CypherSmith: Transforming Text-to-Cypher Generation for LLMs with Synthetic Data

Zeyu Zhang[1], Kexuan Sun[1], Zheng Tang[1], Jens Voeckler[1], Thien Huu Nguyen[1,2], Thuy Vu[1],

[1]Amazon AGI, USA, [2]University of Oregon, USA
{zeyzhan,sunkexua,zhgtang,jenvockl,thiennh,thuyvu}@amazon.com

Abstract

Knowledge Graph (KG) retrieval is a promising augmentation to address knowledge gaps and hallucinations in LLMs. As KGs in practice are stored in graph databases (e.g., Wikidata, Freebase), accurate retrieval requires translating natural language questions into structured queries (query generation). A key challenge of query generation is Text-to-Cypher, which generates Cypher queries for property graphs (e.g., Neo4j), a paradigm increasingly adopted in industry for their scalable architectures and expressive schemas. However, compared to other query generation tasks such as Text-to-SQL or Text-to-SPARQL, Text-to-Cypher remains underexplored due to scarce public KGs and datasets. Existing datasets are small, domain-limited, and lack diversity, constraining LLM progress. To address this, we introduce CypherSmith, an instruction-tuning dataset over 12× larger than prior public Text-to-Cypher datasets, spanning diverse domains to better support LLM fine-tuning. Our key distinction lies in fully leveraging open-source LLMs for large-scale synthetic data generation and introducing a novel likelihood-based filtering technique to ensure high-quality Text-to-Cypher data. Extensive experiments demonstrate the effectiveness of CypherSmith, achieving state-of-the-art LLM performance.

1 Introduction

Knowledge Graphs (KGs) help mitigate hallucination and knowledge cutoff in Large Language Models (LLMs), especially in Graph Retrieval-Augmented Generation (GraphRAG) (Kim et al., 2023; Edge et al., 2024; Hu et al., 2025) and Knowledge Base Question Answering (KBQA) (Berant et al., 2013; Cao et al., 2022; Jiang et al., 2023; Luo et al., 2024). By modeling entities and relations as nodes and edges, KGs enable compact knowledge representation for efficient retrieval and multi-hop reasoning of complex queries. Early work in GraphRAG and KBQA has emphasized relevant subgraph extraction for input questions (Ma et al., 2022; Jiang et al., 2023; Cheng et al., 2024), but struggled to encode structure and semantics (Zhong et al., 2025) and to scale to modern KGs with billions of triples (Feng et al., 2025).

To this end, recent work has explored exact retrieval methods for knowledge graphs, aiming to translate natural language questions into formal KG queries to produce accurate and interpretable answers (query generation) (Su et al., 2016; Chen et al., 2020; Li et al., 2023). While early KG methods only generated simplified logical forms for small-scale graphs (Berant and Liang, 2014; Talmor and Berant, 2018; Nie et al., 2022), recent work has leveraged LLMs to produce complex languages such as SPARQL (Text-to-SPARQL) with large-scale KGs like Freebase and Wikidata (Kovriguina et al., 2023; Su et al., 2024; D'Abramo et al., 2025). With Freebase (Bollacker et al., 2008)—long the most widely used KG for KBQA—shut down since 2015, Wikidata has become the dominant resource for LLM-based research in this area (Xu et al., 2023; Liu et al., 2024b). However, unlike Freebase, Wikidata is a RDF KG, whose oversized schemas, opaque resource identifiers, and redundant relation types exceed LLM context limits and hinder query generation performance (Feng et al., 2025). Moreover, reliance on RDF diverges from industry practices, where property graphs and the Cypher query language (e.g., Neo4j, Amazon Neptune) (Francis et al., 2018) are increasingly preferred for their compact, human-readable schemas. These schemas also align more naturally with the reasoning in leading LLM frameworks such as Langchain and LLamaIndex (Feng et al., 2025).

Given these limitations, advancing Text-to-Cypher query generation – the task of translating natural language questions into Cypher queries – is crucial for supporting practical KG work-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 34665–34682

flows. For instance, given the question "*Who acted in the movie Forrest Gump?*", a Text-to-Cypher system should generate the query: "`MATCH (movie:Movie {title: "Forrest Gump"})<-[:ACTED_IN]-(actor:Person) RETURN actor.name AS actors`". Recent efforts have begun to generate datasets and frameworks for fine-tuning and evaluating LLMs on this task, laying an initial foundation for progress. However, existing instruction-tuning datasets for Text-to-Cypher have mostly utilized rigid templates, heuristic conversions, or costly proprietary LLMs for data generation (Zhao et al., 2023a; Ozsoy et al., 2025; Feng et al., 2025; Zhong et al., 2025; Tiwari et al., 2025), leading to training datasets that lack scale, diversity, and robustness. Moreover, many current Text-to-Cypher datasets are confined to specific domains, limiting their applicability across broader real-world scenarios. Consequently, models trained on such data often underperform on evaluation benchmarks, highlighting the need for methods to develop larger, higher-quality datasets for fine-tuning LLMs on Text-to-Cypher.

To address these challenges, we introduce CypherSmith, a novel approach for creating instruction-tuning data for Text-to-Cypher. Our method offers two key advantages over existing datasets. First, instead of relying on templates, heuristic conversions, or costly proprietary LLMs, CypherSmith fully leverages open-source LLMs to generate larger, more diverse, and higher-quality datasets that enhance Text-to-Cypher performance. To achieve broad domain coverage, CypherSmith uses LLMs to convert database schemas from the large-scale Text-to-SQL dataset SynSQL-2.5M (Li et al., 2025) into ontologies for property KGs. This approach allows our dataset to inherit a wide range of domains from SynSQL-2.5M, providing a strong and general foundation for fine-tuning robust LLMs. In addition, LLMs are used to generate reasoning annotations for each example, providing richer training signals in CypherSmith. To our knowledge, CypherSmith is the first Text-to-Cypher dataset to include reasoning data to support model development.

Second, to improve the quality of the generated data, we introduce a novel filtering framework to remove redundant or noisy examples. Our method involves an automatic validation of query syntax and alignment with ontologies. Importantly, we propose a likelihood-based filtering criteria that assesses each generated Cypher query given its ontology, question, and reasoning context. This ensures that each element, i.e., ontology, question, and reasoning, effectively contributes to query generation, resulting in a more compact and cleaner dataset. As such, unlike previous data filtering method for Text-to-Cypher (Zhong et al., 2025), our method does not require access to actual knowledge graphs or query execution, yet achieves larger and higher-quality datasets. To our knowledge, this is also the first work to leverage LLM likelihood scores to filter synthetic data for Text-to-Cypher.

We conduct extensive experiments on two recent evaluation benchmarks, Text2Cypher (Ozsoy et al., 2025) and CypherBench (Feng et al., 2025). The results demonstrate that our dataset substantially improves LLM fine-tuning for Text-to-Cypher.

2 Related Work

We categorize our related work into four major categories: Knowledge Base Question Answering (KBQA), query generation, Text-to-Cypher, and instruction data selection.

KBQA: Our work contributes to the general problem of KBQA, which answers natural language questions by reasoning over KBs such as Freebase and Wikidata (Yih et al., 2016; Berant et al., 2013; Usbeck et al., 2018; Moon et al., 2019; Dubey et al., 2019; Gu et al., 2021; Cao et al., 2022). A major approach retrieves KB elements via k-hop neighborhoods (Oguz et al., 2022; Jiang et al., 2023; Luo et al., 2024; Cheng et al., 2024) or similarity-based embeddings (Ma et al., 2022; Baek et al., 2023; Yu et al., 2023; Wu et al., 2024). However, embedding cost and KG/question complexity hinder scalability and semantic accuracy of the retrieval approach.

Query Generation: Query generation offers an alternative for KGQA by converting natural questions into queries executable over KBs to retrieve exact information. Most research has focused on Text-to-SQL, driven by the dominance of relational databases in industry (Wang et al., 2020; Zhang et al., 2023; Pourreza et al., 2024; Wang et al., 2025; Li et al., 2025). For query generation over KGs, early methods relied on custom logical forms with simplified syntax, limiting them to small-scale graphs and simple questions (Su et al., 2016; Berant and Liang, 2014; Talmor and Berant, 2018; Nie et al., 2022; Cao et al., 2022). With the advent of LLMs, recent research has shifted to more expressive languages such as SPARQL (Text-to-SPARQL) and to large-scale KGs (Banerjee et al.,

2022; Kovriguina et al., 2023; Su et al., 2024; Xu et al., 2023; Liu et al., 2024b; D'Abramo et al., 2025). As discussed in the introduction, this line of work has largely adopted Wikidata due to its active maintenance; however, it introduces significant challenges for LLM fine-tuning due to the schema-related issues of an RDF graph.

Text-to-Cypher: To better align with industry standards, some recent work has leveraged property graphs with compact and informative schemas, using Cypher queries (Guo et al., 2023; Xu et al., 2024; Feng et al., 2025). However, a major challenge for Text-to-Cypher research is the lack of accessible knowledge graphs and datasets to enable LLM development and evaluation. To this end, existing work has mainly focused on generating synthetic Text-to-Cypher datasets and KGs: CySpider (Zhao et al., 2023a,b) maps SQL-based datasets into Cypher with a rule-based approach but struggles with diverse domain extension; SyntheT2C (Zhong et al., 2025) and (Tiwari et al., 2025) utilize pipelines with GPT-4o, template-filling, and predefined query types and domains, but incurs high costs for proprietary LLMs and human labor to verify generated queries. They also cannot be applied to our synthetic data due to the scale and diversity of generated data, and the lack of actual knowledge graphs for verification in the pipelines; CypherBench (Feng et al., 2025) builds 11 Wikidata-based property graphs but restricts diversity of Cypher queries to 12 predefined patterns; and Text2Cypher (Ozsoy et al., 2025) unifies existing Text-to-Cypher datasets from different sources but inherits their limited domains, inconsistent formulation, and scalability challenges. Our CypherSmith overcomes these limitations by fulling leveraging open-source LLMs with likelihood-based filtering to enable the generation of a substantially larger and more diverse Text-to-Cypher dataset.

Instruction Data Selection: The goal is to identify a compact subset of an original dataset that preserves or enhances the performance of LLMs fine-tuned on the full dataset, while reducing computation cost (Liu et al., 2025). Data quality and diversity serve as the key quantities to drive the development of selection methods for LLMs (Zhou et al., 2023; Chen et al., 2024). Existing work has estimated the quality of instruction data using human-designed features with Instruction-Mining (Cao et al., 2023), differences between one-shot and zero-shot performance with Nuggets (Li et al., 2024c), gradient features with LESS (Xia et al., 2024), intrinsic uncertainty of models with SelectIT (Liu et al., 2024a), model loss disparity for different contexts with IFD (Li et al., 2024b), or proprietary LLMs' predictions with AlpaGasus (Chen et al., 2024). In contrast, diversity is measured via embedding distances (Wu et al., 2023), facility location (Bukharin et al., 2024), or determinant-based functions (Wang et al., 2024). To balance both, recent work combines quality and diversity, e.g., *#InsTag* with semantic and intent tags (Lu et al., 2024), DEITA with fine-tuned LLM scoring (Liu et al., 2024c), CaR with clustering (Ge et al., 2024), and MIG with label graphs (Chen et al., 2025). However, no prior work investigates instruction selection for synthetic datasets enriched with reasoning in Text-to-Cypher, as we do.

3 Method

In our framework, Text-to-Cypher is formulated as a sequence-to-sequence task. The model input includes an ontology o, which defines entities along with their properties and relationships in a KG. Each entity's properties might be associated with a data type and/or example values, while relationships link entities through relation labels. In addition, a natural language question q and a task instruction i are provided as part of the input. The goal is to generate a Cypher query c that retrieves the information requested by the question from the KG. To support reasoning, the model can also be prompted to produce an intermediate reasoning text r before generating the query, leading to the modeling of the probability $P(r, c|i, o, q)$ for instruction tuning. In this work, we use the same task instruction i for all examples as illustrated Figure 2 of Appendix A. We present a two-step framework to generate instruction data for Text-to-Cypher in this formulation, including data synthesis and filtering.

3.1 Data Synthesis

To improve robustness across domains for Text-to-Cypher, we aim to to design instruction-tuning tuples (i, o, q, r, c) that can cover diverse domains in the ontologies o and questions q. As the ontologies in existing Text-to-Cypher datasets are constrained to only a small number of domains, we propose to leverage the diverse database schemas in Text-to-SQL datasets, which have already proven to significantly enhance model performance. In particular, we build on SynSQL-2.5M (Li et al., 2025), a large-scale dataset containing 2,544,390

examples across 16,583 domains. Each example includes a database schema, a natural language question, a SQL query, and reasoning text, enabling robust LLM fine-tuning with significant performance improvement for Text-to-SQL. For our task, we prompt an LLM to convert each database schema into an equivalent KG ontology, where tables map to entities, columns to properties, and foreign keys to relations. To enhance data diversity, we randomly select one of four common ontology formats—class lists, structured YAML, Python Dictionary, or Neo4j constraints—and enforce this format in the prompt for each example. Our full conversion prompt is provided in Appendix A. Note that unlike prior rule-based table-to-ontology conversions, which are only designed for specific domains (Zhao et al., 2023a,b), our LLM-based approach can flexibly handle database schemas in different domains and syntax variants.

Given an ontology o converted from SynSQL-2.5M, the next step is to generate a Cypher query c, a natural question q, and a reasoning text r. In our prompting strategy, the LLM is first instructed to produce a Cypher query c that reflects realistic use of the ontology. The corresponding natural-language question q and reasoning text r are then generated based on this query and the ontology. Following insights from Text-to-SQL research (Zhang et al., 2023; Li et al., 2024a, 2025), this approach improves data quality: generating questions from Cypher queries is more reliable than the reverse, as the precise and well-structured nature of Cypher queries makes it easier for LLMs to produce accurate verbalizations.

To balance complexity and diversity, we design eight levels of query complexity and instruct the LLM to follow one level per example. These levels range from simple single-node filters to advanced cases such as multi-hop traversal, relationship aggregation, subqueries with nested reasoning, and structural or temporal reasoning. To further promote linguistic variety, we also define seven question types and prompt the LLM accordingly, covering forms such as direct questions, imperative commands, elliptical queries, descriptive goals, filtering requests, comparative aggregations, and yes/no questions. For reasoning, the prompt requires step-by-step breakdowns to support accurate query generation from the question. The complexity levels, question types, and ontology formats for our dataset are provided in Appendix A.

Finally, for each SynSQL-2.5M example, we generate the ontology, query, question, and reasoning in a single process to improve efficiency and enable context conditioning across components. We utilize the open-source LLM Qwen2.5-32B-Instruct for data generation.

3.2 Data Filtering

From SynSQL-2.5M, we obtain 2,544,390 instruction-tuning tuples (i, o, q, r, c) for Text-to-Cypher through the previous generation step. However, this raw dataset inevitably contains noise arising from LLM hallucinations and generation errors. To enhance data quality and reduce the computational cost for instruction tuning, we design a two-stage filtering process that prunes syntactically invalid and semantically inconsistent examples. Compared to previous Text-to-Cypher datasets (Zhong et al., 2025; Feng et al., 2025), which rely on executing queries against knowledge graphs for data cleaning, our setting is more challenging as no underlying knowledge graphs exist for the generated ontologies.

Syntax Validation: In the first step, we use the Neo4j library[1] to validate the syntax of the generated Cypher queries. This library allows syntax checking without executing the queries over knowledge graphs, making it well-suited for our generated dataset. This step eliminates 513,648 examples, leaving 2,040,742 for further processing.

In the next step, our objective is to guarantee ontological consistency and precise query–ontology alignment, thereby avoiding mismatches in declared labels and variables. To this end, we systematically parse and analyze the generated queries and ontologies, verifying that every referenced entity, property, and relationship (including names and triples) is explicitly defined within the ontologies. Furthermore, we ensure that all variables in the queries are properly introduced and access only valid properties. This alignment check filters out an additional 141,530 examples, leading to a dataset U of 1,899,212 syntactically verified examples.

Semantic Filtering: A distinctive feature of our dataset is the inclusion of reasoning texts to facilitate the transformation of natural language questions into Cypher queries. However, as with natural questions, verifying the correctness of reasoning texts requires semantic understanding to detect inconsistencies and noise in the generated data. To assess data quality and motivate semantic filtering,

[1] https://pypi.org/project/neo4j

#	Error	%
1	**Question and ontology sufficiency:** the question provides clear and direct information to generate the query from the ontology, making further reasoning unnecessary.	46.5
2	**Misaligned query and question:** the target query does not correspond well to the natural language question.	22.5
3	**Misaligned reasoning and question:** the reasoning interprets the question differently, leading to a mismatch between the two.	13.0

Table 1: Error analysis on 200 sampled examples. An example might exhibit multiple error types.

we analyze 200 sampled examples from U. Table 1 summarizes the major error types, while Table 10 provides illustrative examples and explanations. The most prevalent issues involve misalignment between questions, queries, and reasoning texts, as well as redundancy in reasoning.

Such misalignment and redundancy introduce noise and confusion into the model, leading to reduced confidence and potentially incorrect query generation. Since ground-truth queries are not available for the generated tuples (i, o, q, r, c), we instead propose to evaluate the contribution of each context element—ontology o, question q, and reasoning r—to the model's ability to generate the query c. The key intuition is that if removing a context element $e \in o, q, r$ does not reduce the model's likelihood of generating c, then e provides little or no value, suggesting redundancy or noise in the generated tuples (i, o, q, r, c) with full context information. Conversely, when all context elements are informative, the full context should produce a higher generation probability than any reduced context, thereby strengthening model confidence through the presence of relevant information.

To formalize this intuition, we measure the helpfulness of each context element $e \in \{o, q, r\}$ by comparing generation probabilities. Specifically, we compute $P(c|i, o, q, r)$ using the complete context and $P(c|(i, o, q, r) \setminus e)$ when element e is excluded. A context element e is considered helpful if its removal decreases the generation probability: $P(c|(i, o, q, r) \setminus e) < P(c|i, o, q, r)$. Based on this criterion, we filter out any tuple (i, o, q, r, c) where the full context probability fails to exceed all reduced context probabilities:

$$P(c|i, o, q, r) < \min\{P(c|i, q, r), P(c|i, o, r), P(c|i, o, q)\}.$$

To implement this strategy, we first sample a subset $S \subset U$ of the remaining instruction tuples, ensuring uniform coverage across ontology formats, query complexities, and question types. We then fine-tune the target LLM L on this sampled data, which is subsequently used to compute the likelihoods $P(c|i, o, q, r)$ and $P(c|(i, o, q, r) \setminus e)$ for filtering the remaining examples in $U \setminus S$. To achieve reliable likelihood estimation, the fine-tuning process includes not only the full-context examples in S but also their corrupted variants, where one context element (o, q, or r) is excluded. In this way, our data filtering method is specifically customized for the target LLM L, as it leverages the model's own internal likelihoods for data selection. This step results in 518,730 examples for our final dataset CypherSmith. Figure 1 shows the distributions of query complexity levels, question types, and ontology formats in CypherSmith.

4 Experiments

We evaluate CypherSmith on two recent Text-to-Cypher benchmark datasets: Text2Cypher (Ozsoy et al., 2025) and CypherBench (Feng et al., 2025). Text2Cypher aggregates data from multiple sources, providing 39,554 training examples and 4,833 test examples. Among the test set, 2,471 queries are linked to a Neo4j knowledge graph for execution-based evaluation. This benchmark includes 16 KGs accessible via the Neo4j API[2]. CypherBench contains 8,534 training examples and 2,488 test examples, which are generated from sampled graph patterns across 11 KGs. We adopt the original KGs in CypherBench for our evaluation.

Evaluation Metrics: To ensure compatibility, we adopt the evaluation metrics and prompts defined in the original papers for each dataset. Specifically, Text2Cypher evaluates model performance using Google BLEU scores between predicted and gold queries, as well as Exact Match Accuracy (EX-A) over execution results returned from KGs for the queries. Accordingly, Google BLEU scores are computed for all test examples in Text2Cypher while EX-A is only done over test queries where a Neo4j knowledge graph is provided for execution.

In addition, we note that the original EX-A scores in Text2Cypher (Ozsoy et al., 2025) are computed over the raw tabular results of the predicted and gold queries, including the ordered column names and their corresponding values. However, this can underestimate model performance, as models might use different variables or aliases for outputs while still returning correct values. For

[2] neo4j+s://demo.neo4jlabs.com

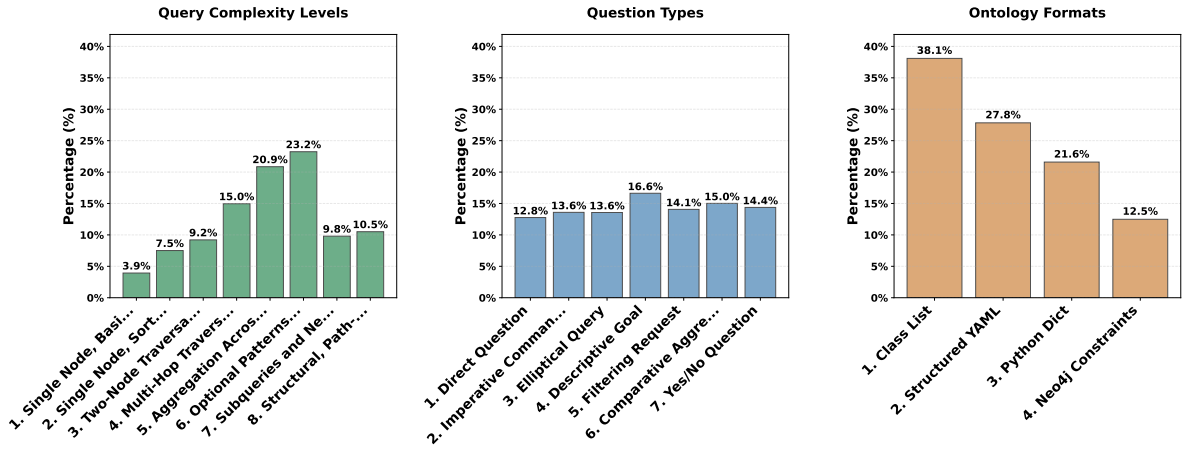

Figure 1: Distributions of query complexity levels, question types, and ontology formats in CypherSmith.

example, in a KG of cities with countries and populations, the gold query "`MATCH (c:City) WHERE c.population > 1000000 RETURN c.name AS city, c.country AS country`" and the predicted query "`MATCH (c:City) WHERE c.population > 1000000 RETURN c.name AS city, c.country AS nation`" both return the same underlying data, but the alias for the country differs ("`country`" vs "`nation`"). A strict EX-A evaluation would mark the predicted query as incorrect, even though it produces correct results. This strict EX-A evaluation differs from standard practices in CypherBench and Text-to-SQL research (Wang et al., 2020; Li et al., 2025), which typically evaluate only the returned values, ignoring column names. To address this, we report an additional evaluation metric for Text2Cypher that compares only the sorted returned values from predicted and gold queries (denoted EX) in our experiments.

Finally, CypherBench reports EX scores, Provenance Subgraph Jaccard Similarity (PSJS)—which measures the similarity of provenance subgraphs between predicted and gold queries—and the executable percentage (Exec) of predicted queries (Feng et al., 2025).

4.1 Evaluation

We use a publicly available LLM with fewer than 10B parameters as the target base model for fine-tuning. In particular, we perform data generation and filtering with LLM, and then fine-tune the model on the final dataset CypherSmith using the standard auto-regressive loss to generate reasoning texts and Cypher queries given instructions, ontologies, and questions. Following previous work (Ozsoy et al., 2025), we include the training data of each dataset—Text2Cypher and CypherBench—in its respective fine-tuning process. The implementation details are presented in Appendix B.

Model	BLUE	EX-A	EX
Proprietary LLMs			
gpt-5	59.3	22.6	49.2
gpt-4o	62.4	27.9	51.8
gpt-3.5-turbo	50.8	19.1	36.1
gpt-4o-mini	54.6	22.0	39.0
Open-source LLMs			
Qwen2.5-72B-Instruct	53.6	17.9	39.9
Qwen2.5-32B-Instruct	56.7	19.7	42.5
Qwen2.5-Coder-32B-Instruct	58.9	22.9	43.8
Qwen2.5-14B-Instruct	52.2	17.6	35.4
Qwen2.5-Coder-14B-Instruct	57.9	22.7	40.9
Fine-tuned LLMs			
LLM (base)	38.9	10.8	17.8
LLM + Text2Cypher	72.7	29.3	51.4
LLM + CypherSmith	**77.6**	**30.9**	**53.9**

Table 2: Performance on Text2Cypher's test set.

Table 2 and 3 report models' performance on the test datasets of Text2Cypher and CypherBench respectively. We also include the performance of typical proprietary and large-scale open-source LLMs for reference. As shown in the tables, GPT-5 and GPT-4o achieve the best performance among both proprietary and open-source models. Table 2 further reveals substantial gaps between EX-A and EX scores, thus demonstrating the importance of reporting EX for a more accurate evaluation on Text2Cypher. Notably, LLM fine-tuned with CypherSmith not only outperforms the base model but also surpasses variants trained on Text2Cypher and CypherBench, clearly demonstrating the value of our generated dataset in upholding the Text-to-Cypher foundation for LLMs. Moreover, our fine-

Model	EX	PSJS	Exec
Proprietary LLMs			
gpt-5	64.8	**82.0**	93.1
gpt-4o	60.2	76.9	94.9
gpt-3.5-turbo	26.3	37.0	92.3
gpt-4o-mini	31.4	45.9	87.4
Open-source LLMs			
Qwen2.5-72B-Instruct	41.9	56.4	86.8
Qwen2.5-32B-Instruct	36.2	56.4	78.5
Qwen2.5-Coder-32B-Instruct	42.7	55.3	84.4
Qwen2.5-14B-Instruct	25.9	42.7	80.4
Qwen2.5-Coder-14B-Instruct	43.1	58.5	91.9
Fine-tuned LLMs			
LLM (base)	18.8	31.0	90.7
LLM + CypherBench	34.2	46.3	86.0
LLM + CypherSmith	**74.8**	77.5	**99.2**

Table 3: Performance on CypherBench's test set.

tuned model matches or exceeds the performance of proprietary systems and open-source models with over 70B parameters, emphasizing the critical role of fine-tuning in advancing Text-to-Cypher.

Dataset	Text2Cypher			CypherBench		
	BLUE	EX-A	EX	EX	PSJS	Exec
Text2Cypher	72.7	29.3	51.4	55.8	64.0	97.4
CypherBench	71.7	29.7	51.2	34.2	46.3	86.0
SyntheT2C	75.4	29.7	51.8	51.5	65.4	87.2
CySpider	**78.5**	29.1	50.9	56.4	63.1	94.8
SynthCypher	71.2	30.0	52.9	62.6	67.2	97.3
CypherSmith	77.6	**30.9**	**53.9**	**74.8**	**77.5**	**99.2**

Table 4: Performance on the test sets of Text2Cypher and CypherBench after fine-tuning on different datasets.

Dataset	#Examples	#Ontos	#Doms	#Ents	#Props	#Rels
Text2Cypher	39,554	966	120	4.6	21.9	7.5
CypherBench	8,534	4	4	5.3	8.9	5.9
SyntheT2C	3,000	2	2	9.0	38.5	17.5
CySpider	4,929	155	155	5.1	23.3	1.6
SynthCypher	25,828	528	528	**21.8**	**72.9**	**19.0**
CypherSmith	**518,730**	**518,358**	**6,957**	9.9	38.8	10.7

Table 5: Ontology statistics of the datasets. **#Ontos** and **#Doms** denote the numbers of unique ontology texts and domains, while **#Ents**, **#Props**, and **#Rels** represent the average numbers of entities, properties, and relations per ontology.

An additional finding worth highlighting appears in Tables 2 and 3: Qwen2.5-72B-Instruct is outperformed by its smaller counterpart Qwen2.5-32B-Instruct for Text2Cypher, and Qwen2.5-Coder-32B-Instruct falls behind Qwen2.5-Coder-14B-Instruct for CypherBench. Our analysis reveals potential reasons for this phenomenon with implicit reasoning and overgeneralization. In particular, larger models tend to apply high-level analytical heuristics learned from broad pretraining, e.g., generating intermediate reasoning steps, restructuring queries, or inferring alternative schema. Smaller models, by contrast, adhere more closely to the template-level pattern-matching required for Text2Cypher and CypherBench tasks. This stricter template compliance makes smaller models more robust on highly structured, schema-constrained query generation in our evaluation datasets.

4.2 Dataset Comparison

To further highlight the advantages of CypherSmith, we compare it with previous datasets for Text-to-Cypher, including Text2Cypher (Ozsoy et al., 2025), CypherBench (Feng et al., 2025), SyntheT2C (Zhong et al., 2025), CySpider (Zhao et al., 2023a), and SynthCypher (Tiwari et al., 2025). Table 4 reports the performance of LLM on Text2Cypher and CypherBench when fine-tuned on different datasets. The results clearly show that CypherSmith yields substantially larger execution gains than the alternatives, achieving up to 4.1% and 32.6% relative improvements in EX score on Text2Cypher and CypherBench, respectively. These demonstrate the effectiveness of CypherSmith and LLM-based synthetic data generation for Text-to-Cypher.

Beyond performance, we examine characteristics of the datasets in Tables 5 and 6. As shown in Table 5, CypherSmith introduces significantly more unique ontology texts and domains than other datasets, providing a stronger foundation for generalization across domains. It also incorporates more entities, properties, and relations (with the exception of relations in SyntheT2C) to enable richer and more diverse ontologies for robust model training. For Cypher queries (Table 6), CypherSmith consistently feature more entities, properties, relations, functions, and clauses. In addition, CypherSmith's queries contains both a greater number and a higher proportion of unique query skeletons (i.e., core structural patterns), thus demonstrating greater structural diversity for the models.

4.3 Data Selection Comparison

This section examines the benefits of the likelihood-based data filtering principle $P(c|i,o,q,r) < \min\{P(c|i,q,r), P(c|i,o,r), P(c|i,o,q)\}$ for the generated tuples (i,o,q,r,c) in our data generation method for Text-to-Cypher. In particular, we study the performance of LLM when fine-tuned on ablated variants of this criterion, where each of the probability components—$P(c|i,q,r)$, $P(c|i,o,r)$, or $P(c|i,o,q)$—is removed to assess the contribu-

Dataset	#Examples	#Queries	%Skels	#Toks	#Ents	#Props	#Rels	#Agg Funcs	#Funcs	#OPT	#WITH
Text2Cypher	39,554	37,306	24%	41.2	2.7	2.3	1.7	0.3	0.6	0.0	3.7
CypherBench	8,534	8,534	2%	54.2	3.3	2.9	3.2	0.2	0.2	0.0	3.8
SyntheT2C	3,000	2,166	4%	35.3	2.3	2.3	1.7	0.1	0.1	0.0	3.2
CySpider	4,929	2,740	9%	30.0	1.9	3.6	0.4	0.4	0.4	0.0	3.0
SynthCypher	25,828	25,430	40%	46.8	3.0	3.0	2.6	0.3	0.5	0.0	0.5
CypherSmith	**518,730**	**517,636**	**50%**	**81.3**	**5.3**	**5.0**	**5.2**	**0.9**	**0.9**	**0.3**	**5.3**

Table 6: Query statistics of the datasets. **#Queries** denotes the number of unique Cypher queries. **#Toks**, **#Ents**, **#Props**, and **#Rels** represent the average numbers of tokens, entities, properties, and relations per query, respectively. **#Agg Funcs**, **#Funcs**, **#OPT**, and **#WITH** indicate the average numbers of aggregate functions (e.g., `COUNT`, `AVG`), functions (e.g., `FILTER`, `SUBSTRING`), `OPTIONAL MATCH` clauses, and `WITH` clauses per query. **%Skels** reports the proportion of unique query skeletons within each dataset. Query skeletons are constructed by replacing all specific identifiers, values, and literals in the original query with [MASK] tokens, while preserving the structural pattern of nodes, relationships, return clauses, and keywords. For example, given the Cypher query "`MATCH (a:Author)-[:WROTE]->(b:Book) RETURN a.name, b.title LIMIT 10`", its skeleton is "`MATCH ([MASK])-[[MASK]]->([MASK]) RETURN [MASK].[MASK], [MASK].[MASK] LIMIT [MASK]`".

tion of ontology o, question q, and reasoning r for query generation. We also report results from a dataset without likelihood-based filtering (**No filtering**). To understand the benefits of reasoning information r in CypherSmith, we include model performance from a variant excluding reasoning texts during fine-tuning (**No reasoning**).

Data	Text2Cypher			CypherBench		
Selection	BLUE	EX-A	EX	EX	PSJS	Exec
CypherSmith	77.6	30.9	53.9	74.8	77.5	99.2
$-P(c\|i,q,r)$	77.5	29.7	52.8	72.2	75.7	98.7
$-P(c\|i,o,r)$	73.0	30.1	52.2	71.7	74.7	99.2
$-P(c\|i,o,q)$	76.6	29.9	53.1	72.7	75.3	98.2
No filtering	73.9	29.7	52.4	72.9	75.7	99.2
No reasoning	70.5	30.6	52.4	69.6	73.0	98.7
Random	75.0	29.6	52.4	70.1	72.9	98.8
Max Length	73.2	30.6	53.6	71.5	74.9	98.3
Diversity	77.2	30.0	52.3	71.5	74.8	97.1
Highest Prob	74.8	29.9	52.4	67.3	70.8	98.1
IFD	76.5	29.8	52.0	70.4	74.3	98.0
#InsTag	76.6	30.0	52.2	52.9	68.4	84.0
CaR	76.8	30.2	52.7	68.2	72.6	98.1
DEITA	77.2	30.2	53.1	69.4	73.3	98.7
MIG	77.0	29.7	52.5	72.0	75.1	99.0

Table 7: Performance on the test sets of Text2Cypher using different data filtering methods. Our full likelihood criterion for CypherSmith is $P(c|i,o,q,r) < \min\{P(c|i,q,r), P(c|i,o,r), P(c|i,o,q)\}$.

In addition, we compare our approach against nine instruction data selection baselines for LLMs: **(i) Random** randomly selects generated tuples; **(iii) Max Length** chooses examples with the longest queries; **(ii) Diversity** clusters the full dataset and uniformly samples examples across clusters; **(iv) Highest Prob** selects examples with the highest likelihood $P(c|i,o,q,r)$; **(v) IFD** (Li et al., 2024b) selects examples with highest Instruction-Following Difficulty scores; **(vi) #InsTag** (Lu et al., 2024) uses a ChatGPT-distilled model to tag examples with fine-grained semantic and intent labels, selecting those with the most tags (highest complexity); **(vii) DEITA** (Liu et al., 2024c) combines predictions from complexity and quality scoring LLMs with clustering for selection; **(viii) CaR** (Ge et al., 2024) ranks examples with an expert-aligned quality model, then clusters to select the top N; and **(ix) MIG** (Chen et al., 2025) leverages semantic label graphs and information gain maximization for selection. All baseline methods sample the same number of examples as CypherSmith.

Table 7 shows LLM performance when fine-tuned on data selected by different methods. Our likelihood-based filtering in CypherSmith consistently surpasses all ablated variants and baseline methods. Among the probability terms, $P(c|i,o,r)$ is the most critical: removing it causes the largest performance drop, confirming the necessity of conditioning on the input question to produce accurate queries. In contrast, the baseline approaches underperform as they are generic data selection methods that cannot exploit the distinctive context provided by ontologies, questions, and reasoning in our generated data. Moreover, they ignore the internal states of LLM (e.g., likelihoods), which are essential for guiding data selection. The superior performance of CypherSmith demonstrates the value of explicitly modeling the contribution of each context component to select optimal data for Text-to-Cypher instruction tuning. Finally, excluding reasoning information significantly degrades performance on both datasets, highlighting the importance of reasoning texts in CypherSmith.

4.4 Generalization

To explore the generalization of our generated data for Text-to-Cypher models, we evaluate the performance of additional base LLMs fine-tuned on CypherSmith. Specifically, we fine-tune two other LLMs on CypherSmith, along with the training sets

of Text2Cypher and CypherBench: (1) DiffArc-Small, a model with fewer than 10B parameters and a different architecture, and (2) SameArc-Large, a larger model (>13B parameters) sharing the same architecture as LLM. Model performance is reported on the test sets of Text2Cypher in Table 8 and CypherBench in Table 9.

Model	BLUE	EX-A	EX
DiffArc-Small + Text2Cypher	65.2	27.8	50.3
DiffArc-Small + CypherSmith	**71.9**	**28.4**	**51.3**
SameArc-Large + Text2Cypher	70.1	26.8	45.4
SameArc-Large + CypherSmith	**76.9**	**28.3**	**49.7**

Table 8: Performance on Text2Cypher's test set.

Model	EX	PSJS	Exec
DiffArc-Small + CypherBench	63.1	70.2	95.3
DiffArc-Small + CypherSmith	**69.6**	**74.6**	**98.0**
SameArc-Large + CypherBench	43.3	59.9	80.0
SameArc-Large + CypherSmith	**61.0**	**68.2**	**99.5**

Table 9: Performance on CypherBench's test set.

The results show that CypherSmith significantly improves the performance of both DiffArc-Small and SameArc-Large on Text2Cypher and CypherBench, consistently yielding large gains across evaluation metrics. As CypherSmith is filtered using the likelihoods from LLM, these findings demonstrate the generalization ability of our generated dataset across LLMs with different architectures and scales, establishing CypherSmith as a strong foundation to boost Text-to-Cypher performance.

5 Conclusion

This paper presents a new framework for generating synthetic instruction data for Text-to-Cypher. Our approach fully leverages open-source LLMs to produce large-scale, high-quality data enriched with reasoning information. We introduce a novel filtering method that exploits the contributions of individual context components to the model's generation likelihoods, effectively removing noisy and redundant examples. The resulting dataset, CypherSmith, is substantially larger than previous resources, delivers significant performance gains for LLMs, and establishes a strong foundation for future research. Looking ahead, we plan extend CypherSmith to further enhance both dataset quality and model performance on Text-to-Cypher.

Limitations

While our work introduces a large-scale, high-quality dataset CypherSmith for Text-to-Cypher, we acknowledge three limitations that open promising directions for future research. First, the current dataset supports only English natural language questions. Extending to additional languages would broaden applicability and foster progress in multilingual Text-to-Cypher. Second, although CypherSmith yields significant performance improvements for Text-to-Cypher, there remains room to further enhance model performance for diverse real-world applications. Exploring new learning techniques and resources could drive further performance gains. Finally, although we evaluate the models on the most comprehensive benchmarks currently available, Text2Cypher and CypherBench are still built on relatively small knowledge graphs compared to industrial settings. Future work can thus benefit from developing larger, more realistic knowledge graphs and datasets to better reflect practical challenges.

References

Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023. Direct fact retrieval from knowledge graphs without entity linking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. 2022. Modern baselines for sparql semantic parsing. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*.

Alexander Bukharin, Shiyang Li, Zhengyang Wang, Jingfeng Yang, Bing Yin, Xian Li, Chao Zhang, Tuo Zhao, and Haoming Jiang. 2024. Data diversity matters for robust instruction tuning. In *Findings of the*

Association for Computational Linguistics: EMNLP 2024.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: Instruction data selection for tuning large language models. *ArXiv*, abs/2311.15653.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024. Alpagasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations*.

Yicheng Chen, Yining Li, Kai Hu, Ma Zerun, HaochenYe HaochenYe, and Kai Chen. 2025. MIG: Automatic data selection for instruction tuning by maximizing information gain in semantic space. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi. 2020. Formal query building with query structure prediction for complex question answering over knowledge base. *ArXiv*, abs/2109.03614.

Keyuan Cheng, Gang Lin, Haoyang Fei, Yuxuan Zhai, Lu Yu, Muhammad Asif Ali, Lijie Hu, and Di Wang. 2024. Multi-hop question answering under temporal knowledge editing. *ArXiv*, abs/2404.00492.

Jacopo D'Abramo, Andrea Zugarini, and Paolo Torroni. 2025. Investigating large language models for text-to-SPARQL generation. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *ArXiv*, abs/2404.16130.

Yanlin Feng, Simone Papicchio, and Sajjadur Rahman. 2025. CypherBench: Towards precise retrieval over full-scale modern knowledge graphs in the LLM era. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*.

Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Mahong Xia, Zhang Li, Boxing Chen, Hao Yang, Bei Li, Tong Xiao, and JingBo Zhu. 2024. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking. *ArXiv*, abs/2305.15066.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji rong Wen. 2023. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *ArXiv*.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023. FactKG: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Liubov Kovriguina, Roman Teucher, Daniil Radyush, and Dmitry Mouromtsev. 2023. Sparqlgen: One-shot prompt-based approach for sparql query generation. In *International Conference on Semantic Systems*.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025. Omnisql: Synthesizing high-quality text-to-sql data at scale. *ArXiv*, abs/2503.02240.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024a. Codes: Towards building open-source language models for text-to-sql. *Proc. ACM Manag. Data*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. From quantity to quality: Boosting

LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. 2024c. One-shot learning as instruction data prospector for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Liangxin Liu, Xuebo Liu, Derek F. Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024a. Selectit: Selective instruction tuning for llms via uncertainty-aware self-reflection. In *Neural Information Processing Systems*.

Shicheng Liu, Sina Semnani, Harold Triedman, Jialiang Xu, Isaac Dan Zhao, and Monica Lam. 2024b. SPINACH: SPARQL-based information navigation for challenging real-world questions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024c. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Ziche Liu, Rui Ke, Yajiao Liu, Feng Jiang, and Haizhou Li. 2025. Take the essence and discard the dross: A rethinking on data selection for fine-tuning large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

K. Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, and Chang Zhou. 2024. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *International Conference on Learning Representations*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open domain question answering with a unified knowledge interface. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Lunyiu Nie, Shulin Cao, Jiaxin Shi, Jiuding Sun, Qi Tian, Lei Hou, Juanzi Li, and Jidong Zhai. 2022. GraphQ IR: Unifying the semantic parsing of graph query languages with one intermediate representation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. UniK-QA: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

Makbule Gulcin Ozsoy, Leila Messallem, Jon Besga, and Gianandrea Minneci. 2025. Text2Cypher: Bridging natural language and graph databases. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan Ö. Arik. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. *ArXiv*, abs/2410.01943.

Chang Su, Jiexing Qi, He Yan, Kai Zou, and Zhouhan Lin. 2024. Enhancing sparql generation by triplet-order-sensitive pre-training. *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for QA evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

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. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*.

Ricardo Usbeck, Ria Hari Gusmita, Axel-Cyrille Ngonga Ngomo, and Muhammad Saleem. 2018. 9th challenge on question answering over linked data (qald-9) (invited paper). In *Semdeep/NLIWoD@ISWC*.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics*.

Peiqi Wang, Yikang Shen, Zhen Guo, Matt Stallone, Yoon Kim, Polina Golland, and Rameswar Panda. 2024. Diversity measurement and subset selection for instruction tuning datasets. *ArXiv*, abs/2402.02318.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. 2023. Self-evolved diverse data sampling for efficient instruction tuning. *arXiv preprint arXiv:2311.08182*.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. 2024. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *ArXiv*, abs/2404.13207.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*.

Huangchao Xu, Baohua Zhang, Zhong Jin, Tiannian Zhu, Quansheng Wu, and Hongming Weng. 2024. Enhancing large language models with domain-specific knowledge: The case in topological materials. volume abs/2409.13732v1.

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina Semnani, and Monica Lam. 2023. Fine-tuned LLMs know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over Wikidata. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *International Conference on Learning Representations*.

Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. 2023. Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems. *Proc. VLDB Endow.*

Ziyu Zhao, Wei Liu, Tim French, and Michael Stewart. 2023a. Cyspider: A neural semantic parsing corpus with baseline models for property graphs. In *AI 2023: Advances in Artificial Intelligence: 36th Australasian Joint Conference on Artificial Intelligence, AI 2023, Brisbane, QLD, Australia, November 28–December 1, 2023, Proceedings, Part II*.

Ziyu Zhao, Wei Liu, Tim French, and Michael Stewart. 2023b. Rel2graph: Automated mapping from relational databases to a unified property knowledge graph. *arXiv preprint arXiv:2310.01080*.

Zijie Zhong, Linqing Zhong, Zhaoze Sun, Qingyun Jin, Zengchang Qin, and Xiaofan Zhang. 2025. SyntheT2C: Generating synthetic data for fine-tuning large language models on the Text2Cypher task. In *Proceedings of the 31st International Conference on Computational Linguistics*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: less is more for alignment. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

A Data Generation Prompts

Figure 3 shows our data generation prompt for CypherSmith. The `ontology_format`, `query_complexity_level`, and {`question_type`} variables are chosen randomly among four formats for KG ontologies in Figure 4, eight query complexity levels in Figure 5, and seven question types in Figure 6, respectively. We also present the task instruction for CypherSmith in Figure 2.

B Implementation Details

We use Qwen2.5-32B-Instruct to generate the initial Text-to-Cypher dataset using the prompt shown in Figure 3. To enhance data diversity, we randomly select one of four temperature values (0.6, 0.7, 0.8, or 0.9) for each generation. We fine-tune LLM on the datasets to evaluate their quality. All models are fully fine-tuned for two epochs using a learning rate of $2e$-5 with the AdamW optimizer. The learning rate schedule includes a linear warmup for the first 5% of training steps, followed by cosine decay to 10% of the peak rate. Training is conducted on a p5.48xlarge EC2 instance equipped with 8 H100 GPUs (80GB each), using DeepSpeed ZeRO-3 and bf16 precision for distributed training. We set the batch size per device to 32, gradient accumulation steps to 2, and context length cutoff to 4000 tokens.

You are an expert in knowledge graphs, ontologies, and Cypher query generation.
Your task is to generate a Cypher query for a natural language question based on a given knowledge graph ontology.
Please also generate a step-by-step reasoning explanation before you generate the Cypher query.
Instructions:
You will receive:
An ontology defining node types, properties, and relationships in the knowledge graph.
A natural language question about the graph.
Query Requirements:
The query must accurately retrieve the information requested by the question.
Use only the exact class names, property names, and relationship types defined in the ontology.
Respect relationship directions as shown in the ontology.
The query must be complete, syntactically correct, and executable.
Use clear, meaningful variable names.
Reasoning Requirements:
Provide a step-by-step breakdown of how you interpret the question and map it to the ontology.
The reasoning should clarify the query construction process to ensure accuracy.
Now provide a step-by-step reasoning explanation and the Cypher query for the given question and ontology.

Figure 2: The task instruction i in our CypherSmith dataset.

Error	%	Example

Question and ontology sufficiency: the natural language question provides clear and direct information to generate the query from the ontology, making further reasoning unnecessary.	46.5%	**Generated Ontology**:
		`- abstract class: Entity`
		`properties:`
		`  INSTANCE_OF [N]!: Entity`
		`  name [0:1]: String`
		`. . .`
		`- class: Game « Entity`
		`properties:`
		`  publisher [0:1]: String`
		`  developer [0:1]: String`
		`  release_year [0:1]: Integer`
		`  console [0:1]: Console`
		`. . .`
		Generated Question: *Find the names of the top 10 Nintendo-developed games released after 2010, sorted by their release year.*
		Generated Cypher:
		`MATCH (g:Game) WHERE g.release_year > 2010 AND g.developer = 'Nintendo' RETURN g.name ORDER BY g.release_year ASC LIMIT 10`
		Generated Reasoning:
		1. Identify the main class of interest: Game.
		2. Apply filters to the Game nodes: release_year greater than 2010 and developer equal to 'Nintendo'.
		3. Select the game names to return.
		4. Order the results by the release year in ascending order.
		5. Limit the results to the top 10 entries.
		
		Explanation: The natural language question is clear and straightforward, providing all necessary information to construct the Cypher query without additional reasoning. The reasoning restates the question's requirements without adding new insights.

Misaligned query and question: the target query does not correspond well to the natural language question.	22.5%	**Generated Question:** *Which active users have administered doses in regions where the variant impact has a transmission rate greater than 1.5, and what is the total number of doses administered?* **Generated Cypher:** `MATCH (u:User status: 'active')-[:ADMINISTERED_DOSES] ->(cs:CoverageStatistic)-[:COVERED_BY_STRATEGY] ->(vs:VaccinationStrategy)-[:USED_VARIANT]->(vi:Variant Impact) WHERE vi.transmissionRate > 1.5 AND cs.dosesAdministered > 10000 RETURN u.username, vs.region, vi.variantName, SUM(cs.dosesAdministered) AS totalDosesAdministered ORDER BY totalDosesAdministered DESC` **Explanation:** The query includes a condition "cs.dosesAdministered > 10000" which is not mentioned in the question. The question asks for the total number of doses administered, but does not specify a threshold for doses administered.
Misaligned reasoning and question: the reasoning interprets the question differently, leading to a mismatch between the two.	13.0%	**Generated Question:** *Which users have generated the most reports and what is the average energy value they accessed on January 1, 2020?* **Generated Reasoning:** 1. Identify the users who have generated reports and count the number of reports each user has generated. 2. For each user, find the access logs where they accessed energy data on January 1, 2020. 3. Calculate the average energy value for the energy data accessed by each user on the specified date. 4. Return the user's email, role, the number of reports they have generated, and the average energy value they accessed. 5. Order the results by the number of reports generated in descending order and the average energy value in ascending order. 6. Limit the results to the top 10 users. **Explanation:** The reasoning suggests identifying users who have generated reports and counting them, but it does not clarify that the query should focus on users with the most reports. The reasoning also fails to emphasize that the energy data accessed should be specifically on January 1, 2020, which is a critical part of the question.

Table 10: Error analysis on 200 sampled examples. An example may exhibit multiple error types.

You are an expert in relational databases, knowledge graphs, and query generation using SQL and Cypher.
Your task is to convert a relational database schema into a knowledge graph ontology, then generate a valid Cypher query that reflects realistic use of that ontology in a Neo4j database. Follow this with a natural language question equivalent to the query, and a step-by-step reasoning text explaining how to derive the Cypher query from the question.
Instructions:
You will be given a database schema describing tables, columns, and relationships (e.g., foreign keys).
The converted ontology must be provided in the format specified by the `ontology_format` variable, and must be interpreted as a valid ontology for use in Cypher.
The generated query must follow the ontology and use all class, property, and relationship names as defined. Do not change names or add new ones.
The query should demonstrate typical graph operations (e.g., traversal, filtering, aggregation). It should use meaningful variable names and apply filters as appropriate.
The query must have the complexity level specified by the `query_complexity_level` variable.
The natural language question should faithfully reflect the semantics of the Cypher query. The question should follow the style specified in the `question_type` variable.
The reasoning text should break down the process of constructing the query from the question into smaller, simpler steps to facilitate understanding and accurate generation.
ontology_format: {ontology_format}
query_complexity_level: {query_complexity_level}
question_type: {question_type}
Database Schema:
{database_schema}
Your output must only contain the ontology, query, question, and reasoning without further information or explanations. The output should be returned in JSON format as follows:

```
{{
 "ontology": "text",
 "query": "text",
 "question": "text",
 "reasoning": "text"
}}
```

Figure 3: CypherSmith's generation prompt for Text-to-Cypher data.

1. Class List: A flat list of classes, value properties, and relationships. Example:

```
Classes:
 - Person
 - Book
Properties:
 - Person.name: string
 - Book.title: string
Relationships:
 - Person WROTE Book
```

2. Structured YAML: A structured format with inheritance, data types, property cardinality, and relationships expressed as properties. Example:

```
- abstract class: Entity
properties:
 INSTANCE_OF [N]!: Entity
 name [0:1]: String
- class: SoccerTeam « SportTeam
properties:
 HAS_HEAD_COACH [0:N]: Coach
 HAS_ASSISTANT_COACH [0:N]: Coach
```

3. Python Dict: Node types, property schemas, and relationship patterns using nested dictionaries. Example:

```
{"Person": {"name": "str"}, "Book": {"title": "str"},
"relationships": {"WROTE": {"from": "Person", "to": "Book"}}}
```

4. Neo4j Constraints: Node labels, property types, and relationship patterns in Neo4j Cypher. Example:

```
(:Person)
(:Book)
(:Person)-[:WROTE]->(:Book)
Person.name: STRING
Book.title: STRING
```

Figure 4: CypherSmith's ontology formats.

1. Single Node – Basic Filter: *Query a single class with simple filtering or projection. Pattern: One node type, 1–2 value properties, WHERE, and RETURN. Example: Find all Person nodes where age > 30.*
2. Single Node – Sorting, Aggregation, or Complex Filtering: *Query a single class and apply ordering, limiting, simple aggregation, or advanced value filters. Pattern: One node type with functions like COUNT, AVG, MAX, ORDER BY, LIMIT, or filters involving partial string matches and regular expressions. Example: Find the average salary of all Employee nodes. Find all Persons whose name starts with A.*
3. Two-Node Traversal – Basic Join: *Query two related classes via a single relationship. Pattern: Two node types, one relationship, optional filter on either side. Example: Find all Authors who wrote a Book published after 2010.*
4. Multi-Hop Traversal – Chain of Joins: *Traverse paths across 3 or more node types via multiple relationships. Pattern: at least 3 nodes, at least 2 relationships, filters along the path. Example: Find Students enrolled in Courses taught by Professors in the CS department.*
5. Aggregation Across Relationships: *Traverse and aggregate over related nodes or relationships. Pattern: COLLECT, DISTINCT, COUNT, grouped results per node. Example: For each Team, list all unique Players who scored in a Game.*
6. Optional Patterns and Conditional Logic: *Handle incomplete or alternative data using optional matches or logic. Pattern: OPTIONAL MATCH, OR, CASE, partial matches. Example: Find all Movies with a known Director or Producer.*
7. Subqueries and Nested Reasoning: *Use subqueries to compute intermediate values or nested logic. Pattern: CALL {{}} subqueries, computed filters, scoped aggregations. Example: Find Employees whose average project rating is above their department's average.*
8. Structural, Path-Based, or Temporal Reasoning: *Perform reasoning over graph structure, paths, or time. Pattern: Variable-length paths, shortest paths, temporal comparisons, node degree. Example: Find the longest collaboration chain between two Researchers.*

Figure 5: CypherSmith's query complexity levels.

1. Direct Question: *Phrase the question as a direct question, such as 'Who...', 'What...', 'Which...',*
2. Imperative Command: *Phrase the question as a command or instruction, such as 'List...', 'Find...', 'Show me...',*
3. Elliptical Query: *Use a short, fragment-style query omitting auxiliary verbs or subjects (e.g., 'Players on team'),*
4. Descriptive Goal: *Express the user's goal or intent in natural language, such as 'I want to find...' or 'Looking for...',*
5. Filtering Request: *Focus the question on filters or constraints, such as 'Only include...', 'Exclude those...', or 'With salary above 50K...',*
6. Comparative Aggregation: *Use comparisons or aggregations, such as 'Which team has the most wins?', 'Top 5 players by score',*
7. Yes/No Question: *Pose the question as a yes/no inquiry, such as 'Is John a coach?', 'Has the team won a championship?'.*

Figure 6: CypherSmith's question types.