CypherKD: Span-to-Context Grounding Distillation for Text-to-Cypher Generation

Anonymous ACL submission

Abstract

Text-to-Cypher generation maps a natural language question and a graph schema into an executable Cypher query. Although large language models can perform this translation effectively, deploying them for graph querying remains expensive, making compact distilled models attractive. Standard LLM distillation typically matches teacher and student output distributions at the logit level. This signal is only indirectly aligned with the structural grounding needed for Text-to-Cypher, where generated clauses, patterns, and expressions must be supported by the user question and valid schema elements. Such supervision can teach a compact student fluent Cypher syntax while leaving the teacher's links between Cypher query fragments and question-schema context underspecified. We introduce **CypherKD**, a distillation method that combines logit-based KD with a feature-based span-relation loss. CypherKD uses both gold targets and student-generated outputs for KD, extracts Cypher-aware spans including clauses, node patterns, triplets, and expressions, and aligns teacher and student span-context relation scores over the question-schema context. This feature-level objective encourages the compact model to approximate how the teacher associates query fragments with the relevant question and schema tokens. Experiments on Text-to-Cypher benchmarks show that CypherKD consistently improves execution accuracy and provenance subgraph similarity over token-level distillation baselines.

1 Introduction

Knowledge graphs (KGs) have become an important way to connect large language models (LLMs) with structured knowledge, helping improve factual accuracy and reduce outdated or hallucinated responses, especially in Graph Retrieval-Augmented Generation (GraphRAG) (Kim et al., 2023; Edge et al., 2024; Hu et al., 2025) and Knowledge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., 2023a; 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 (Zhang et al., 2022; Jiang et al., 2023b; Sanmartin, 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). LLMs have become a strong foundation for Text-to-Cypher because they can connect user intent with schema descriptions and compose multi-step query plans. As generators, they identify relevant labels, relationships, properties, filters, and return targets before arranging them into valid Cypher. As sources of supervision, they also support model training and data construction: fine-tuning on consolidated Text-to-Cypher data improves query generation (Ozsoy et al., 2025), SyntheT2C generates task-specific supervision for domain graphs (Zhong et al., 2025), and Auto-Cypher uses LLM-supervised generation and verification to construct synthetic Cypher data (Tiwari et al., 2025). These developments suggest that large LLMs learn useful links between questions and schemas, rather than only surface Cypher syntax.

The deployment question is how to transfer this behavior into compact models. Directly serving large LLMs is costly for interactive graph retrieval, where latency and memory footprint matter, while small models often lose the question-schema associations that support correct query generation. LLM distillation offers a natural solution. Existing methods can be grouped into output-level KD and feature-level KD. Output-level KD, often implemented as logit-based KD, trains the student to match the teacher's next-token distribution (Hinton et al., 2015). Within this group, off-policy KD compares the teacher and student under reference

prefixes, so the student learns to mimic the teacher on given sequences. On-policy KD instead lets the student generate prefixes first, and then uses the teacher to supervise those student-generated prefixes, reducing training-inference mismatch (Ko et al., 2024). Feature-level KD transfers internal representations or relations rather than only output logits (Sun et al., 2019; Jiao et al., 2020; Wang et al., 2020).

For Text-to-Cypher, however, distillation remains underexplored. To our knowledge, Distil-CypherGPT (Chong et al., 2025) is the only recent work that directly studies distillation for Cypher-based KGQA, and it mainly follows an output-level distillation paradigm. This suggests that Cypher distillation is a promising direction that has not been studied deeply. More importantly, existing distillation objectives are not designed to capture the relations that make a Cypher query correct. A valid query must connect clauses, graph patterns, properties, filters, and return expressions to the relevant parts of the question and schema. Token-level matching can transfer likely next-token preferences, but it does not explicitly teach the student how each generated query fragment is grounded in the question-schema context. As a result, compact students may produce fluent Cypher that uses incorrect labels, relationships, properties, or conditions.

We propose **CypherKD**, a distillation method for compact Text-to-Cypher models. CypherKD transfers two kinds of teacher behavior: next-token prediction and question-schema grounding. It keeps the student anchored to gold Cypher queries with an SFT loss, and applies logit-level KD so that the student matches the teacher's next-token distributions. CypherKD further adds an on-policy feature-level signal. During training, we distill not only on gold queries but also on queries generated by the student, so the teacher provides supervision on outputs that the student is likely to produce at inference time. From each distillation query, CypherKD extracts Cypher-aware spans, such as clauses, graph patterns, filters, and expressions. It then aligns teacher and student span-context relation scores between these spans and the question-schema context. This encourages the compact model to learn which parts of the question and schema support each generated query fragment, rather than only learning likely Cypher tokens.

Our contributions are:

- We introduce CypherKD, an on-policy feature-level distillation framework for compact Text-to-Cypher models, combining logit-level KD with teacher-derived span-to-question/schema associations.
- We define a Cypher-aware span-to-context objective that aligns teacher and student relations between Cypher spans and the question-schema context, providing a task-specific feature-level signal beyond token-level KD.
- Extensive experiments on CypherBench, Mind-the-Query, and Neo4j-Text2Cypher show that CypherKD achieves state-of-the-art performance among distillation methods on executability, execution accuracy, and provenance subgraph similarity.

2 Background & Related Work

2.1 Text-to-Cypher Generation

Graph databases such as Neo4j are increasingly used to store complex, interconnected data (Tiwari et al., 2025). Accessing this data often requires Cypher, a query language that demands specialized knowledge. Text-to-Cypher addresses this access problem by mapping a natural language question and graph schema into an executable Cypher query (Ozsoy et al., 2025). A valid Cypher query must be syntactically well formed, schema-consistent, and semantically aligned with the question.

Recent work improves Text-to-Cypher data and training through synthetic question-query generation (Zhong et al., 2025), LLM-supervised generation and verification (Tiwari et al., 2025), and consolidated fine-tuning and evaluation resources (Ozsoy et al., 2025). Other work focuses on schema use: semantic schema prompting improves schema presentation (Wan et al., 2025), and Cypher-RI integrates schema selection into generation through reinforcement learning (Su et al., 2025). Related text-to-SQL work also suggests that question-schema relation modeling is important for structured query generation (Cao et al., 2021).

Benchmarks still show that Text-to-Cypher remains difficult. CypherBench highlights the challenge of precise retrieval over large property-graph schemas (Feng et al., 2025). Mind-the-Query shows that models still produce schema-invalid queries, wrong graph traversals, invalid Cypher operations, and aggregation hallucinations (Chauhan et al., 2025). These errors often reflect weak

grounding between generated query fragments and the question-schema context.

2.2 Knowledge Distillation for Language Models

Knowledge distillation transfers behavior from a large teacher to a compact student by matching the teacher's output distribution (Hinton et al., 2015). Off-policy LLM distillation matches teacher and student distributions under reference prefixes; examples include generalized f-divergence objectives for sequence-level KD (Wen et al., 2023) and Concrete Score Distillation over continuous token relaxations (Kim et al., 2025). On-policy methods reduce training-inference mismatch by distilling on states visited by the student. DistiLLM uses skewed KL losses with student-generated outputs and adaptive replay (Ko et al., 2024), while AMiD stabilizes matching with an α-mixture assistant distribution (Shin et al., 2026).

Feature-based distillation transfers internal information instead of only output distributions. Patient KD aligns intermediate hidden states (Sun et al., 2019), TinyBERT aligns task-adaptive representations (Jiao et al., 2020), and MiniLM-style methods transfer attention relations (Wang et al., 2020, 2021). Relational KD matches pairwise distances (Park et al., 2019), while FDD aligns feature dynamics (Gong et al., 2025). These objectives provide useful compression signals, but remain general-purpose: they do not tell the student which parts of a generated Cypher query should be supported by the question and which should be supported by the schema.

CypherKD builds on these lines by combining logit-level distillation with a task-specific span-context relation objective for Text-to-Cypher.

3 Span-to-Context Distillation

In this section, we present **CypherKD**, a span-to-context distillation framework for compact Text-to-Cypher models. The core idea of CypherKD is to transfer not only token-generation behavior, but also teacher-derived associations between Cypher query structures and the question-schema context. To achieve this, CypherKD combines logit-level distillation with a span-relation objective that aligns teacher and student relation scores between Cypher-aware spans and question-schema evidence. Figure 1 provides an overview of this span-context relation distillation process.

We detail the method through its main components. We first introduce the Cypher-aware span inventory and the question-schema context representations used to model span-to-context associations (Section 3.1). We then define the span-context relation loss that aligns teacher and student relation scores (Section 3.2). Finally, we describe the mixed KD data construction adopted from DistiLLM and then define the joint training objective that combines SFT, logit-based KD, and the proposed feature-level span-relation loss (Section 3.3).

3.1 Cypher-Aware Span and Context Representations

Cypher-Aware Span Construction. For each Cypher sequence used for distillation, we construct a span inventory using lightweight Cypher-aware rules, illustrated in Figure 2. Within the query string, we identify four span categories: (1) *clause keywords*, such as `MATCH`, `WHERE`, `RETURN`, `ORDER BY`, and `LIMIT`; (2) *node patterns*, such as `(p:Person)` or `(m{title: t})`; (3) *triplets*, covering local path structures that combine a source node, relationship pattern, and target node, such as `(p)-[:WORKS_AT]->(c)`; and (4) *expressions*, including `WHERE` conditions and comma-separated expressions in `RETURN`, `WITH`, and `ORDER BY`. These spans often contain the node labels, relationship types, directions, property names, literal values, filters, and aggregations that define the query semantics.

Suppose one training instance contains N^{sp} extracted Cypher spans, indexed by $k \in \{1, \ldots, N^{\text{sp}}\}$. For each span k, tokenizer offset mappings are used to map character offsets to the corresponding token-index set $\mathcal{S}_k$.

Span Representation. For a model $m \in \{S, T\}$, let $H_t^{(m)}$ denote the last hidden state of token t. For each span k, we compute its representation $U_k^{(m)}$ as a weighted average of its constituent token hidden states:

$$U_k^{(m)} = \frac{\sum_{t \in \mathcal{S}_k} w_t^{(m)} H_t^{(m)}}{\sum_{t \in \mathcal{S}_k} w_t^{(m)}}, \qquad (1)$$

where $\mathcal{S}_k$ is the set of token indices in span k, and $w_t^{(m)}$ is the importance weight of token t. Let $\mathcal{O}$ be the set of non-padding target-token positions. We derive $w_t^{(m)}$ from a pairwise relevance matrix by measuring the incoming relevance mass of each

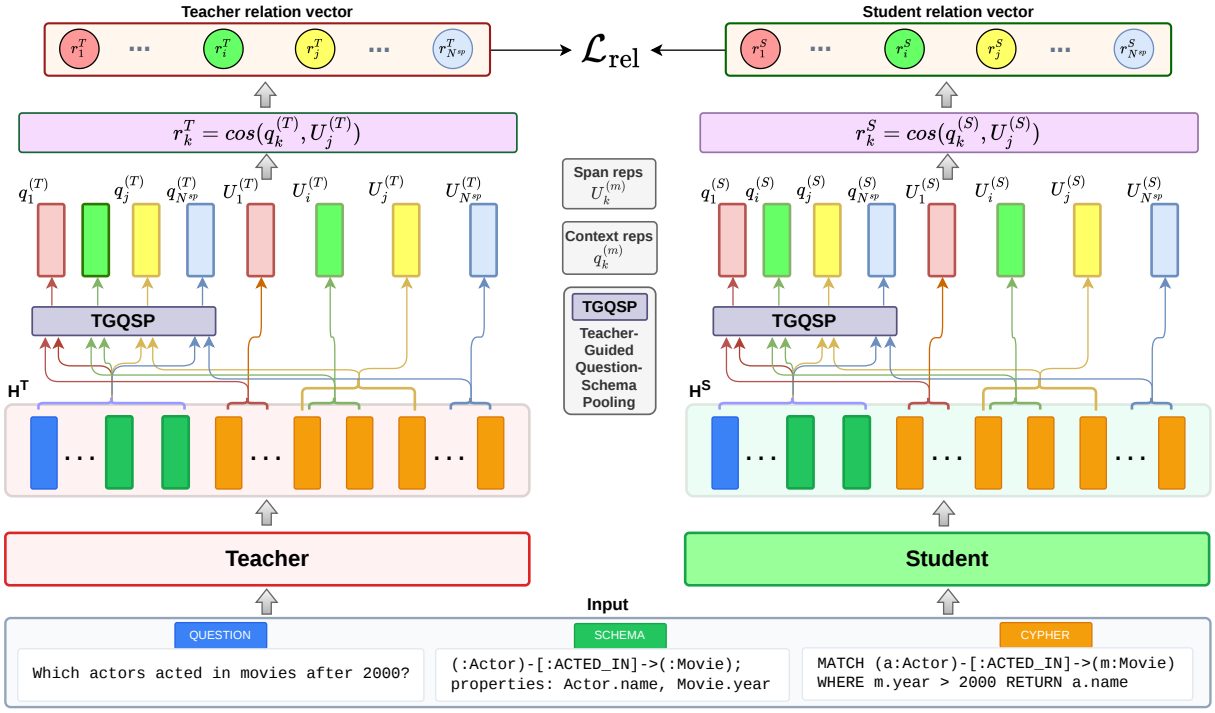

Figure 1: Overview of the CypherKD span-context relation loss. Given a question, schema, and Cypher query, CypherKD extracts Cypher-aware spans and computes teacher and student span-context relation scores. The relation loss aligns these score vectors rather than directly matching hidden states.

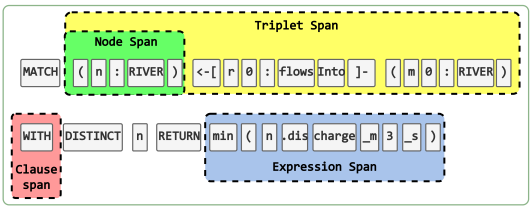

Figure 2: Cypher-aware span construction process.

target token:

$$w_t^{(m)} = \frac{\widetilde{w}_t^{(m)}}{\sum_{t' \in \mathcal{O}} \widetilde{w}_{t'}^{(m)}}, \quad \widetilde{w}_t^{(m)} = \sum_{\substack{s \in \mathcal{O} \\ s \neq t}} \alpha_{s \to t}^{(m)}. \quad (2)$$

Here $\alpha_{s \to t}^{(m)}$ is the pairwise relevance score from token s to t, computed via scaled dot products of feature-normalized hidden states. Padding and self-pairs are masked (details in Appendix E). Teacher and student span representations are computed separately from their last hidden states.

Question-Schema Context. The context mask keeps prompt tokens from the `QUESTION` and `SCHEMA` regions and excludes unrelated instruction tokens. Let $\mathcal{C}$ denote the selected question-schema token positions. In our prompt template, the user message contains explicit `QUESTION:` and `SCHEMA:` blocks followed by the generation instruction; $\mathcal{C}$ contains tokens inside those two blocks and excludes the system prompt, formatting instructions, and padding tokens.

Teacher-Guided Question-Schema Context Pooling. For each Cypher span, we derive a context vector over the selected question-schema tokens. The attention is teacher-guided: the teacher selects source evidence, and the student is evaluated against the same evidence. The teacher source score is

$$e_{k,t}^{(T)} = \frac{(U_k^{(T)})^\top H_t^{(T)}}{\sqrt{d_T}}, \quad t \in \mathcal{C} \quad (3)$$

where d_T denote the hidden dimension of the teacher model. The attention weight over question-schema tokens is then

$$\alpha_{k,t}^{(T)} = \text{softmax}_{t \in \mathcal{C}}\left(e_{k,t}^{(T)}\right). \quad (4)$$

We then obtain the teacher and student context vectors with the same attention weights:

$$\begin{aligned} \mathbf{q}_k^{(T)} &= \sum_{t \in \mathcal{C}} \alpha_{k,t}^{(T)} H_t^{(T)}, \\ \mathbf{q}_k^{(S)} &= \sum_{t \in \mathcal{C}} \alpha_{k,t}^{(T)} H_t^{(S)}. \end{aligned} \quad (5)$$

By reusing the teacher attention weights for both context vectors, the loss compares teacher and student span-context relations against the same teacher-selected evidence. This isolates relation matching from the separate problem of learning a student-specific evidence selector.

3.2 Span-Context Relation Loss

At the feature level, CypherKD defines the span-relation loss $\mathcal{L}_{\text{rel}}$. The span representation $U_k^{(m)}$ and the teacher-guided context vector $\mathbf{q}_k^{(m)}$ are used to compute a scalar grounding relation for each span. The loss matches this relation between teacher and student, avoiding direct hidden-state matching while preserving the teacher's span-context geometry.

We assign each Cypher span a teacher-derived importance score by aggregating teacher token weights over its tokens and normalizing across spans:

$$\omega_k^{\text{sp}} = \frac{\widetilde{\omega}_k^{\text{sp}}}{\sum_{j=1}^{N^{\text{sp}}} \widetilde{\omega}_j^{\text{sp}}}, \quad \text{with} \quad \widetilde{\omega}_k^{\text{sp}} = \sum_{t \in \mathcal{S}_k} w_t^{(T)} \quad (6)$$

where N^{sp} is the number of spans. The relation score for span k is the cosine similarity between its weighted span representation and its teacher-guided question-schema context vector:

$$r_k^{(m)} = \cos(U_k^{(m)}, \mathbf{q}_k^{(m)}), \quad m \in \{S, T\}. \quad (7)$$

We use cosine similarity because the relation objective is intended to match span-context geometry while being less sensitive to teacher-student differences in hidden-state scale and dimensionality than direct vector matching. The span-relation loss is

$$\mathcal{L}_{\text{rel}} = \sum_{k=1}^{N^{\text{sp}}} \omega_k^{\text{sp}} \cdot \left(r_k^{(S)} - r_k^{(T)} \right)^2. \quad (8)$$

This loss is a teacher-weighted mean squared error between the student and teacher span-context relation scores. The teacher-derived span weights ω_k^{sp} bias the loss toward spans that receive higher teacher relevance mass, allowing the student to prioritize more salient Cypher structures during grounding-aware distillation. Compared with uniform span weighting, this strategy encourages compact students to focus more on important span-context relations while reducing the influence of less informative spans.

3.3 Mixed KD Data and Joint Training Objective

Mixed KD data. For each training example $(\mathbf{x}, \mathbf{s}, \mathbf{y})$, the SFT loss uses the gold query $\mathbf{y}_{\text{GT}}$. For KD, we use both the gold SFT target $\mathbf{y}_{\text{GT}}$ and a student-generated query $\mathbf{y}_{\text{s}} \sim p_S(\cdot \mid P(\mathbf{x}, \mathbf{s}))$:

$$\mathcal{D}_{\text{KD}} = \{(\mathbf{x}, \mathbf{s}, \mathbf{y}) : \mathbf{y} \in \{\mathbf{y}_{\text{GT}}, \mathbf{y}_{\text{s}}\}\}. \quad (9)$$

Following DistiLLM (Ko et al., 2024), the gold sequence anchors distillation to the supervised data, while the student-generated sequence exposes the teacher to prefixes that the student actually visits during decoding. In practice, these student-generated outputs may be reused through adaptive off-policy replay rather than regenerated at every update (Appendix A). This mixed KD data preserves the benefit of student-generated, on-policy states while reducing the cost of fully on-policy generation. We adopt this mixed-data construction for the logit-based KD term, and CypherKD further adds the Cypher-aware span-relation loss defined above.

Logit-based KD loss. The logit-based loss aligns teacher and student next-token distributions at each position. By default, we use the Adaptive SRKL loss from DistiLLM, while alternative token-level divergences are evaluated in §4.3.

Feature-based KD loss. At the feature level, CypherKD uses hidden states only to compute a relation between each Cypher span and the question-schema context. The feature-based KD loss is the span-relation loss, which matches these relation scores between the teacher and student.

Training objective. Student training combines supervised fine-tuning with logit-based and feature-based KD:

$$\mathcal{L}_{\text{total}} = (1-\beta)\mathcal{L}_{\text{CE}} + \beta \left(\mathcal{L}_{\text{logit}} + \lambda_{\text{rel}} \mathcal{L}_{\text{rel}} \right), \quad (10)$$

where $\mathcal{L}_{\text{CE}}$ is computed only on the gold SFT query $\mathbf{y}$, while both KD terms are computed on the mixed distillation data $\mathcal{D}_{\text{KD}}$ from Equation 9. $\mathcal{L}_{\text{logit}}$ is the logit-based KD loss, and $\mathcal{L}_{\text{rel}}$ is the feature-based span-relation loss defined in Equation 8. The coefficient $\beta \in [0, 1]$ balances supervised learning and teacher-guided learning, and λ_{rel} scales the feature-based relation loss. In the default setting, $\mathcal{L}_{\text{logit}}$ uses Adaptive SRKL. We ablate the logit-based KD divergence using Reverse KL (RKL), Skew Forward KL (SFKL), Concrete Score Distillation (CSD), and Adaptive SRKL in Table 5.

4 Experiments

4.1 Experimental Setup

Training and Evaluation Data. All teacher and student models are trained on the **CypherBench** (Feng et al., 2025) training split, which covers diverse graph matching patterns (e.g., multi-hop reasoning, aggregation). To evaluate in-domain and out-of-domain generalization, we test on three complementary Text-to-Cypher datasets: the CypherBench test split, **Mind-the-Query** (Chauhan et al., 2025) for diverse real-world domains, and **Neo4j-Text2Cypher** (Ozsoy et al., 2025), a standardized consolidation of public resources.

Implementation and Metrics. We evaluate two primary same-family distillation settings: Qwen3-4B-Instruct-2507 (Yang et al., 2025) to Qwen3-0.6B, and Llama-3.2-8B (Grattafiori et al., 2024) to Llama-3.2-1B. This setting examines whether the method remains effective when the teacher and student differ in model version and layer structure. All models are trained via LoRA, with the teacher initially adapted to the Text-to-Cypher data (details in Appendix C). Following CypherBench (Feng et al., 2025), we report three metrics. We treat **Execution Accuracy** (EX) as the primary metric, as practical Text-to-Cypher systems must ultimately return correct answers. Additionally, we report **Executability** to verify syntactic validity, and **PSJS** (provenance subgraph Jaccard similarity) to evaluate structural graph grounding.

Baselines. We compare against representative fine-tuning and distillation baselines: (1) **SFT**, the student fine-tuned on the same training data without distillation; (2) **FKL** (Hinton et al., 2015), (3) **RKL** (Wen et al., 2023), (4) **SFKL/SRKL** (Ko et al., 2024), (5) **DistiLLM** (Ko et al., 2024), (6) **FDD** (Gong et al., 2025), (7) **CSD** (Kim et al., 2025), (8) **AMiD** (Shin et al., 2026). We also report teacher performance as a reference point.

4.2 Text-to-Cypher Results

Overall Performance. Table 1 presents the main Text-to-Cypher results. CypherKD consistently improves answer correctness over the strongest logit-level baselines. On CypherBench, EX increases from 62.18% to 64.61% (+2.43) for Qwen3-0.6B and from 60.09% to 62.11% (+2.02) for Llama-3.2-1B. On Mind-the-Query, EX improves from 27.87% to 29.93% (+2.06) and from 25.66% to 27.09% (+1.43), respectively. These gains are important because EX directly reflects whether the generated Cypher query returns the correct answer on the target graph database. CypherKD also improves structural grounding. For Qwen3-0.6B, it achieves the best PSJS on all three benchmarks, including gains over the strongest logit-level baseline from 59.55% to 61.42% on CypherBench, from 34.39% to 35.95% on Mind-the-Query, and from 29.54% to 30.26% on Neo4j-Text2Cypher. For Llama-3.2-1B, CypherKD obtains the best EX across all benchmarks and the best PSJS on CypherBench and Mind-the-Query, while remaining competitive on Neo4j-Text2Cypher. Overall, the consistent EX gains together with stronger PSJS suggest that span-context grounding improves semantic query correctness rather than merely increasing syntactic validity.

4.3 Ablation Study

We conduct ablation experiments to analyze the contribution of each component in CypherKD. Unless otherwise noted, ablations use the Qwen3-4B/Qwen3-0.6B teacher-student pair and the training setup reported in Appendix C. The default model uses Adaptive SRKL as the token-level objective, sets the distillation ratio to $\beta = 0.6$, and sets the span-relation weight to $\lambda_{\text{rel}} = 0.6$. Before the component ablations, we report this diagnostic setting. We then study four questions: how strongly to weight $\mathcal{L}_{\text{rel}}$, whether triplet spans are necessary, whether Cypher-aware span relations are better than generic token-token relations, and whether the relation loss remains useful across different token-level KD objectives.

Distillation across model configurations. Table 2 shows that SFT alone performs poorly in the Qwen3-to-Qwen2.5 setting, while distillation gives large gains. CypherKD achieves the best EX on both benchmarks, improving over the strongest baseline by 1.91% on CypherBench and 2.51% on Mind-the-Query. These results suggest that span-context relation distillation remains useful across different model configurations.

Effect of λ_{rel}. Figure 3 shows the effect of varying the span-relation loss weight λ_{rel} using EX and PSJS; executability is provided in Appendix D. When $\lambda_{\text{rel}} = 0$, the model reduces to token-level distillation only. Adding $\mathcal{L}_{\text{rel}}$ improves both answer correctness and structural grounding, with $\lambda_{\text{rel}} = 0.6$ performing best on both bench-

Table 1: Main Text-to-Cypher results. EX, PSJS, and Executable denote execution accuracy, provenance subgraph Jaccard similarity, and executability.

Method	CypherBench			Mind-the-Query			Neo4j-Text2Cypher		
	EX (%)	PSJS (%)	Executable (%)	EX (%)	PSJS (%)	Executable (%)	EX (%)	PSJS (%)	Executable (%)
Qwen3-4B (Teacher)	70.32	68.52	98.08	53.05	65.63	87.00	17.89	44.40	77.85
Qwen3-0.6B (SFT)	50.68	46.33	99.11	5.71	7.80	93.37	6.23	12.36	89.72
FKL	58.28	54.93	98.25	27.87	34.39	88.51	7.98	15.49	82.71
RKL	61.24	58.48	98.47	26.46	33.28	87.53	9.51	29.54	81.70
SFKL	60.26	56.77	97.61	27.45	33.87	84.57	8.22	28.32	77.57
DistiLLM	62.18	58.03	98.51	27.36	33.24	88.32	8.99	26.35	76.23
FDD	56.43	53.11	98.68	12.17	13.02	88.17	7.29	20.64	**87.98**
CSD	61.58	59.55	**99.06**	25.80	31.72	88.85	9.15	27.61	82.06
AMiD	60.39	57.81	98.47	26.12	32.34	85.86	8.79	26.84	77.09
CypherKD	**64.61**	**61.42**	98.98	**29.93**	**35.95**	**89.39**	**9.84**	**30.26**	85.63
Llama-3.2-8B (Teacher)	70.44	68.78	98.30	35.44	40.94	81.75	16.96	37.69	79.31
Llama-3.2-1B (SFT)	17.93	9.09	90.55	2.20	2.76	67.06	3.40	2.66	71.34
FKL	57.88	55.23	96.85	25.66	28.79	88.58	9.68	24.49	83.68
RKL	58.09	55.33	97.02	25.33	28.77	88.82	9.23	24.23	83.56
SFKL	58.22	55.29	97.87	24.62	28.67	91.16	8.87	22.97	90.28
DistiLLM	58.77	56.96	98.30	22.62	26.48	91.44	9.03	25.74	87.85
FDD	54.13	51.69	95.78	9.56	10.51	84.91	7.69	19.97	83.08
CSD	60.09	57.98	98.17	20.61	24.42	91.09	9.55	**26.07**	**91.21**
AMiD	57.20	54.69	98.25	24.00	27.59	89.87	9.51	22.60	90.04
CypherKD	**62.11**	**58.81**	**98.54**	**27.09**	**31.13**	**92.28**	**9.88**	26.03	91.17

Table 2: Qwen3-to-Qwen2.5 diagnostic results.

Method	CypherBench		Mind-the-Query	
	EX (%)	PSJS (%)	EX (%)	PSJS (%)
Qwen3-4B (Teacher)	70.32	68.52	53.05	65.63
Qwen2.5-0.5B (SFT)	6.86	2.49	1.63	1.39
DistiLLM	56.52	51.49	19.28	20.97
CSD	57.16	53.11	18.23	18.6
AMiD	54.64	49.11	19.58	**22.65**
CypherKD	**59.07**	**53.69**	**22.09**	22.39

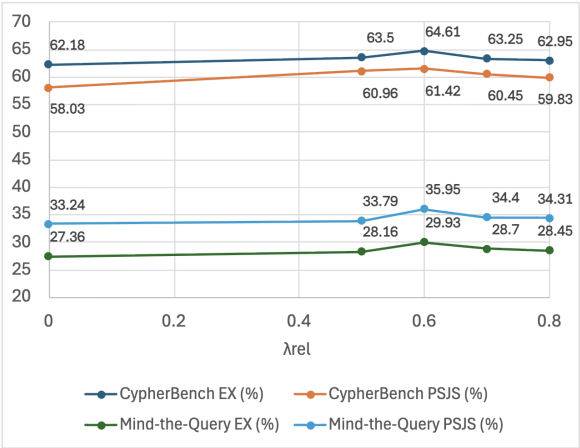

Figure 3: Effect of λ_{rel} on CypherBench and Mind-the-Query. Full metrics are reported in Appendix D.

marks: CypherBench EX increases from 62.18% to 64.61% (+2.43) and PSJS from 58.03% to 61.42% (+3.39), while Mind-the-Query EX increases from 27.36% to 29.93% (+2.57) and PSJS from 33.24% to 35.95% (+2.71). Beyond this point, both metrics drop on both benchmarks, suggesting that an overly strong span-relation signal can interfere with token-level learning. We therefore use $\lambda_{\text{rel}} = 0.6$ as the default weight.

Effect of Triplet Spans. Table 3 examines whether complete source-relationship-target spans are necessary in the span inventory. Removing triplet spans leaves clauses, node patterns, and expressions, but removes the most direct representation of local graph traversal structure; relationship tokens may still appear inside expressions or node-adjacent contexts, but they are no longer grouped with their source and target nodes. Adding triplet spans improves both CSD and Adaptive SRKL on both benchmarks. The effect is largest with Adaptive SRKL, where CypherBench EX rises from 60.18% to 64.61% (+4.43) and Mind-the-Query EX rises from 26.92% to 29.93% (+3.01). This suggests that path-level spans provide information beyond isolated node and clause spans, especially for transferring relationship type and traversal structure.

Table 3: Effect of triplet spans in the Cypher-aware span inventory.

Method	Span Inventory	CypherBench		Mind-the-Query	
		EX (%)	PSJS (%)	EX (%)	PSJS (%)
CSD	w/o triplet spans	61.58	59.28	27.09	31.60
	w/ triplet spans	63.07	60.12	29.38	35.20
DistiLLM	w/o triplet spans	60.18	56.44	26.92	32.10
	w/ triplet spans	**64.61**	**61.42**	**29.93**	**35.95**

Token-Relation versus Span-Relation Loss. Table 4 compares a generic token relational loss with

the proposed span-relation loss. The generic approach aligns dense token-token similarity matrices between the teacher and student. Unlike $\mathcal{L}_{\text{rel}}$, it is not Cypher-aware and does not explicitly bind query spans to question-schema evidence. The proposed span-relation loss consistently outperforms the generic token-relation loss for both CSD and DistiLLM. For CSD, replacing $\mathcal{L}_{\text{tok-rel}}$ with $\mathcal{L}_{\text{rel}}$ improves EX by 2.42 points on CypherBench and 2.48 points on Mind-the-Query; for DistiLLM, the gains are 3.71 and 2.52 points, respectively. PSJS also improves consistently, showing that span-level matching enhances both answer correctness and structural grounding. Under a large teacher-student capacity gap, dense token-token relation matching may impose a heavy, untargeted constraint on the compact student (Zhang et al., 2023, 2025). Therefore, the relation target should align with the structure of Text-to-Cypher generation.

Table 4: Generic token-relation loss vs. the proposed span-relation loss on CypherBench and Mind-the-Query.

Method	CypherBench		Mind-the-Query	
	EX (%)	PSJS (%)	EX (%)	PSJS (%)
CSD *w/* $\mathcal{L}_{\text{tok-rel}}$	60.65	57.72	26.9	33.25
CSD *w/* $\mathcal{L}_{\text{rel}}$	63.07	60.12	29.38	35.20
DistiLLM *w/* $\mathcal{L}_{\text{tok-rel}}$	60.90	57.15	27.41	33.53
DistiLLM *w/* $\mathcal{L}_{\text{rel}}$	**64.61**	**61.42**	**29.93**	**35.95**

Ablation on Distillation Objectives. Table 5 compares four token-level KD objectives (Reverse KL, SFKL, CSD, and Adaptive SRKL) on CypherBench and Mind-the-Query, both with and without our span-relation loss ($\mathcal{L}_{\text{rel}}$). We report EX and PSJS in the main table to emphasize answer correctness and structural grounding, with executability included in Appendix D. Adding $\mathcal{L}_{\text{rel}}$ improves EX for every token-level objective on both datasets, and generally improves PSJS as well. The EX gains range from 1.49 to 2.81 points on CypherBench and from 1.85 to 3.58 points on Mind-the-Query, indicating that the span-level structural signal is complementary to different token-level KD losses. We adopt Adaptive SRKL + $\mathcal{L}_{\text{rel}}$ as our default full objective, as it achieves the best overall EX on both benchmarks.

Computational Overhead. Table 6 shows the training cost of adding $\mathcal{L}_{\text{rel}}$ under the same two-GPU setup. The overhead is modest: step time increases by 2.7-8.0% across objectives (3.13 to 3.31s for Adaptive SRKL), and average allocated memory increases by about 0.92GB. The overhead is training-only and does not affect deployed student latency. Overall, $\mathcal{L}_{\text{rel}}$ is computationally efficient and easy to integrate with token-level distillation.

Table 5: Token-level KD losses with and without $\mathcal{L}_{\text{rel}}$ on CypherBench and Mind-the-Query. Full metrics are reported in Appendix D.

Method	CypherBench		Mind-the-Query	
	EX (%)	PSJS (%)	EX (%)	PSJS (%)
RKL	61.24	58.48	26.46	33.28
w/ $\mathcal{L}_{\text{rel}}$	64.05	**61.67**	29.13	34.66
SFKL	60.26	56.77	27.45	33.87
w/ $\mathcal{L}_{\text{rel}}$	62.18	58.40	29.30	34.93
CSD	61.58	59.55	25.80	31.72
w/ $\mathcal{L}_{\text{rel}}$	63.07	60.12	29.38	35.20
DistiLLM	62.18	58.03	27.36	33.24
w/ $\mathcal{L}_{\text{rel}}$	**64.61**	61.42	**29.93**	**35.95**

Table 6: Training overhead of logits distillation methods with and without the proposed span-relation loss.

Method	Time/step (s)	Avg. alloc. (GB)	Peak alloc. (GB)
RKL	2.99	10.24	22.65
w/ $\mathcal{L}_{\text{rel}}$	3.11	11.16	23.56
SFKL	2.93	10.24	22.40
w/ $\mathcal{L}_{\text{rel}}$	3.01	11.16	22.81
CSD	2.74	10.24	20.13
w/ $\mathcal{L}_{\text{rel}}$	2.96	11.16	20.54
DistiLLM	3.13	10.24	24.67
w/ $\mathcal{L}_{\text{rel}}$	3.31	11.16	25.58

5 Conclusion

We presented CypherKD, a span-relation distillation framework for compact Text-to-Cypher generation. CypherKD combines logit-based distillation with a feature-based span-relation loss that aligns teacher-student span-context relation scores. It extracts Cypher-aware spans from the distillation query, conditions each span on selected question and schema tokens, and matches the resulting relation scores using the final hidden states. This design provides a training-time relation signal for the structural nature of Cypher generation, where executable correctness often depends on question intent and schema elements rather than surface token overlap alone. Because span-relation alignment is used only during training, CypherKD transfers teacher-derived span-context associations while preserving the inference cost of the compact student.

6 Limitations

This work is an empirical study of span-context distillation for Text-to-Cypher generation, and its conclusions are bounded by the datasets, model families, and training settings evaluated here. Broader experiments with additional graph domains, model scales, and deployment constraints would further clarify when the method is most effective.

CypherKD also follows the standard teacher-student distillation setting, which requires running the teacher during training. This requirement affects training cost, although the distilled student is used normally at inference time.

Finally, the method targets query generation quality rather than end-to-end production governance. Practical systems should still rely on the usual validation, execution, and access-control mechanisms around generated database queries.

References

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, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: line graph enhanced text-to-sql model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2541–2555. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8128–8140. Association for Computational Linguistics.

Vashu Chauhan, Shobhit Raj, Shashank Mujumdar, Avirup Saha, and Anannay Jain. 2025. Mind the query: A benchmark dataset towards text2cypher task. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025 - Industry Track, Suzhou, China, November 4-9, 2025*, pages 1890–1905. Association for Computational Linguistics.

You Li Chong, Chin-Poo Lee, and Ming Kim Lim. 2025. Distilcyphergpt: enhancing large language models for knowledge graph question answering in cypher through knowledge distillation. *Data Min. Knowl. Discov.*, 39(6):74.

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. *CoRR*, 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), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 8934–8958. Association for Computational Linguistics.

Guoqiang Gong, Jiaxing Wang, Jin Xu, Deping Xiang, Zicheng Zhang, Leqi Shen, Yifeng Zhang, JunhuaShu JunhuaShu, ZhaolongXing ZhaolongXing, Zhen Chen, Pengzhang Liu, and Ke Zhang. 2025. Beyond logits: Aligning feature dynamics for effective knowledge distillation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 23067–23077. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

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, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, Findings of ACL, pages 4145–4157. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9237–9251. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023b. Unikgqa: Unified retrieval and reasoning for

solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, Findings of ACL, pages 4163–4174. Association for Computational Linguistics.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Findings of ACL, pages 9410–9421. Association for Computational Linguistics.

Yeongmin Kim, Donghyeok Shin, Mina Kang, Byeonghu Na, and Il-Chul Moon. 2025. Distillation of large language models via concrete score matching. *CoRR*, abs/2509.25837.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. Distillm: Towards streamlined distillation for large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 24872–24895. PMLR / OpenReview.net.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. 2024. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Findings of ACL, pages 2039–2056. Association for Computational Linguistics.

Makbule Gulcin Ozsoy, Leila Messallem, Jon Besga, and Gianandrea Minneci. 2025. Text2cypher: Bridging natural language and graph databases. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025 - Workshops, Abu Dhabi, UAE, January 19-24, 2025*, pages 100–108. Association for Computational Linguistics.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational knowledge distillation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3967–3976. Computer Vision Foundation / IEEE.

Diego Sanmartin. 2024. KG-RAG: bridging the gap between knowledge and creativity. *CoRR*, abs/2405.12035.

Donghyeok Shin, Yeongmin Kim, Suhyeon Jo, Byeonghu Na, and Il chul Moon. 2026. AMid: Knowledge distillation for LLMs with α-mixture assistant distribution. In *The Fourteenth International Conference on Learning Representations*.

Hanchen Su, Xuyuan Li, Yan Zhou, zhuoyi lu, Ziwei Chai, Haozheng Wang, Chen Zhang, and YANG YANG. 2025. Cypher-ri: Reinforcement learning for integrating schema selection into cypher generation. In *Advances in Neural Information Processing Systems*, volume 38, pages 100204–100225. Curran Associates, Inc.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4322–4331. Association for Computational Linguistics.

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, NAACL 2025 - Volume 2: Short Papers, Albuquerque, New Mexico, April 29 - May 4, 2025*, pages 623–640. Association for Computational Linguistics.

Yuwei Wan, Zheyuan Chen, Ying Liu, Chong Chen, and Michael S. Packianather. 2025. Prompting large language models based on semantic schema for text-to-cypher transformation towards domain q&a. *Decis. Support Syst.*, 199:114553.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. Minilmv2: Multi-head self-attention relation distillation for compressing pre-trained transformers. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, Findings of ACL, pages 2140–2151. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. 2023. f-divergence minimization for sequence-level knowledge distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10817–10834. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Chen Zhang, Qiuchi Li, Dawei Song, Zheyu Ye, Yan Gao, and Yao Hu. 2025. Towards the law of capacity gap in distilling language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 22504–22528. Association for Computational Linguistics.

Chen Zhang, Yang Yang, Jiahao Liu, Jingang Wang, Yunsen Xian, Benyou Wang, and Dawei Song. 2023. Lifting the curse of capacity gap in distilling language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4535–4553. Association for Computational Linguistics.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5773–5784. Association for Computational Linguistics.

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, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 672–692. Association for Computational Linguistics.

A DistiLLM

To address the optimization instability of standard KD and the cost of fully on-policy student generation, Ko et al. (2024) proposed DistiLLM. This framework introduces two primary innovations:

Skew (Reverse) Kullback-Leibler Divergence: Standard KLD often leads to optimization instability due to gradient explosion when the student model assigns low probability to the teacher's support. To mitigate this, DistiLLM utilizes **Skew KLD**, which interpolates the target distribution with the student's distribution. Formally, the α-Skew KLD is defined as:

$$D_{SKL}^{(\alpha)}(p, q_\theta) = D_{KL}(p \parallel \alpha p + (1-\alpha) q_\theta) \quad (11)$$

Similarly, to leverage the mode-seeking behavior of Reverse KLD while maintaining gradient stability, they proposed **Skew Reverse KLD (SRKL)**:

$$D_{SRKL}^{(\alpha)}(p, q_\theta) = D_{KL}(q_\theta \parallel (1-\alpha) p + \alpha q_\theta) \quad (12)$$

where p is the teacher, q_θ is the student, and $\alpha \in [0, 1]$ controls the mixing ratio. These modifications prevent the denominator in the gradient term from vanishing, thereby bounding the gradient norm and ensuring smoother convergence.

Adaptive Off-Policy Generation: While utilizing Student-Generated Outputs (SGOs) addresses training-inference mismatch, generating fresh sequences at every iteration is computationally expensive. DistiLLM proposes an efficient off-policy strategy using a replay buffer $\mathcal{D}_R$ to store and reuse SGOs. Furthermore, it employs an *adaptive SGO scheduler* that dynamically adjusts the probability ϕ of sampling SGOs based on the student's validation loss trends. Thus, DistiLLM still benefits from student-generated outputs, but it is not fully on-policy because stored SGOs can be reused after the student has been updated. This mechanism ensures SGOs are introduced only when necessary to correct distribution shifts, balancing training efficiency with the mitigation of noisy feedback.

B Dataset Description

We conduct our experiments on three representative and diverse Text-to-Cypher benchmarks: CypherBench (Feng et al., 2025), Mind-the-Query (Chauhan et al., 2025), and Neo4j-Text2Cypher (Ozsoy et al., 2025). These datasets are selected to evaluate both in-domain performance and cross-domain generalization across different graph do-

mains and query patterns. For training, we construct our training data by using only the training split of CypherBench, ensuring that the student is exposed to a broad set of graph structures and query intents during learning.

Table 7 gives a detailed summary of the databases and sample counts used in our experiments. The CypherBench training set contains four domains: biology, soccer, art, and terrorist attack, comprising a total of 8,295 samples. The CypherBench evaluation set contains seven held-out domains, including geography, flight accident, politics, company, fictional character, movie, and NBA. This split allows us to evaluate zero-shot generalization to unseen domains from the same benchmark family.

Mind-the-Query provides an additional evaluation setting with real-world graph domains, including bloom50, healthcare, and wwc. Neo4j-Text2Cypher further covers heterogeneous graph schemas across 15 domains, including social-network, company, movie, recommendation, game, and finance-related databases. No examples from Mind-the-Query or Neo4j-Text2Cypher are used for training; these datasets are used only for evaluation to measure the robustness of distilled models on unseen domains.

Table 7: Dataset composition used in training and evaluation.

Dataset	Database	Samples
Training	biology	2289
	soccer	1820
	art	2750
	terrorist attack	1436
CypherBench	geography	366
	flight accident	189
	politics	390
	company	347
	fictional character	385
	movie	401
	nba	270
Mind-the-Query	bloom50	1547
	healthcare	2566
	wwc	2946
Neo4j-Text2Cypher	bluesky	39
	buzzoverflow	120
	companies	275
	fincen	146
	gameofthrones	148
	grandstack	182
	movies	203
	neoflix	261
	network	109
	northwind	197
	offshoreleaks	115
	recommendations	253
	stackoverflow2	91
	twitch	185
	twitter	146

C Implementation Details

Prompt and mask construction. The generator prompt uses a system instruction followed by a user message with explicit `QUESTION` and `SCHEMA` fields. The system prompt is:

```
You are a Cypher Query Generator.

Your task is to generate a valid Cypher query that answers the user's natural language question using only the provided graph schema.

Rules:

- Use only labels, relationships, and properties that exist in the schema.
- Do not invent any schema elements.
- Return only the fields needed to answer the question.
- If duplicates are possible, use DISTINCT when appropriate.
- If the question requires aggregation, sorting, or limiting, use the correct Cypher clauses.
- Ensure the query is syntactically valid.

Output format: {"cypher": "The complete Cypher query"}

Return only the JSON object and nothing else.
```

The user prompt is:

```
QUESTION: {question}

SCHEMA: {schema}

Generate a Cypher query that answers the question using only the provided schema. Return only the JSON object in the required format.
```

The question-schema context mask $\mathcal{C}$ is built from tokenizer offsets by selecting tokens whose character spans fall inside the `QUESTION` or `SCHEMA` fields of the user prompt. Tokens from the system instruction, output-format instruction, and padding region are excluded from $\mathcal{C}$.

Cypher-aware spans are extracted from the Cypher sequence used for KD, either the gold target query or the student-generated output. The extraction rules identify clause keywords, node patterns, triplets, and expressions, then map their character offsets to target token positions using tokenizer offset mappings. These spans supervise only the training-time relation loss and are not used at inference time.

Table 8: Maximum generated tokens used during inference.

Benchmark	Qwen	Llama
CypherBench	1034	1053
Mind-the-Query	2427	2445
Neo4j-Text2Cypher	3092	2884

Training. All teacher and student models are adapted with LoRA (rank 32, $\alpha = 64$, dropout 0.1). Unless otherwise specified, training uses 5 epochs, AdamW, cosine learning-rate decay, weight decay 1×10^{-2}, gradient clipping at 1.0, per-device batch size 2, and 8 gradient accumulation steps. The maximum sequence and prompt lengths are 892 and 797 tokens for Qwen, and 899 and 810 tokens for Llama, respectively. We first fine-tune the teacher on the CypherBench training split with learning rate 1×10^{-5}, and then train the student with the same configuration but learning rate 1×10^{-4}. Training is run with `torchrun`, DeepSpeed ZeRO-1, and bf16 mixed precision.

Inference. At test time, only the student is used. LoRA adapters are merged before decoding, and each instance is formatted with the same prompt. We sample with temperature 0.5, top-p 0.95, and top-k 0. The maximum number of generated tokens is set by model family and benchmark, as shown in Table 8. We then parse the `cypher` field as the prediction.

Evaluation and Knowledge Graph usage. The schema is the only Knowledge Graph information given to the model at inference time. The underlying graph database is used only at evaluation time, when both the gold and predicted Cypher queries are executed on the corresponding Neo4j database. CypherBench and Mind-the-Query are evaluated on local Neo4j graph databases, while Neo4j-Text2Cypher is evaluated through the public Neo4j demo graph database endpoint. Following CypherBench (Feng et al., 2025), we report three evaluation metrics: **Executability**, which checks whether the predicted query can be run successfully; **Execution Accuracy** (EX), which checks whether it produces the same answer as the gold query; and **PSJS**, which measures the Jaccard similarity between the provenance subgraphs induced by the predicted and gold Cypher patterns.

D Full Ablation Metrics

Tables 9, 10, and 11 provide the full metrics, including executability, for the ablations in Figure 3, Tables 3, and 5.

Table 9: Full metrics for the effect of λ_{rel} on CypherBench and Mind-the-Query.

λ_{rel}	CypherBench			Mind-the-Query		
	EX (%)	PSJS (%)	Executable (%)	EX (%)	PSJS (%)	Executable (%)
0.0	62.18	58.03	98.51	27.36	33.24	88.32
0.5	63.50	60.96	98.81	28.16	33.79	84.77
0.6	**64.61**	**61.42**	**98.98**	**29.93**	**35.95**	**89.39**
0.7	63.25	60.45	98.76	28.70	34.40	85.37
0.8	62.95	59.83	98.89	28.45	34.31	85.15

Table 10: Full metrics for the effect of triplet spans in the Cypher-aware span inventory.

Method	Span Inventory	CypherBench			Mind-the-Query		
		EX (%)	PSJS (%)	Executable (%)	EX (%)	PSJS (%)	Executable (%)
CSD	w/o triplet spans	61.58	59.28	98.76	27.09	31.60	86.09
	w/ triplet spans	63.07	60.12	**99.19**	29.38	35.20	89.18
Adaptive SRKL	w/o triplet spans	60.18	56.44	98.30	26.92	32.10	83.61
	w/ triplet spans	**64.61**	**61.42**	98.98	**29.93**	**35.95**	**89.39**

Table 11: Full metrics for token-level KD losses with and without $\mathcal{L}_{\text{rel}}$ on CypherBench and Mind-the-Query.

Method	CypherBench			Mind-the-Query		
	EX (%)	PSJS (%)	Executable (%)	EX (%)	PSJS (%)	Executable (%)
RKL	61.24	58.48	98.47	26.46	33.28	87.53
w/ $\mathcal{L}_{\text{rel}}$	64.05	**61.67**	98.80	29.13	34.66	**92.56**
SFKL	60.26	56.77	97.61	27.45	33.87	84.57
w/ $\mathcal{L}_{\text{rel}}$	62.18	58.40	99.11	29.30	34.93	85.45
CSD	61.58	59.55	99.06	25.80	31.72	88.85
w/ $\mathcal{L}_{\text{rel}}$	63.07	60.12	**99.19**	29.38	35.20	89.18
Adaptive SRKL	62.18	58.03	98.51	27.36	33.24	88.32
w/ $\mathcal{L}_{\text{rel}}$	**64.61**	**61.42**	98.98	**29.93**	**35.95**	89.39

E Token Importance Computation

This section gives the exact pairwise relevance computation used for Equation 2. Let $\mathcal{O}$ be the set of non-padding target-token positions in the prompt-target sequence. Prompt tokens are not included in this token-importance normalization; they are handled separately by the question-schema context mask $\mathcal{C}$. For model m, we first normalize each target last hidden state by its feature standard deviation:

$$\widehat{H}_t^{(m)} = \frac{H_t^{(m)}}{\sigma(H_t^{(m)})}. \tag{13}$$

For $s, t \in \mathcal{O}$ and $s \neq t$, the pairwise score and row-normalized relevance are

$$\rho_{s,t}^{(m)} = \frac{(\widehat{H}_s^{(m)})^\top \widehat{H}_t^{(m)}}{\sqrt{d_m}}, \tag{14}$$

$$\alpha_{s \to t}^{(m)} = \frac{\exp(\rho_{s,t}^{(m)})}{\sum_{\substack{u \in \mathcal{O} \\ u \neq s}} \exp(\rho_{s,u}^{(m)})}. \tag{15}$$

Self-pairs and padding positions are masked before the softmax. The unnormalized token weight $\widetilde{w}_t^{(m)}$ is the incoming relevance mass received by token t from all other target tokens, as shown in Equation 2.