When Does a Language Model Help a Graph Neural Network on Relational Data? A Rigorous Study of Graph-Conditioned LLMs and a Portable GNN-for-RAG Architecture

Sunil Kumar, Perception. Contact: contact@perception.club.

Abstract

Relational deep learning (RDL) turns a multi-table database into a temporal heterogeneous graph and learns directly over it with a graph neural network (GNN), avoiding manual feature engineering. A recent line of work (e.g., Rel-LLM) couples such a GNN to a large language model (LLM) by injecting a *graph soft-prompt* into the model, claiming improved predictive accuracy and a path to natural-language interaction. We conduct a careful, paired-baseline study of this claim on the RelBench `rel-f1` / `driver-dnf` entity-classification task. Our central methodological finding is that **the apparent gains of graph-conditioned LLMs are highly sensitive to the strength of the GNN baseline**: when compared against a *well-tuned, paired, best-validation* GNN, every faithful LLM configuration we tested — decoupled vs. joint training, 1.5B vs. 8B models, and single- vs. nested multi-token graph prompts — **ties but does not beat** the GNN on prediction (joint 8B nested prompt: 0.7327 ± 0.010 AUROC vs. GNN 0.7286 ± 0.010; Δ = +0.004, within run-to-run noise). Earlier "wins" we observed (+0.025 to +0.037 AUROC) were artifacts of an under-tuned or cross-run baseline. Conversely, when we compare the GNN encoding against the classical alternative — **flat-text retrieval-augmented generation (RAG)** that serializes the same neighborhood into the prompt — the GNN wins decisively on *both* accuracy (≈0.73 vs. ≈0.34 zero-shot) and *context efficiency* (≈3–17 vs. ≈870 tokens per entity, ≈50×). We reconcile these results with a deployment-oriented architecture: the GNN is the *predictor*; the LLM's value is *retrieval, natural-language interaction, and explanation*. Because the soft-prompt technique is white-box only, we describe a **portable GNN-for-RAG** design in which the GNN acts as a model-agnostic *tool* whose structured output can be consumed by open-weight or frontier LLMs alike. The code, per-run logs, and an implementation manual are available on request.

Keywords: relational deep learning, graph neural networks, large language models, retrieval-augmented generation, tabular/relational ML, RelBench.

1. Introduction

Most enterprise data lives in relational databases — many tables linked by primary- and foreign-key (PK/FK) relationships, with per-row timestamps. The dominant predictive workflow flattens this structure into a single feature table by hand, discarding relational and temporal signal and incurring large engineering cost. **Relational deep learning (RDL)** [Fey et al., 2024] reframes the database as a *relational entity graph* and learns over it end-to-end with a GNN, and **RelBench** [Robinson et al., 2024] standardizes this as a benchmark across classification, regression, and recommendation tasks.

A natural and appealing idea is to combine the GNN with an LLM. The LLM brings world knowledge, natural-language interaction, and in-context reasoning; the GNN brings an efficient, leakage-free encoding of the relevant subgraph. **Rel-LLM** [Wu et al., 2025] realizes this by projecting GNN node embeddings into the LLM's embedding space as a *graph soft-prompt* (via `inputs_embeds`) and fine-tuning with LoRA, reporting accuracy improvements over a GNN baseline. This raises two questions that motivate our study:

1. **Does conditioning an LLM on a GNN actually improve predictive accuracy over a strong GNN?**
2. **If we instead feed the same relational neighborhood to the LLM as plain text (classical RAG), how does a structured GNN encoding compare on accuracy and on token cost?**

We answer both with a deliberately conservative experimental protocol. Our key insight is that **the answer to (1) depends almost entirely on how well the GNN baseline is tuned and how fairly it is compared.** Using a *paired, same-seed, best-validation* GNN head as the baseline — rather than a cross-run or under-tuned number — we find that graph-conditioned LLMs tie but do not surpass the GNN, even at 8B scale with the full nested-prompt recipe. For (2), the GNN's structured encoding is both far more accurate and ≈50× more token-efficient than zero-shot flat-text RAG.

These results are individually unsurprising in hindsight but, taken together, sharpen the design space: **for relational *prediction*, the GNN is the engine; the LLM earns its place through *retrieval, language, and explanation*, not through extra predictive accuracy.** Because the soft-prompt mechanism cannot attach to a frontier API model (which accepts text, not embeddings) and is not even portable across open-weight models (the projection is per-LLM), we argue for a **portable GNN-for-RAG** architecture in which the GNN is a standalone, model-agnostic *tool*. We detail this design and accompany it with a practitioner manual.

Contributions.

- A **paired-baseline methodology** for evaluating graph-conditioned LLMs, and evidence that prior positive results are fragile to baseline strength (Section 5.1). We document a concrete false-positive (+0.037 AUROC) that vanished under a fair baseline.
- A **systematic sweep** over decoupled/joint training, 1.5B/8B model scale, and single/nested graph prompts, all of which **tie** a well-tuned GNN on `rel-f1` / `driver-dnf` (Section 5.1).
- A **GNN-vs-RAG comparison** isolating the value of structured graph encoding over flat-text serialization on both accuracy and token efficiency (Section 5.2).
- A **portable GNN-for-RAG architecture** that reconciles these findings for deployment with open-weight and frontier LLMs, plus an open implementation (Section 6, Appendix).

---

2. Related Work

Relational deep learning and RelBench. RDL [Fey et al., 2024] casts a database as a temporal heterogeneous graph (the *relational entity graph*) and learns task-specific predictions with message-passing GNNs over leakage-free temporal neighbor samples. RelBench [Robinson et al., 2024] provides datasets and standardized tasks; the reference GNN combines column encoders [PyTorch-Frame; Hu et al., 2024], a heterogeneous GraphSAGE backbone [Hamilton et al., 2017], and per-task heads. **RelGNN**

[Yuan et al., 2025] introduces atomic-route composite message passing for state-of-the-art classification/regression, and **ContextGNN** [Fey et al., 2024b] targets recommendation with a hybrid pairwise/two-tower head.

LLMs over tabular and relational data. A body of work serializes rows/tables into text for in-context learning, which is sensitive to prompt design and context length. **Rel-LLM** [Wu et al., 2025] instead injects GNN embeddings into a frozen LLM as a soft-prompt with LoRA adaptation, reporting gains over a GNN baseline — the primary claim our study scrutinizes.

Parameter-efficient adaptation and quantization. LoRA [Hu et al., 2021] and 4-bit QLoRA [Dettmers et al., 2023] make LLM fine-tuning feasible on a single GPU; we use 4-bit NF4 inference/adaptation throughout.

Retrieval-augmented generation. RAG [Lewis et al., 2020] augments an LLM with retrieved context. Our "text-RAG" baseline is the relational instantiation: retrieve an entity's temporally-valid neighborhood, serialize to text, and prompt the LLM. This is the model-agnostic alternative to the soft-prompt, and the comparison isolates the effect of *structured* (GNN) vs. *flat-text* encoding.

---

3. Method

3.1 Database → Relational Entity Graph (REG)

Given tables with declared PK/FK constraints and per-row timestamps, we construct a heterogeneous graph: each row is a typed node; each PK→FK reference is a typed edge. **Bridge/junction tables** (rows with ≥2 FKs) are detected and used to form single-hop cross-entity connections ("atomic routes"). For a prediction about entity e at *seed time* t, temporal neighbor sampling admits only rows with timestamp $< t$, preventing label leakage. Column values are encoded per-type by a column encoder, yielding initial node features.

3.2 GNN core (the predictor)

The GNN core ("RelCore") is a heterogeneous message-passing encoder over sampled temporal neighborhoods, with a task head (binary classification / regression / link prediction). Our backbone follows the RDL reference (heterogeneous GraphSAGE with temporal encoding); we additionally implement an **atomic-route composite** variant (RelGNN-style) with a learned bridge-routing gate. Training uses focal loss [Lin et al., 2017] for imbalanced binary tasks; model selection is by **best validation** AUROC. This core is the system's *predictor* and is the strong baseline that all LLM variants are measured against.

3.3 Two ways to attach an LLM

We study two mechanisms for letting an LLM use the same relational neighborhood.

(a) Graph soft-prompt (white-box; "RelLM"). GNN embeddings are projected into the LLM's token-embedding space and prepended as virtual tokens via `inputs_embeds`; a LoRA-adapted, frozen LLM reads a Yes/No answer from the next-token distribution. We evaluate:

- *Decoupled*: GNN frozen, embeddings precomputed, only the LLM side trains.

- *Joint*: gradients flow GNN↔LLM end-to-end (the paper's recipe), with gradient accumulation to raise the effective GNN batch under the LLM's small physical batch.
- *Single vs. nested prompt*: one pooled graph token, vs. a nested multi-token prompt (a seed token plus one token per incoming relation, each the mean of neighbors along that relation).

This mechanism is **white-box only**: it requires access to the LLM's embedding matrix and hidden states, so it cannot target a frontier API and is not portable across open-weight models (the projection is per-LLM).

(b) Flat-text RAG (model-agnostic). The entity and its temporally-valid neighbors are serialized to JSON-like text, placed in a prompt with a task question, and the LLM's Yes/No logits are read (zero-shot). This is *model-agnostic*: the same prompt shape ports to any causal LM and to frontier APIs. It is the classical RAG baseline against which the GNN's structured encoding is measured.

3.4 Evaluation protocol (the crux)

Comparisons between an LLM variant and "the GNN" are only meaningful if the GNN baseline is (i) **well tuned**, (ii) **paired** — trained on the *same* embeddings/seed where applicable — and (iii) selected by **best validation**, not final epoch. We additionally report **multi-seed mean ± std**, because run-to-run variance on this task is ≈±0.01 AUROC, large enough to manufacture spurious single-run "wins." Section 5.1 shows that relaxing any of these conditions produces a false positive.

---

4. Experimental Setup

Task. RelBench `rel-f1` / `driver-dnf`: predict whether a driver will record a did-not-finish (DNF) in the near future (binary classification; positive = DNF, a minority class). Metric: test AUROC. We chose this task as small but knowledge-rich (Formula 1 entities are well represented in LLM pretraining), which is the most favorable setting for an LLM to help.

Models. GNN core as in Section 3.2. LLMs: `Qwen2.5-1.5B-Instruct` (fp16) and `Qwen3-8B` (4-bit NF4) via HuggingFace Transformers + PEFT (LoRA). For classification we read the final-token Yes/No logits (`logits_to_keep=1`, computing only the last position).

Hardware. A single NVIDIA T4 (16 GB) for the 1.5B inference and a single L4 (24 GB) for 8B joint training; both via cloud VMs. GNN training fits comfortably on a single 16 GB GPU.

Baseline protocol. Paired, best-validation GNN head; 3 seeds (42, 1, 2); mean ± std reported. Text-RAG is evaluated zero-shot on a 1000-row sample of val/test.

---

5. Results

5.1 Graph-conditioned LLMs tie, but do not beat, a well-tuned GNN

Table 1. Entity prediction on `rel-f1` / `driver-dnf` (test AUROC). All LLM rows use the graph soft-prompt ("RelLM").

Method	AUROC (mean ± std)	Note
Published RDL (reference)	0.726	[Robinson et al., 2024]
GNN (ours, well-tuned, best-val)	**0.729 ± 0.010**	the bar
RelCore composite routing (RelGNN-style)	≈ GNN (gate → 0)	no significant gain
RelLM decoupled (1.5B / 8B)	tie / slightly behind	opaque frozen embedding
RelLM joint, 1.5B, single token	0.7251	tie
RelLM joint, 8B, single token	0.7285	tie (+0.004 vs. fair GNN)
RelLM joint, 8B, nested prompt (full recipe)	**0.7327 ± 0.010**	**tie (+0.004, within noise)**

The strongest LLM configuration — joint training, 8B model, nested multi-token graph prompt — reaches 0.7327 ± 0.010 vs. the paired GNN's 0.7286 ± 0.010, a difference of **+0.004 AUROC**, well within the ±0.01 run-to-run noise. The **per-seed deltas straddle zero** (−0.009, −0.005, +0.027): the single positive seed reflects a low-variance draw of the GNN baseline rather than a systematic LLM advantage.

A documented false positive. In an earlier iteration we observed RelLM "beating" the GNN by +0.025 to +0.037 AUROC. Investigation showed these were **baseline artifacts**: (i) a *cross-run* GNN number rather than a same-seed paired head, and (ii) a GNN baseline selected at the final epoch under a small-batch configuration rather than best-validation under the canonical configuration. Re-running with the paired best-validation GNN erased the gain. We report this explicitly because it is, we believe, the central practical hazard in this subfield: **the measured benefit of a graph-conditioned LLM is dominated by how the GNN baseline is constructed.**

Recommendation track (supporting evidence). On a recommendation task (`rel-trial` / `condition-sponsor-run`), a text-free two-tower pipeline reached MAP ≈ 0.015 vs. a published text-on ≈0.114; the head choice barely mattered (published two-tower 11.36 ≈ ID-GNN 11.48). The lever on these datasets is **features/text, not GNN architecture** — consistent with Table 1.

5.2 The GNN decisively beats flat-text RAG on accuracy *and* token cost

Table 2. GNN encoding vs. flat-text RAG into the *same* 1.5B-class LLM, on `rel-f1` / `driver-dnf`.

Approach	AUROC	tokens / entity
GNN (no LLM)	**0.729**	—
RelLM (GNN soft-prompt → LLM)	0.725	**≈3–17**
Text-RAG (flat serialized records → LLM, zero-shot)	**≈0.34** (degenerate, f1 = 0)	**≈870**

Zero-shot 1.5B text-RAG is **non-discriminative** (AUROC ≈0.34, below chance, with f1 = 0 indicating near-constant predictions), and was robust to prompt framing (three question phrasings yielded 0.30–0.42). Meanwhile it consumes ≈870 prompt tokens per entity, versus ≈3–17 tokens for the GNN encoding — an **≈50× context-efficiency advantage**. The GNN's structured encoding both extracts the predictive signal that a small zero-shot LLM cannot pull from flat text *and* does so at a tiny fraction of the context cost.

Caveat (fairness). The text-RAG point is *zero-shot* with a *small* (1.5B) model; the degeneracy partly reflects model capacity. A **fine-tuned** text-RAG (LoRA on serialized text) and/or a **frontier model** (Claude/Gemini/GPT) is the fairer *accuracy* baseline and is the natural next experiment. The **token-efficiency** conclusion, however, is model-independent and stands regardless. (We also identified and fixed an engineering subtlety — right-truncation silently dropping the trailing question on long prompts — which we correct in the released code.)

---

6. Discussion: a portable GNN-for-RAG architecture

Two facts shape deployment. First, **the LLM does not improve relational prediction** over a well-tuned GNN at the scales we tested (Section 5.1). Second, **structured GNN encoding dominates flat-text RAG** on both accuracy and token cost (Section 5.2). Together they imply a clean division of labor:

> **The GNN is the predictor; the LLM is the interface** — for retrieval, natural-language exploration, and explanation — not for squeezing extra accuracy.

The soft-prompt realization of "GNN + LLM," however, is white-box only and per-LLM. To make the GNN *portable* — attachable to open-weight *and* frontier models — we cast it as a **model-agnostic tool**:

- **GNN core (standalone).** Trained on the enterprise schema and (optionally) business terminology; exposes `predict(entity, seed_time)` and `retrieve(entity, seed_time) → structured context`.
- **Open-weight path.** May optionally use the soft-prompt for tightest coupling, *or* the tool path below.
- **Frontier path (portable).** The GNN's output is rendered as **compact structured text** or returned as a **tool result** to a frontier LLM via its tool-use API. The LLM orchestrates: it calls the GNN tool for predictions/retrieval and uses its own language ability for the user-facing answer and explanation. This requires no access to model internals and is identical across providers.

This is exactly the GNN-vs-RAG axis of Section 5.2, used constructively: the GNN tool returns a ≈10-token structured encoding (a calibrated prediction plus a small set of the most relevant evidence rows) instead of an ≈870-token raw dump, giving the orchestrating LLM a cheaper, higher-signal context. The accompanying implementation manual specifies the tool schema and the agent loop for both open-weight (HuggingFace) and frontier (Anthropic Claude tool-use) backends.

---

7. Limitations

- **Single pilot task.** Our deepest results are on one task (`rel-f1` / `driver-dnf`). The methodology (paired best-val baseline, multi-seed) generalizes, but absolute conclusions should be re-checked per dataset; tasks where pretraining knowledge is decisive may favor the LLM more.
- **Text-RAG accuracy is a floor, not a ceiling.** It is zero-shot + 1.5B; fine-tuned and frontier RAG are unmeasured here and are the right next baselines.
- **Scale.** We test up to 8B; we do not rule out gains at much larger scale, though our trend (no gain from 1.5B→8B) is not encouraging for the prediction-accuracy claim specifically.

- **Soft-prompt portability.** The white-box soft-prompt is not portable; our portable design uses the tool path, whose end-to-end accuracy with a fine-tuned frontier orchestrator we leave to future work.

---

8. Conclusion

We studied whether a language model helps a graph neural network on relational prediction, under a deliberately fair, paired, multi-seed protocol. Across decoupled/joint training, 1.5B/8B scale, and single/nested graph prompts, graph-conditioned LLMs **tie but do not beat** a well-tuned GNN — and we show that previously observed "wins" are artifacts of an under-tuned baseline. In contrast, structured GNN encoding **decisively beats** flat-text RAG on both accuracy and token efficiency (≈50×). The practical conclusion is an architecture in which the **GNN is the predictor and the LLM is a model-agnostic interface** for retrieval, language, and explanation. The code, per-run logs, a consolidated results document, and an implementation manual are available on request to support reproduction and deployment.

Reproducibility

The installable `relational_llm/` package, notebooks, the cloud-VM recipe, per-run JSON logs, and a consolidated results document are available on request. Engineering details (temporal sampling, focal loss, the `logits_to_keep` last-token-logits optimization, prompt-truncation handling, and cloud-GPU/stack pitfalls) are documented in the project journal and product manual.

References

1. M. Fey, W. Hu, K. Huang, J. E. Lenssen, R. Ying, J. Leskovec, and others. *Relational Deep Learning: Graph Representation Learning on Relational Databases.* ICML Position Paper, 2024. arXiv:2312.04615.
2. J. Robinson, R. Ranjan, W. Hu, K. Huang, J. Zhang, et al. *RelBench: A Benchmark for Deep Learning on Relational Databases.* NeurIPS Datasets and Benchmarks, 2024. arXiv:2407.20060.
3. W. Wu, et al. *Rel-LLM: Large Language Models for Relational Deep Learning* (graph soft-prompt conditioning of LLMs over relational entity graphs). 2025. arXiv:2506.05725.
4. T. Yuan, et al. *RelGNN: Composite Message Passing for Relational Deep Learning.* 2025. arXiv:2502.06784.
5. M. Fey, et al. *ContextGNN: Beyond Two-Tower Recommendation Systems.* 2024. arXiv:2411.19513.
6. W. Hamilton, R. Ying, and J. Leskovec. *Inductive Representation Learning on Large Graphs (GraphSAGE).* NeurIPS, 2017. arXiv:1706.02216.
7. W. Hu, et al. *PyTorch Frame: A Modular Framework for Multi-Modal Tabular Learning.* 2024. arXiv:2404.00776.
8. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models.* ICLR, 2022. arXiv:2106.09685.

9. T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. *QLoRA: Efficient Finetuning of Quantized LLMs.* NeurIPS, 2023. arXiv:2305.14314.
10. P. Lewis, E. Perez, A. Piktus, et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS, 2020. arXiv:2005.11401.
11. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. *Focal Loss for Dense Object Detection.* ICCV, 2017. arXiv:1708.02002.
12. A. Vaswani, et al. *Attention Is All You Need.* NeurIPS, 2017. arXiv:1706.03762.
13. Z. Hu, Y. Dong, K. Wang, and Y. Sun. *Heterogeneous Graph Transformer (HGT).* WWW, 2020. arXiv:2003.01332.
14. Qwen Team. *Qwen2.5 Technical Report.* 2024. arXiv:2412.15115.

Correspondence: Sunil Kumar, Perception — contact@perception.club.