10064

[Pan et al.(2024)] Shirui Pan, Linhao Chen, Yuxiang Wang, Chen Zhang, Xiao Wang, Jiayang Luo, Yi Li, Yufan Chen, Hao Wang, and Cheng Shen. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 4 (2024), 4023–4044. doi:10.1109/TKDE.2023.3301781

[Patil et al.(2023)] Aishwarya Patil, Rishita Das, Tushar Khot, and Peter Clark. 2023. Exploring Dimensions of Generalizability and Few-shot Transfer for Text-to-SQL Semantic Parsing. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, Vol. 203. PMLR, Honolulu, USA, 27367–27382. https://proceedings.mlr.press/v203/patil23a.html

[Reimers and Gurevych(2019)] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410

[Sharma et al.(2025)] Shubham Sharma, Geerija Lavania, Vibhor Bhatt, and Vaibhav Goyal. 2025. GraphQuery: A Natural Language Interface for Graph Database. In *Proceedings of*

the National Conference on Information Technology & Digital Application (NCITSA-2025) (2025). Infogain Publication, India, 64–69. doi:10.22161/ijaems.ncitsa.2025.59

[Sun et al.(2024)] Geng Sun, Yixian Wang, Zemin Sun, Qingqing Wu, Jiawen Kang, Dusit Niyato, and Victor C. M. Leung. 2024. *Multi-Objective Optimization for Multi-UAV-assisted Mobile Edge Computing*. arxiv. arXiv:2404.15292 [eess] doi:10.48550/arXiv.2404.15292

[Team(2023a)] BAAI Team. 2023a. BAAI General Embedding (BGE). https://huggingface.co/BAAI/bge-base-zh. Accessed: 2025-06-25.

[Team(2023b)] Qwen Team. 2023b. Qwen: A Family of Open Large Language Models for the Chinese Language. https://github.com/QwenLM/Qwen. Accessed: 2025-06-25.

[Tiwari et al.(2025)] Aman Tiwari, Shiva Krishna Reddy Malay, Vikas Yadav, Masoud Hashemi, and Sathwik Tejaswi Madhusudhan. 2025. *Auto-Cypher: Improving LLMs on Cypher Generation via LLM-supervised Generation-Verification Framework*. arxiv. arXiv:2412.12612 [cs] doi:10.48550/arXiv.2412.12612

[Tran et al.(2025)] Quoc-Bao-Huy Tran, Aagha Abdul Waheed, Syed Mudasir, and Sun-Tae Chung. 2025. Refining Text2Cypher on Small Language Model with Reinforcement Learning Leveraging Semantic Information. *Applied Sciences* 15, 15 (2025), 8206.

[Tsampos and Marakakis(2025)] Ioannis Tsampos and Emmanouil Marakakis. 2025. Domain- and Language-Adaptable Natural Language Interface for Property Graphs. *Computers* 14, 5 (2025), 183. doi:10.3390/computers14050183

[Wan et al.(2025)] Yuwei Wan, Zheyuan Chen, Ying Liu, Chong Chen, and Michael Packianather. 2025. Empowering LLMs by Hybrid Retrieval-Augmented Generation for Domain-Centric Q&A in Smart Manufacturing. *Advanced Engineering Informatics* 65 (2025), 103212. doi:10.1016/j.aei.2025.103212

[Zhang et al.(2023)] Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-Context Learning for Text-to-SQL with Automatically-Generated Chain-of-Thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Singapore, 3501–3532. doi:10.18653/v1/2023.findings-emnlp.227

[Zhao et al.(2021)] Tianze Zhao, Eric Wallace, Shihao Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-Shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*. PMLR, Virtual Event, 10977–10988. https://proceedings.mlr.press/v139/zhao21c.html

[Zhong et al.(2024)] Ziije Zhong, Linqing Zhong, Zhaoze Sun, Qingyun Jin, Zengchang Qin, and Xiaofan Zhang. 2024. *SyntheT2C: Generating Synthetic Data for Fine-Tuning Large Language Models on the Text2Cypher Task.* arxiv. arXiv:2406.10710 [cs] doi:10.48550/arXiv.2406.10710

[Zhu et al.(2025)] Rongzhi Zhu, Xiangyu Liu, Zequn Sun, Yiwei Wang, and Wei Hu. 2025. Mitigating Lost-in-Retrieval Problems in Retrieval Augmented Multi-Hop Question Answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025), Volume 1: Long Papers.* Association for Computational Linguistics, Toronto, Canada, 22362–22375. https://aclanthology.org/2025.acl-long.1089.pdf