Large Language Models are Good Relational Learners

Fang Wu
Stanford University
fangwu97@stanford.edu

Vijay Prakash Dwivedi
Stanford University
vdwivedi@cs.stanford.edu

Jure Leskovec†
Stanford University
jure@cs.stanford.edu

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities across various domains, yet their application to relational deep learning (RDL) remains underexplored. Existing approaches adapt LLMs by traversing relational links between entities in a database and converting the structured data into flat text documents, but this text-based serialization disregards critical relational structures, introduces redundancy, and often exceeds standard LLM context lengths. We introduce Rel-LLM, a novel architecture that employs a graph neural network (GNN) based encoder to create structured relational prompts for LLMs within a retrieval-augmented generation (RAG) framework. Unlike traditional text-based serialization approaches, our method preserves the inherent relational structure of databases while enabling LLMs to effectively process and reason over complex entity relationships. Specifically, the GNN encoder extracts a local subgraph around an entity to build feature representations that contain relevant entity relationships and temporal dependencies. These representations are transformed into structured prompts using a denormalization process, effectively allowing the LLM to reason over relational structures. Through extensive experiments, we demonstrate that Rel-LLM outperforms existing methods on key RDL tasks, offering a scalable and efficient approach to integrating LLMs with structured data sources. Code is available at `https://github.com/smiles724/Rel-LLM`.

1 Introduction

Large language models (LLMs) (Zhao et al., 2023; Minaee et al., 2024), with their exceptional generalization capabilities in zero or few-shot settings (Raffel et al., 2020; Wei et al., 2022; Xu et al., 2024; Tang et al., 2025; Chen et al., 2025), have become foundational tools in diverse areas such as natural language processing (Achiam et al., 2023), computer vision (Liu et al., 2024), and information retrieval (Hou et al., 2024). These advances stem from a series of groundbreaking techniques, including web-scale unsupervised pretraining (Brown et al., 2020), instruction finetuning (Wei et al., 2022), and methods to ensure value alignment (Wolf et al., 2023).

Despite remarkable achievements, LLMs face challenges in processing and reasoning over complex structured data, such as relational databases (Fey et al., 2024; Qin et al., 2024). Such databases, comprising vast quantities of interconnected data organized into rows and columns, expose LLMs' known restrictions, including hallucinations (Zhang et al., 2023), susceptibility to knowledge cutoffs (Gao et al., 2023), and inefficiencies in capturing explicit relationships in data (Wydmuch et al., 2024). Though retrieval-augmented generation (RAG) (Lewis et al., 2020) has shown promise in alleviating some of these constraints, existing RAG techniques do not model arbitrary connectivity patterns explicitly present in real-world structured data, especially when dealing with relational databases, the backbone of global enterprise data, with approximately 73% of the world's data (Li et al., 2024).

Relational databases differ fundamentally from single tables due to their interconnected structures. They comprise collections of tables linked by primary and foreign keys, representing one-to-many and many-to-many relationships. This intrinsic complexity poses unique challenges for deep learning (DL), particularly for leveraging the rich semantic and structural information encoded in these links. Recent years have witnessed an increased interest in Relational Table Learning (Džeroski, 2010; Zahradník et al., 2023) to unlock the predictive power of these databases. Furthermore, resources such as GitTables, TabLib (Eggert et al., 2023), and other tabular evaluations (Gardner et al., 2024) have emerged to facilitate this research area.

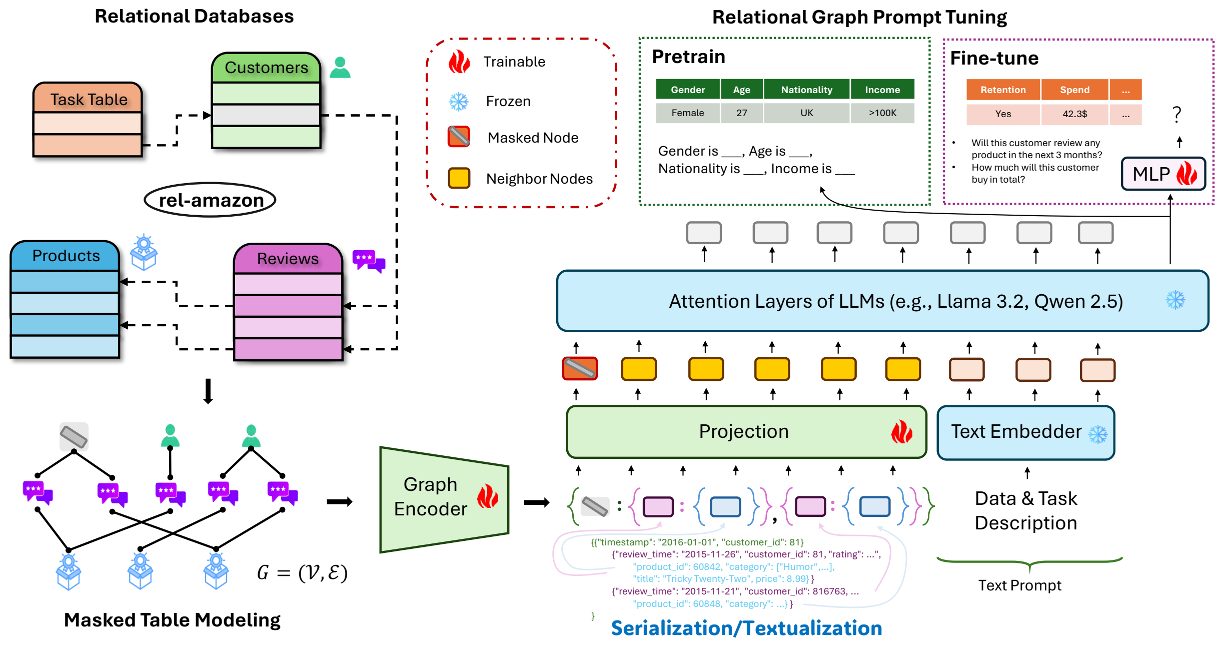

Relational Databases
Task Table
Customers
rel-amazon
Products
Reviews
Masked Table Modeling
$G = (\mathcal{V}, \mathcal{E})$
Graph Encoder
Trainable
Frozen
Masked Node
Neighbor Nodes
Relational Graph Prompt Tuning
Pretrain
Gender | Age | Nationality | Income
Female | 27 | UK | >100K
Gender is ___, Age is ___, Nationality is ___, Income is ___
Fine-tune
Retention | Spend | ...
Yes | 42.3$ | ...
Will this customer review any product in the next 3 months?
How much will this customer buy in total?
?
MLP
Attention Layers of LLMs (e.g., Llama 3.2, Qwen 2.5)
Projection
Text Embedder
Data & Task Description
Text Prompt
{{"timestamp": "2016-01-01", "customer_id": 81}
{"review_time": "2015-11-26", "customer_id": 81, "rating": ...",
"product_id": 60842, "category": ["Humor",...],
"title": "Tricky Twenty-Two", price": 8.99} }
{"review_time": "2015-11-21", "customer_id": 816763, ...
"product_id": 60848, "category": ...} }
}
Serialization/Textualization

Figure 1: Overview of the Rel-LLM framework. Input from a relational database is fed to an LLM through a graph encoder that captures local connectivity entities in different tables. The graph encoder, along with other trainable components, is pretrained with masked table modeling and fine-tuned for downstream tasks such as customer behavior prediction.

Benchmarks like CTU (Motl and Schulte, 2015), SJTUTable (Li et al., 2024), and RelBench (Robinson et al., 2024) provide datasets for relational tasks. To better exploit them, a favorable direction (Fey et al., 2024; Robinson et al., 2024) recasts relational databases as graph representations, which represent entities as nodes, primary-foreign key relationships as edges, and extract node features using deep tabular models, effectively capturing structural and semantic intricacies.

However, applying LLMs to relational databases remains underexplored. Recent studies (Li et al., 2024; Wydmuch et al., 2024) adapt LLMs by traversing relational links and constructing text-based documents as model inputs. While this aligns with the text-based nature of LLMs, it has significant drawbacks. First, converting relational data into flat text obscures the unique relationships between tables, especially primary-foreign key connections, diminishing LLMs' ability to fully leverage structural insights. Second, the document construction process often results in substantial redundancy, particularly in cases involving nested loop interconnections. This redundancy can lead to repetitive entities appearing multiple times in prompts, complicating the task of extracting key information and increasing computational inefficiency. Finally, real-world relational databases frequently implicate massive datasets with numerous heterogeneous data types and relationships (Fey et al., 2023). Processing such large contexts often exceeds the maximum input length of standard LLMs, leading to substantial memory overhead, reduced computational efficiency, and reduced performance (Wang et al., 2024). These challenges underscore the need for innovative approaches to harness LLMs for relational data processing, driving further research into scalable, structure-aware methodologies capable of unlocking the full potential of relational databases.

In this work, we propose Rel-LLM (see Figure 1), a novel framework that combines the structured reasoning capabilities of GNNs with the linguistic fluency of LLMs through RAG. Central to Rel-LLM is a graph prompt tuning mechanism wherein structured graph embeddings are projected into the LLM's latent space, conditioning its response generation without requiring full fine-tuning. This enables the model to reason over structured and unstructured information while preserving the relational semantics inherent in databases. Our approach leverages temporal-aware subgraph sampling to ensure causal consistency, a heterogeneous GNN encoder for relational feature extraction, and a denormalization-based prompt construction that organizes structured data into a format optimized for LLM processing. In addition, Rel-LLM is pretrained using a self-supervised objective that aligns graph-text representations through masked attribute prediction. This pretraining phase ensures that the model can effectively reconstruct entity attributes from corrupted subgraphs, enhancing its ability to reason over relational data and realize zero-shot prediction in low-data regimes. We evaluate our model on benchmark RDL datasets, demonstrating notable improvements in predictive accuracy, relational reasoning, and temporal consistency compared to existing baselines. A recap on related works is in Appendix 5.

2 Preliminaries

Relational Data. A relational database (Fey et al., 2024) $(\mathcal{T}, \mathcal{L})$ is comprised of a collection of tables $\mathcal{T} = \{T_1, \ldots, T_n\}$, and links between tables $\mathcal{L} \subseteq \mathcal{T} \times \mathcal{T}$. A link $L = (T_{\text{fkey}}, T_{\text{pkey}}) \in \mathcal{L}$ between tables exists if a foreign key column in T_{fkey} points to a primary key column of T_{pkey}. Each table is a set $T = \{v_1, \ldots, v_{n_T}\}$, whose elements $v_i \in T$ are called rows or entities. Each entity $v \in T$ has four constituent parts $v = (p_v, \mathcal{K}_v, x_v, t_v)$.

Specifically, p_v is the primary key uniquely identifying the entity v. $\mathcal{K}_v \subseteq \{p_{v'} : v' \in T' \wedge (T, T') \in \mathcal{L}\}$ are the foreign keys and define links between element $v \in T$ to elements $v' \in T'$, where $p_{v'}$ is the primary key of an entity v' in table T'. a_v is an attribute that holds the entity's information. t_v is an optional timestamp, indicating the time an event occurred. Generally, the attributes in table T contain a tuple of d_T values: $a_v = (a_v^1, ..., a_v^{d_T})$. Besides, all entities in the same table have the same columns but values can be absent.

Relational Entity Graph. The schema graph (SchG) describes the table-level data structure. Given a relational database $(\mathcal{T}, \mathcal{L})$, $\mathcal{L}^{-1} = \{(T_{\text{pkey}}, T_{\text{fkey}}) \mid (T_{\text{fkey}}, T_{\text{pkey}}) \in \mathcal{L}\}$ denotes its inverse link set. Then, SchG is the graph $(\mathcal{T}, \mathcal{R})$ that arises from the relational database $(\mathcal{T}, \mathcal{L})$, with node set $\mathcal{T}$ and edge set $\mathcal{R} = \mathcal{L} \cup \mathcal{L}^{-1}$. Inverse links ensure that all tables within the SchG are reachable.

Each relational entity graph (REG) is a heterogeneous graph denoted as $G = (\mathcal{V}, \mathcal{E}, \phi, \psi)$. REG has a node set $\mathcal{V}$, an edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, node type mapping functions $\phi : \mathcal{V} \to \mathcal{T}$, and edge type mapping functions $\psi : \mathcal{E} \to \mathcal{R}$. Here, each node $v \in \mathcal{V}$ belongs to a node type $\phi(v) \in \mathcal{T}$ and each edge $e \in \mathcal{E}$ belongs to an edge type $\psi(e) \in \mathcal{R}$. The sets $\mathcal{T}$ and $\mathcal{R}$ from the SchG define REG's node and edge types.

Given a SchG $(\mathcal{T}, \mathcal{R})$ with tables $T = \{v_1, \ldots, v_{n_T}\} \in \mathcal{T}$, the node set in REG is defined as the union of all entries in all tables $\mathcal{V} = \bigcup_{T \in \mathcal{T}} T$. Its edge set is then defined as the entity-level pairs that arise from the primary-foreign key relationships in the database, written as:

$$\mathcal{E} = \{(v_1, v_2) \in \mathcal{V} \times \mathcal{V} \mid p_{v_2} \in \mathcal{K}_{v_1} \text{ or } p_{v_1} \in \mathcal{K}_{v_2}\}. \tag{1}$$

Moreover, each REG is equipped with three categories of key information. First is type mapping functions $\phi : \mathcal{V} \to \mathcal{T}$ and $\psi : \mathcal{E} \to \mathcal{R}$. They map nodes and edges to respective elements of the SchG, making the REG heterogeneous. We set $\phi(v) = T$ for all $v \in T$ and $\psi(v_1, v_2) = (\phi(v_1), \phi(v_2)) \in \mathcal{R}$ if $(v_1, v_2) \in \mathcal{E}$. Second is the time mapping function $\tau : \mathcal{V} \to \mathcal{D}$. It maps nodes to their timestamp: $\tau : v \mapsto t_v$, which introduces time as a central component and establishes the temporality of the graph. The value $\tau(v)$ denotes the point in time in which the table row v became available or $-\infty$ in the case of non-temporal rows. The last is the embedding vectors $\mathbf{h}_v \in \mathbb{R}^{d_{\phi(v)}}$ for each $v \in \mathcal{V}$, which contains an embedding vector for each node in the graph. Initial embeddings are obtained via multimodal column encoders (Hu et al., 2024).

3 Methods

In this section, we introduce Rel-LLM, a novel architecture tailored for RDL, which integrates the strengths of GNNs, LLMs, and RAG. To enable efficient fine-tuning while preserving the LLM's pretrained language capabilities, we adopt a soft prompting approach by freezing the LLM and conditioning it on the structured outputs of the GNN.

3.1 Graph Representation Acquisition

Graph prompt tuning leverages the rich relational structure of databases (He et al., 2024) to improve downstream predictions by extracting relevant temporal and structural information.

Temporal-aware Subgraph Sampling. We employ temporal neighbor sampling to construct a subgraph centered around each entity node at a given seed time t^*. The seed time t represents the point in history at which a prediction is made. To maintain causality and prevent information leakage, the model exclusively incorporates data from prior to the seed time and ensures no future data is included. During mini-batch training, all nodes within the sampled subgraph are guaranteed to have timestamps earlier than t^* (Hamilton et al., 2017). This strategy systematically eliminates temporal leakage and results in a retrieved subgraph denoted as $G^\diamond = (\mathcal{V}^\diamond, \mathcal{E}^\diamond)$.

Graph Encoder. To capture the relational structure of $G^\diamond$, we utilize a heterogeneous variant of the GraphSAGE model (Hamilton et al., 2017) with sum-based neighbor aggregation. Given initial node embeddings $\left\{\mathbf{h}^{(0)}_{v_1}, ..., \mathbf{h}^{(0)}_{v_{n_T}}\right\}$, an L-layer GNN iteratively updates the embeddings through message passing, producing $\mathbf{h}^{(L)}_i \in \mathbb{R}^{d_g}$, where d_g denotes the graph encoder's output dimension. The encoding process is formally defined as:

$$\mathbf{h}^{(L)}_i = \mathrm{GNN}_{\phi_1}\left(G^\diamond\right), \quad \mathbf{h}^{(L)}_g = \mathrm{POOL}\left(\mathbf{h}^{(L)}_i\right), \tag{2}$$

where POOL represents a mean pooling operation aggregating node embeddings.

Projection Layer. To align graph embeddings with the LLM's vector space, we introduce a projection layer using a multilayer perceptron (MLP):

$$\hat{\mathbf{h}}_i = \mathrm{MLP}_{\phi_2}\left(\mathbf{h}^{(L)}_i\right), \hat{\mathbf{h}}_g = \mathrm{MLP}_{\phi_2}\left(\mathbf{h}^{(L)}_g\right), \tag{3}$$

where $\hat{\mathbf{h}}_i, \hat{\mathbf{h}}_g \in \mathbb{R}^{d_l}$ and d_l corresponds to the hidden dimension of the LLM.

3.2 Graph Prompt Construction

Retrieval-augmented systems for LLMs often utilize structured prompts derived from external data sources (Sundar and Heck, 2023; Qin et al., 2022; Ye et al., 2023; Wu and Li, 2024). In the context of relational databases, constructing effective prompts is challenging due to heterogeneous tabular data, inconsistent feature scales, and complex interdependencies. Given a relational database $(\mathcal{T}, \mathcal{L})$ and a target prediction task, our approach generates structured documents that encapsulate relevant relational information. Each document is associated with a specific entity v^* from the target table T^* and consists of the following structured components.

Task Context. Each structured prompt begins with a task description and a question prompt, denoted as x_{task} and x_{quest}. They detail the database schema, relevant entity relationships, and the objective (e.g., classification or regression). The task description follows templates inspired by prior work (Robinson et al., 2024).

Denormalization Process To preserve the relational structure, we apply a denormalization process to every entity in the graph prompt (Wydmuch et al., 2024). The denormalization follows these recursive steps:

1. For the seed entity v^*, we recursively follow links from its primary key $\top^*_{\text{pkey}} \in T_{\text{pkey}}$ to foreign keys T_{fkey}, selecting up to n_{nest} entities from each joined table in a breadth-first manner, up to a recursion depth of ζ.

2. Aggregate the graph node embedding of all linked entities $\left\{\left\{\hat{\mathbf{h}}_{i,j}\right\}_{i=1}^{n_{\text{nest}}}\right\}_{j=1}^{\zeta}$ into the document representation.

3. Avoid redundant inclusion by skipping tables already visited in prior denormalization steps.

Efficient execution of this process is ensured by leveraging hash indexes on all primary and foreign keys, allowing rapid retrieval.

Serialization Format Designing an effective prompt is a non-trivial task, and many research topics have branched out from prompt engineering alone (Fang et al., 2024). Here, the fully denormalized entity representation is serialized as a JSON object:

- Each entity v_i is represented as a graph neural object $\hat{\mathbf{h}}_i$ reflecting its attributes.

- Linked entities appear as nested structures within the parent entity v_i as $\{v_i : \{v_{i,1}, ..., v_{i,n_{\text{nest}}}\}\}$, where $(\top_{\text{pkey}_i}, \top_{\text{fkey}_{i,j}}) \in \mathcal{L}$ for $j = 1, ..., n_{\text{nest}}$. This reduces the need for multi-hop inference.

This leads to the complete graph prompt:

$$\mathbf{H}^* = \{\hat{\mathbf{h}}^* : \{\{\hat{\mathbf{h}}^*_i : \{\hat{\mathbf{h}}^*_{i,j} : \{...\}\}_{j=1}^{n_{\text{nest}}}\}_{i=1}^{n_{\text{nest}}}\}\}, \tag{4}$$

where $\mathbf{H}^*$ contains at most $n_{\text{nest}}\zeta$ elements. The choice of JSON format is motivated by prior findings (Singha et al., 2023), demonstrating its effectiveness in encoding tabular and relational data for LLMs. By structuring input documents in this manner, we enable LLMs to effectively reason over relational data while preserving contextual and temporal integrity. Figure 4 in the Appendix provides a clear visualization of this process.

3.3 Answer Generation

Text Embedder. To leverage the text-reasoning capabilities of LLMs, we transform the task context x_{task} and the question prompt x_{quest} to an embedding $\mathbf{h}_{\text{text}}$ using a text embedder, which is the first layer of a pretrained and frozen LLM:

$$\mathbf{h}_{\text{text}} = \text{TextEmbedder}\left([x_{\text{task}}; x_{\text{quest}}]\right), \quad (5)$$

where $\mathbf{h}_{\text{text}} \in \mathbb{R}^{n_{\text{text}} \times d_l}$. $[\cdot;\cdot]$ represents the concatenation operation, and n_{text} is the number of total textual tokens.

LLM Generation with Graph Prompt Tuning. The final stage involves generating the answer Y given the graph neural prompt $\mathbf{H}^*$, acting as a soft prompt, and the text embedder output $\mathbf{h}_{\text{text}}$. These inputs are fed through the self-attention layers of a pretrained frozen LLM, with parameter θ. The generation process is represented as follows:

$$\begin{aligned} p_{\theta,\phi_1,\phi_2}\left(Y \mid G^{\diamond}, x_q\right) = \\ \prod_{i=1}^{r} p_{\theta,\phi_1,\phi_2}\left(y_i \mid y_{<i}, [\mathbf{H}^*; \mathbf{h}_{\text{text}}]\right), \end{aligned} \quad (6)$$

where $[\mathbf{H}^*; \mathbf{h}_{\text{text}}]$ concatenates the graph prompt token $\mathbf{H}^*$ and the text embedder output $\mathbf{h}_{\text{text}}$. While θ is frozen, the graph token $\mathbf{H}^*$ receives gradients, enabling the optimization of the parameters of the graph encoder $\phi_1(\cdot)$ and the projection layer $\phi_2(\cdot)$ through standard backpropagation.

Answer Generation Strategy. We explore three sorts of answer generation methods, including plain text, token distribution, and MLP transformation.

- Plain text generation directly outputs a sequence of tokens forming the human-readable text. It is direct and interpretable but lacks richness for complex tasks.
- Token distribution outputs a distribution over possible tokens, which can be useful for probabilistic or multi-modal tasks.
- MLP transformation applies a lightweight neural network to refine or project the LLMs' latent representations into task-specific spaces.

Empirical evaluations on RelBench reveal that different tasks benefit from distinct strategies, underscoring the importance of selecting an appropriate answer generation method based on task-specific requirements.

3.4 Pretraining

We design a pretraining objective to align the graph-text representations through self-supervised attribute prediction.

Masked Subgraph Construction. We randomly select a subset of nodes $\mathcal{V}_{\text{mask}} \subseteq \mathcal{V}^{\diamond}$ with probability p_{mask}. Then for each masked entity $v_{\text{mask}} \in \mathcal{V}_{\text{mask}}$, we build a temporal-aware subgraph $G^{\diamond}_{\text{mask}}$ as described before. Besides, the raw feature of those masked entities v_{mask} is replaced with a learnable mask token $\mathbf{h}_{\text{mask}} \in \mathbb{R}^{d_g}$, creating a corrupted subgraph.

Graph Encoding with Noise. We forward and process $G^{\diamond}_{\text{mask}}$ through the graph encoder and projection layer:

$$\hat{\mathbf{h}}_{\text{mask}} = \text{MLP}\,\phi_2\left(\text{GNN}\,\phi_1\left(G^{\diamond}_{\text{mask}}\right)\right). \quad (7)$$

Masked Attribute Prediction. After obtaining $\hat{\mathbf{h}}_{\text{mask}}$ for each masked node $v_i \in \mathcal{V}_{\text{mask}}$, we require the LLM to reconstruct its original attributes through the plain text generation. Let $A_i = \{(k_j, a_j)\}_{j=1}^{d^T}$ denote the column-value pairs of v_i 's attributes. We:

1. Apply random permutation π to the column order: $A_i^{\pi} = \left\{\left(k_{\pi(j)}, a_{\pi(j)}\right)\right\}_{j=1}^{m}$
2. Format the target sequence as Y_i ="$k_{\pi(1)}$ is $a_{\pi(1)}, \ldots, k_{\pi(m)}$ is $a_{\pi(m)}$" and the masked prompt template as X_i ="$k_{\pi(1)}$ is [MASK], $\ldots, k_{\pi(m)}$ is [MASK]", where [MASK] is the mask token of LLMs.

The pretraining loss is computed as:

$$\begin{aligned} \mathcal{L}_{\text{pretrain}}(\phi_1, \phi_2, \mathbf{h}_{\text{mask}}) = -\frac{1}{|\mathcal{V}_{\text{mask}}|} \\ \sum_{v_i \in \mathcal{V}_{\text{mask}}} \sum_{t=1}^{|Y_i|} \log p_{\theta}\left(y_i^{(t)} \mid y_i^{(<t)}, \hat{\mathbf{h}}_{\text{mask}}\right) \end{aligned} \quad (8)$$

where $y_i^{(t)}$ is the t-th token in Y_i. This objective trains ϕ_1, ϕ_2, and $\mathbf{h}_{\text{mask}}$ to encode graph structures into text-compatible representations that preserve attribute semantics.

4 Experiments

4.1 Experimental Setups

Basic. We leverage the Llama 3.2-1B (Touvron et al., 2023) as p_{θ}, which supports context-size up to 128k tokens. More details are elaborated in Appendix B.

Table 1: Entity classification results (AUROC, higher is better) on RELBENCH. The best and the second best values are in bold and underlined, respectively. *Rel-Zero* records the zero-shot performance of our methods, while *Rel-LLM* corresponds to the performance after fine-tuning. Standard deviations are reported in Appendix Table 7.

Dataset	Task	Split	LightGBM	RDL	ICL	ICL + MLP	Rel-Zero	Rel-LLM
rel-amazon	user-churn	Val	52.05	70.45	– –	– –	59.97	**71.85**
		Test	52.22	70.42	60.56	66.56	60.07	**71.89**
	item-churn	Val	62.39	82.39	– –	– –	64.21	**82.97**
		Test	62.54	82.81	71.96	80.16	64.10	**83.37**
rel-avito	user-visits	Val	53.31	69.65	– –	– –	58.32	**70.26**
		Test	53.05	66.20	60.28	64.98	56.17	**67.01**
	user-clicks	Val	55.63	64.73	– –	– –	58.89	**65.25**
		Test	53.60	65.90	61.32	71.31	62.28	**66.74**
rel-event	user-repeat	Val	67.76	71.25	– –	– –	61.93	**73.03**
		Test	68.04	76.89	76.38	76.72	68.12	**79.26**
	user-ignore	Val	87.96	**91.70**	– –	– –	57.47	89.95
		Test	79.93	81.62	78.55	**84.02**	61.32	83.74
rel-f1	driver-dnf	Val	68.42	71.36	– –	– –	70.35	**76.04**
		Test	68.56	72.62	65.81	**78.41**	71.84	77.15
	driver-top3	Val	67.76	77.64	– –	– –	58.57	**78.03**
		Test	73.92	75.54	**88.47**	87.36	70.64	82.22
rel-hm	user-churn	Val	56.05	70.42	– –	– –	56.83	**70.89**
		Test	55.21	69.88	64.34	68.72	55.95	**70.55**
rel-stack	user-engagement	Val	65.12	90.21	– –	– –	69.97	**90.75**
		Test	63.39	90.59	81.01	87.09	69.46	**91.21**
	user-badge	Val	65.39	89.86	– –	– –	64.47	**90.80**
		Test	63.43	88.86	71.13	88.19	62.12	**89.64**
rel-trial	study-outcome	Val	68.30	68.18	– –	– –	58.23	**70.93**
		Test	70.09	68.60	55.72	68.38	59.02	**71.04**
Average		Val	64.18	76.49	– –	– –	61.31	**77.56**
		Test	63.66	75.83	69.63	76.83	63.42	**77.82**

Dataset and Baselines. RELBENCH (Robinson et al., 2024) contains 7 datasets with 30 predictive tasks each with a rich relational structure, providing a challenging environment for developing and comparing RDL methods. Several baselines are selected, including (1) **LightGBM** learns a LightGBM (Ke et al., 2017) regressor over the raw entity features to predict the numerical targets, where only information from the single entity table is used. (2) **RDL** combines GNN predictive models with deep tabular models that extract initial entity-level representations from raw tables. (3) **ICL** (Wydmuch et al., 2024) leverages the power of LLMs' in-context learning on tabular data. Its variant **ICL + MLP** tests the strength of LLMs' representations. (4) **Entity mean/median** calculates the mean/median label value for each entity in training data and predicts the mean/median value for the entity. (5) **Global mean/median** calculates the global mean/median label value over the training data and predicts the same mean/median value across all entities. (6) **Global zero** predicts zero for all entities.

4.2 Entity Classification

Table 1 presents the results of entity classification on the RELBENCH benchmark, where AUROC is used as the evaluation metric. Our method, Rel-LLM, consistently outperforms or matches all baselines across different datasets and tasks. Notably, Rel-LLM achieves the highest average AUROC of 77.82 on the test set, surpassing RDL (75.83) and outperforming ICL + MLP (76.83) – a method that relies on an exhaustive search to determine the best combination of document generation parameters. Rel-LLM particularly excels in datasets where entity behavior follows structured patterns, such as REL-F1, where it achieves state-of-the-art performance. This can be attributed to the fact that Formula 1 drivers and their historical performance are well-documented in large-scale text corpora used for LLM pretraining, allowing the model to leverage pre-existing factual knowledge. However, in other datasets, such as REL-AMAZON and REL-EVENT, where LLMs cannot directly retrieve pretrained knowledge, the model must rely on learning from historical interactions – such as purchase

Table 2: Entity regression results (MAE ↓) on RELBENCH. The best and suboptimal values are in bold and underlined, separately. Standard deviations are reported in Appendix Table 6.

Dataset	**Task**	**Split**	**Global Zero**	**Global Mean**	**Global Median**	**Entity Mean**	**Entity Median**	**LightGBM**	**RDL**	**ICL + MLP**	**Rel-LLM**
rel-amazon	user-ltv	Val	14.141	20.740	14.141	17.685	15.978	14.141	12.132	– –	**11.773**
		Test	16.783	22.121	16.783	19.055	17.423	16.783	14.313	14.864	**14.087**
	item-ltv	Val	72.096	78.110	59.471	80.466	68.922	55.741	45.140	– –	**44.167**
		Test	77.126	81.852	64.234	78.423	66.436	60.569	50.053	52.682	**48.224**
rel-avito	ad-ctr	Val	0.048	0.048	0.040	0.044	0.044	0.037	0.037	– –	**0.033**
		Test	0.052	0.051	0.043	0.046	0.046	0.041	0.041	**0.0036**	0.037
rel-event	user-attendance	Val	0.262	0.457	0.262	0.296	0.268	0.262	0.255	– –	**0.239**
		Test	0.264	0.470	0.264	0.304	0.269	0.264	0.258	0.293	**0.251**
rel-f1	driver-position	Val	11.083	4.334	4.136	7.181	7.114	3.450	3.193	– –	**3.050**
		Test	11.926	4.513	4.399	8.501	8.519	4.170	4.022	**3.539**	3.967
rel-hm	item-sales	Val	0.086	0.142	0.086	0.117	0.086	0.086	0.065	– –	**0.060**
		Test	0.076	0.134	0.076	0.111	0.078	0.076	0.056	0.057	**0.052**
rel-stack	post-votes	Val	0.062	0.146	0.062	0.102	0.064	0.062	0.059	– –	**0.057**
		Test	0.068	0.149	0.068	0.106	0.069	0.068	0.065	0.090	**0.062**
rel-trial	study-adverse	Val	57.083	75.008	56.786	57.083	57.083	45.774	46.290	– –	**45.395**
		Test	57.930	73.781	57.533	57.930	57.930	44.011	44.473	51.845	**43.682**
	site-success	Val	0.475	0.462	0.475	0.447	0.450	0.417	0.401	– –	**0.397**
		Test	0.462	0.468	0.462	0.448	0.441	0.425	0.400	0.441	**0.397**
Average		Val	17.260	19.939	15.051	18.158	16.668	13.330	11.952	– –	**11.685**
		Test	18.299	20.393	15.985	18.325	16.801	14.045	12.631	13.761	**12.306**

behaviors or user engagement patterns. A key limitation of ICL-based methods (ICL and ICL + MLP) is their strong dependency on the quality, order, and quantity of in-context examples. Their effectiveness can be significantly constrained by the context length limits of modern LLMs (*e.g.*, 128K tokens for Llama 3.2-1B), making them sensitive to prompt design. In contrast, Rel-LLM eliminates the need for explicitly providing textual in-context examples by incorporating graph-based node embeddings into its reasoning process. This allows Rel-LLM to capture historical behaviors more efficiently and robustly, making it well-suited for modeling complex temporal and relational dependencies in entity classification tasks.

4.3 Entity Regression

Entity-level regression tasks involve predicting the numerical labels of an entity at a given seed time. We use Mean Absolute Error (MAE) as our metric and document results in Table 2. Across all datasets and tasks, Rel-LLM consistently achieves the lowest MAE (12.306), highlighting its strong generalization capability across diverse regression problems, including user behavior modeling, item ranking, and entity-based forecasting. Notably, ICL + MLP attains a higher MAE of 13.761, underscoring the challenges LLMs face in performing numerical reasoning over relational databases. In contrast, by integrating graph-based entity reasoning, Rel-LLM effectively captures long-range entity dependencies, enhancing its robustness compared to heuristic approaches and traditional machine learning models.

4.4 Ablation Studies and Discussion

Zero-shot Performance. Traditional models such as LightGBM and RDL rely on supervised training with labeled datasets. However, Rel-LLM can generalize using only its pre-existing knowledge, bypassing the need for manual annotation. Specifically, Rel-Zero achieves reasonable AUROC scores (63.42) without any labeled data across various datasets, comparable to LightGBM (63.66). For instance, on the REL-EVENT (USER-REPEAT) task, Rel-Zero achieves an AUCROC of 68.12, which is already close to fine-tuned models like RDL (76.89). On REL-STACK (USER-ENGAGEMENT), Rel-Zero achieves an AUCROC of 69.46, which is quite high compared to LightGBM (63.39) and is competitive with other baselines. This adaptability makes Rel-LLM highly valuable for real-world applications where labeled data is scarce or expensive to obtain. Nevertheless, there are cases where Rel-Zero struggles more, such as REL-EVENT (USER-IGNORE), where it achieves 61.32, which is far behind RDL (81.62), and the fine-tuned Rel-LLM (83.74). Some failure exploration over many-shot in-context learning for Rel-Zero can be found in Appendix C.2.

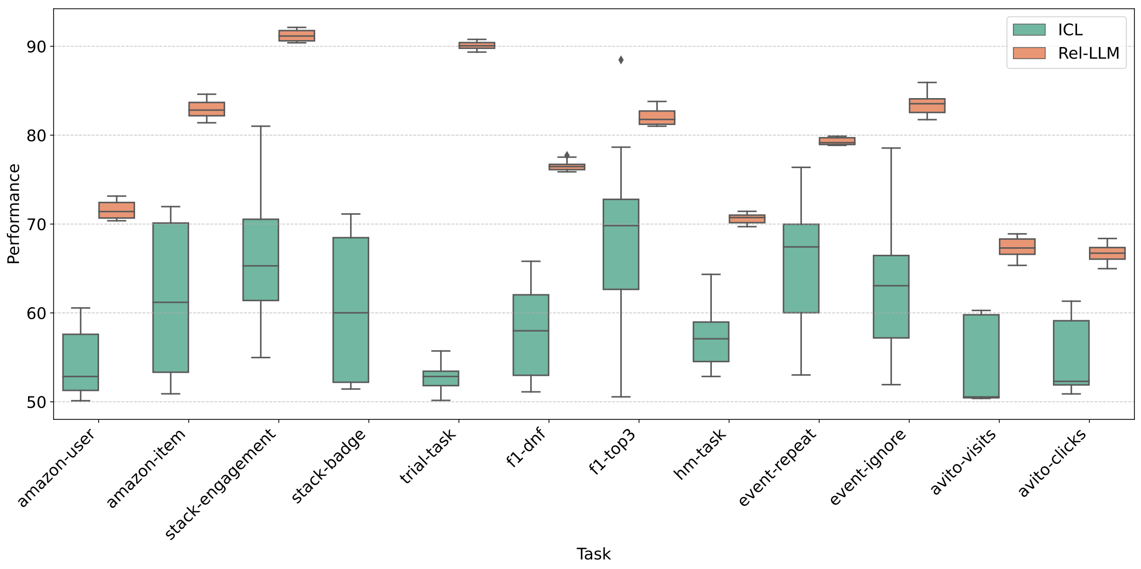


Figure 2: Comparision of model prediction variation across different classification tasks.

Robustness to In-Context Examples. While ICL offers a convenient way to forecast an entity's future behavior without end-to-end training, its predictions are highly variable and sensitive to the order and number of demonstrations. Figure 2 illustrates the fluctuations in ICL's performance across different document parameter settings, such as the number of in-context and related examples. The results indicate that ICL is notably unstable under varying document generation conditions, consistent with prior findings on ICL's unpredictability (Lu et al., 2021; Min et al., 2022; Garg et al., 2022). Furthermore, additional factors, such as the permutation of examples, remain unexplored. In contrast, Rel-LLM demonstrates strong robustness to the randomness of subgraph sampling and variations in in-context examples (see Appendix C.1), making it a more reliable and trustworthy choice for real-world applications.

Effect of LLM Model Size. We investigate the impact of scaling up the model size of Rel-LLM to understand whether larger models yield substantial performance improvements. Due to computational resource limitations, we compare the performance of the 1B (billion) parameter version of Rel-LLM with its 3B counterpart, specifically Llama-3.2-3B. Figure 3 illustrates the comparison, revealing that the 3B model generally outperforms the 1B model across most tasks. However, the performance gap between the two models is relatively modest, suggesting that while increasing model size provides benefits, the returns may diminish at this scale. Notably, the 3B model demonstrates a more pro-

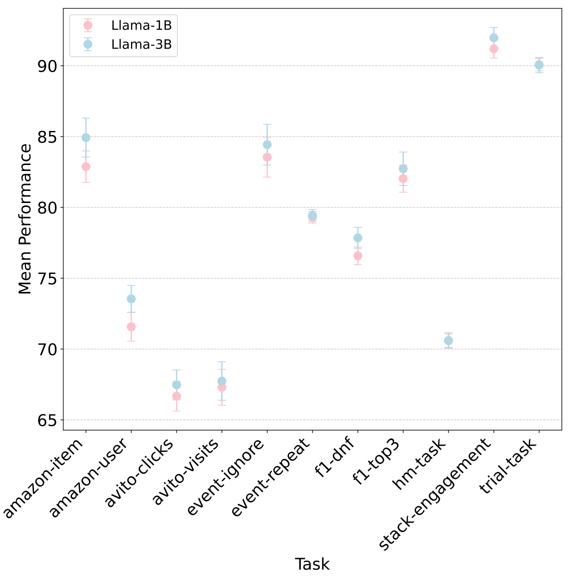


Figure 3: Ablation on LLMs' model size.

nounced improvement in the REL-F1 task, where its larger capacity allows it to better leverage pretraining knowledge, particularly in scenarios requiring complex reasoning or deeper contextual understanding. This indicates that model size plays a critical role in tasks where knowledge utilization and generalization are paramount, though the extent of improvement may vary depending on the specific task requirements.

5 Related works

RDL. Relational learning seeks to capture inter-entity relationships (Struyf and Blockeel, 2010), with early methods like inductive logic programming (De Raedt, 2008), probabilistic logic (De Raedt and Kersting, 2008), and relational RL (Zambaldi et al., 2018) limited by scalability. RDL (Chen et al., 2024b) overcomes this by combining deep learning's expressiveness with relational reasoning to model non-Euclidean structures, unlike conventional DL optimized for grid-like data (LeCun et al., 2015). Earlier approaches (Nickel et al., 2011; Perozzi et al., 2014) often neglected dependencies, prompting the rise of GNNs (Kipf and Welling, 2016; Chen et al., 2021; Wu et al., 2023, 2024). RDL is now key to domains such as social networks (Hamilton et al., 2017), recommender systems (Ying et al., 2018), knowledge graphs (Nickel et al., 2016), drug discovery (Rozemberczki et al., 2021), fraud detection (Ma et al., 2021), and supply chains (Wasi et al., 2024).

LLMs for Tabular Data. Breakthroughs in LLMs facilitate rigorous exploration of tabular data modeling (Fang et al., 2024), advancing tasks such as prediction (Hegselmann et al., 2023; Wang et al., 2023), tabular data synthesis (Gulati and Roysdon, 2024), question answering (Ye et al., 2023), and table understanding (Sui et al., 2023). To effectively adapt LLMs for tabular data, researchers have employed techniques focusing on key components like data retrieval (Padhi et al., 2021; Wang and Sun, 2022), serialization (Iida et al., 2021; Sui et al., 2024), table manipulations (Liu et al., 2023a), prompt engineering (Chen, 2022), self-supervised learning (Yang et al., 2023; Iida et al., 2021), and end-to-end agentic workflows (Abraham et al., 2022). Despite these strides, challenges remain. Key areas for improvement include reducing bias in tabular data interpretation (Liu et al., 2023b), mitigating hallucinations in model outputs (Akhtar et al., 2023), enhancing interpretability (Hegselmann et al., 2023), and achieving greater computational efficiency (Ruan et al., 2024). Furthermore, there has been limited progress in addressing the complexities of relational data within more intricate tabular structures.

RAG on Tables. RAG (Gao et al., 2023; Chen et al., 2024a) enhances LLMs with contextual grounding via data preparation, indexing, retrieval, and augmentation (Gupta et al., 2024). In large tables, it reduces processing load using strategies like random sampling (TAPEX (Liu et al., 2021)), sliding windows (TabBert (Padhi et al., 2021)), row partitioning (TUTA (Wang et al., 2021)), truncation (TABBIE (Iida et al., 2021),(Qin et al., 2022)), and dimension thresholds (TAGOP(Zhu et al., 2021)). More recent efforts optimize retrieval relevance, with cTBLS (Sundar and Heck, 2023) using dense dual encoders and DAMO-ConvAI (Ye et al., 2023) leveraging prompt-based row/column selection.

6 Conclusion

We introduced Rel-LLM, a framework combining GNNs and LLMs to enhance relational database processing. By integrating graph prompt tuning, temporal-aware sampling, and self-supervised pretraining, Rel-LLM effectively captures relational semantics while maintaining efficiency. Experiments show noteworthy improvements in accuracy, reasoning, and consistency over baselines, demonstrating its potential for scalable, structure-aware LLM applications. Rel-LLM advances the integration of structured and unstructured data, paving the way for more robust AI systems.

7 Limitations and Future Works

Despite the progress, there is still plenty of room to push the frontier of LLMs for RDL. For instance, it is interesting to investigate a delicate modification of the cross-attention mechanism to better incorporate the graph prompt and LLMs' textual embeddings. Additionally, more advanced and larger LLM architectures, such as DeepSeek-R1 (Guo et al., 2025), can be leveraged to improve prompt accuracy due to their stronger pre-existing knowledge and reasoning capacity. Furthermore, exploring how Rel-LLM can be applied to other structured data types, such as knowledge graphs, biomedical datasets, and financial systems, can broaden its impact.

References

Abhijith Neil Abraham, Fariz Rahman, and Damanpreet Kaur. 2022. Tablequery: Querying tabular data with natural language. *arXiv preprint arXiv:2202.00454*.

Josh Achiam and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, and 1 others. 2025. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966.

Mubashara Akhtar, Abhilash Shankarampeta, Vivek Gupta, Arpit Patil, Oana Cocarascu, and Elena Simperl. 2023. Exploring the numerical reasoning capabilities of language models: A comprehensive analysis on tabular data. *arXiv preprint arXiv:2311.02216*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024a. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Wenhu Chen. 2022. Large language models are few (1)-shot table reasoners. *arXiv preprint arXiv:2210.06710*.

Yang Chen, Cong Fang, Zhouchen Lin, and Bing Liu. 2024b. Relational learning in pre-trained models: A theory from hypergraph recovery perspective. *arXiv preprint arXiv:2406.11249*.

Yuzhou Chen, Baris Coskunuzer, and Yulia Gel. 2021. Topological relational learning on graphs. *Advances in neural information processing systems*, 34:27029–27042.

Zhaoling Chen, Xiangru Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. 2025. Locagent: Graph-guided llm agents for code localization. *arXiv preprint arXiv:2503.09089*.

Luc De Raedt. 2008. *Logical and relational learning*. Springer Science & Business Media.

Luc De Raedt and Kristian Kersting. 2008. Probabilistic inductive logic programming. In *Probabilistic inductive logic programming: theory and applications*, pages 1–27. Springer.

Sašo Džeroski. 2010. *Relational data mining*. Springer.

Gus Eggert, Kevin Huo, Mike Biven, and Justin Waugh. 2023. Tablib: A dataset of 627m tables with context. *arXiv preprint arXiv:2310.07875*.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Jane Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, Christos Faloutsos, and 1 others. 2024. Large language models (llms) on tabular data: Prediction, generation, and understanding-a survey.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. 2023. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. 2024. Position: Relational deep learning-graph representation learning on relational databases. In *Forty-first International Conference on Machine Learning*.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. 2024. Large scale transfer learning for tabular data via language modeling. *arXiv preprint arXiv:2406.12031*.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

Manbir Gulati and Paul Roysdon. 2024. Tabmt: Generating tabular data with masked transformers. *Advances in Neural Information Processing Systems*, 36.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions. *arXiv preprint arXiv:2410.12837*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. Tabllm: Few-shot classification of tabular data with large language models. In *International*

Conference on Artificial Intelligence and Statistics, pages 5549–5581. PMLR.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer.

Weihua Hu, Yiwen Yuan, Zecheng Zhang, Akihiro Nitta, Kaidi Cao, Vid Kocijan, Jure Leskovec, and Matthias Fey. 2024. Pytorch frame: A modular framework for multi-modal tabular learning. *arXiv preprint arXiv:2404.00776*.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584*.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Weichen Li, Xiaotong Huang, Jianwu Zheng, Zheng Wang, Chaokun Wang, Li Pan, and Jianhua Li. 2024. rllm: Relational table learning with llms. *arXiv preprint arXiv:2407.20157*.

T Lin. 2017. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Tianyang Liu, Fei Wang, and Muhao Chen. 2023a. Rethinking tabular data understanding with large language models. *arXiv preprint arXiv:2312.16702*.

Yanchen Liu, Srishti Gautam, Jiaqi Ma, and Himabindu Lakkaraju. 2023b. Investigating the fairness of large language models for predictions on tabular data. *arXiv preprint arXiv:2310.14607*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12012–12038.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Jan Motl and Oliver Schulte. 2015. The ctu prague relational learning repository. *arXiv preprint arXiv:1511.03086*.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, and 1 others. 2011. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584.

Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. 2021. Tabular transformers for modeling multivariate time series. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3565–3569. IEEE.

Jane Pan. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. Master's thesis, Princeton University.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca

Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Can Qin, Sungchul Kim, Handong Zhao, Tong Yu, Ryan A Rossi, and Yun Fu. 2022. External knowledge infusion for tabular pre-training models with dual-adapters. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1401–1409.

Zongyue Qin, Chen Luo, Zhengyang Wang, Haoming Jiang, and Yizhou Sun. 2024. Relational database augmented large language model. *arXiv preprint arXiv:2407.15071*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E Lenssen, Yiwen Yuan, Zecheng Zhang, and 1 others. 2024. Relbench: A benchmark for deep learning on relational databases. *arXiv preprint arXiv:2407.20060*.

Benedek Rozemberczki, Stephen Bonner, Andriy Nikolov, Michael Ughetto, Sebastian Nilsson, and Eliseo Papa. 2021. A unified view of relational deep learning for drug pair scoring. *arXiv preprint arXiv:2111.02916*.

Yucheng Ruan, Xiang Lan, Jingying Ma, Yizhi Dong, Kai He, and Mengling Feng. 2024. Language modeling on tabular data: A survey of foundations, techniques and evolution. *arXiv preprint arXiv:2408.10548*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. 2023. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. *arXiv preprint arXiv:2310.10358*.

Jan Struyf and Hendrik Blockeel. 2010. Relational learning.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2023. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. *arXiv preprint arXiv:2312.09039*.

Anirudh S Sundar and Larry Heck. 2023. ctbls: Augmenting large language models with conversational tables. *arXiv preprint arXiv:2303.12024*.

Xiangru Tang, Daniel Shao, Jiwoong Sohn, Jiapeng Chen, Jiayi Zhang, Jinyu Xiang, Fang Wu, Yilun Zhao, Chenglin Wu, Wenqi Shi, and 1 others. 2025. Medagentsbench: Benchmarking thinking models and agent frameworks for complex medical reasoning. *arXiv preprint arXiv:2503.07459*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ruiyu Wang, Zifeng Wang, and Jimeng Sun. 2023. Unipredict: Large language models are universal tabular predictors. *arXiv preprint arXiv:2310.03266*.

Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. 2024. Beyond the limits: A survey of techniques to extend the context length in large language models. *arXiv preprint arXiv:2402.02244*.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Zifeng Wang and Jimeng Sun. 2022. Transtab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35:2902–2915.

Azmine Toushik Wasi, MD Islam, Adipto Raihan Akib, and Mahathir Mohammad Bappy. 2024. Graph neural networks in supply chain analytics and optimization: Concepts, perspectives, dataset and benchmarks. *arXiv preprint arXiv:2411.08550*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yotam Wolf, Noam Wies, Oshri Avnery, Yoav Levine, and Amnon Shashua. 2023. Fundamental limitations of alignment in large language models. *arXiv preprint arXiv:2304.11082*.

Fang Wu, Siyuan Li, Xurui Jin, Yinghui Jiang, Dragomir Radev, Zhangming Niu, and Stan Z Li. 2023. Rethinking explaining graph neural networks via nonparametric subgraph matching. In *International conference on machine learning*, pages 37511–37523. PMLR.

Fang Wu, Siyuan Li, and Stan Z Li. 2024. Discovering the representation bottleneck of graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*.

Fang Wu and Stan Li. 2024. Insertgnn: A hierarchical graph neural network for the toefl sentence insertion problem. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 173–180.

Marek Wydmuch, Łukasz Borchmann, and Filip Graliński. 2024. Tackling prediction tasks in relational databases with llms. *arXiv preprint arXiv:2411.11829*.

Zerui Xu, Fang Wu, Yuanyuan Zhang, and Yue Zhao. 2024. Retrieval-reasoning large language model-based synthetic clinical trial generation. *arXiv preprint arXiv:2410.12476*.

Yazheng Yang, Yuqi Wang, Guang Liu, Ledell Wu, and Qi Liu. 2023. Unitabe: Pretraining a unified tabular encoder for heterogeneous tabular data. *arXiv preprint arXiv:2307.09249*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983.

Lukáš Zahradník, Jan Neumann, and Gustav Šír. 2023. A deep learning blueprint for relational databases. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, and 1 others. 2018. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*.

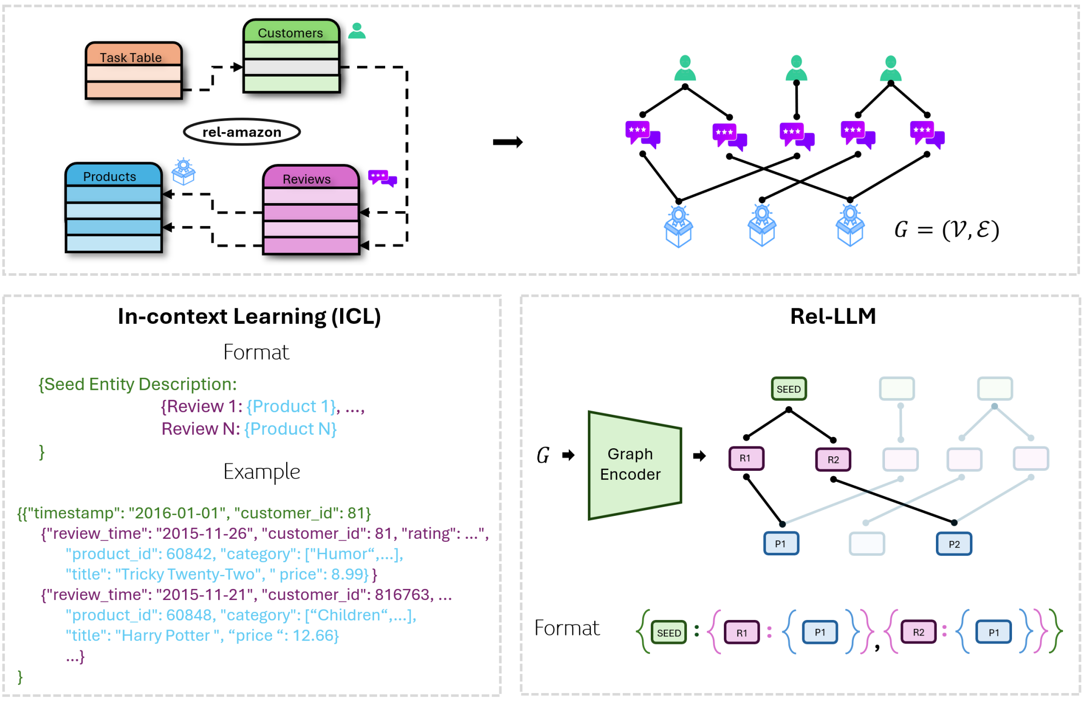
Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*.

A Prompt Construction Comparison

The Figure 4 shows the difference between ordinary ICL and Rel-LLM to formulate the prompts. ICL requires the serialization of tabular data, where great redundancy can be introduced and the document length can be extremely long. As a remedy, Rel-LLM constructs its graph prompt using the graph node embeddings, which are extracted using a graph encoder.



```
{Seed Entity Description:
                {Review 1: {Product 1}, ...,
                Review N: {Product N}
}
```

```
{{"timestamp": "2016-01-01", "customer_id": 81}
   {"review_time": "2015-11-26", "customer_id": 81, "rating": ...",
      "product_id": 60842, "category": ["Humor“,...],
      "title": "Tricky Twenty-Two", " price": 8.99} }
   {"review_time": "2015-11-21", "customer_id": 816763, ...
      "product_id": 60848, "category": [“Children“,...],
      "title": "Harry Potter ", “price “: 12.66}
      ...}
}
```

Figure 4: Visualization of different algorithms to compose the task-specific prompts.

B Experimental Details

All experiments are conducted on 4 NVIDIA A100 GPUs with 80G memory, mainly relying on Pytorch (Paszke et al., 2019), PyG (Fey and Lenssen, 2019), and Huggingface. Pretraining is conducted once. After that, each fine-tuning result is replicated three times, utilizing different random seeds for each run to ensure robustness and reproducibility. The mean results are reported in the main text and the prediction variance is put in the Appendix.

B.1 Answer Generation Details

During pretraining, we leverage plain text generation as the output strategy as the column values of different entities can be of diverse data types. For classification tasks, we exploit the token distribution output strategy, where we set the *maximum_new_tokens* to 1mainly focus on the token of "YES" and "NO". We rely on the MLP transformation output strategy for the regression tasks to ensure that the predictions always have float numbers.

Notably, we do not include zero-shot performance results for Rel-LLM on regression tasks. This is because, after pretraining, LLMs are not inherently designed to generate numerical outputs, such as floating-point numbers, in a structured manner when using a plain text output strategy. Instead, their responses are primarily optimized for generating natural language text. This limitation becomes more pronounced in smaller LLMs, such as Llama 3.2-1B, which struggle to follow explicit instructions in prompts, even when explicitly asked to produce floating-point numbers. As a result, these models often fail to provide reliable numerical outputs for regression tasks, making their zero-shot performance difficult to assess meaningfully.

B.2 Task Context Details

Below we elaborate on how we formulate the dataset description x_{task} and the question prompt x_{quest}. We have explored different prompt templates but observed little performance difference.

Dataset	**Task**	**Description Prompt**
rel-event	user-attendance	This task is to predict how many events each user will respond yes or maybe in the next seven days.
rel-event	user-ignore	This task is to predict whether a user will ignore more than 2 event invitations in the next 7 days.
rel-event	user-repeat	This task is to predict whether a user will attend an event (by responding yes or maybe) in the next 7 days if they have already attended an event in the last 14 days.
rel-amazon	user-churn	This task is to predict if the customer will review any product in the next 3 months or not.
rel-amazon	item-churn	This task is to predict if the product will receive any reviews in the next 3 months or not.
rel-amazon	user-ltv	This task is to predict the $ value of the total number of products each user will buy and review in the next 3 months.
rel-amazon	item-ltv	This task is to predict the $ value of the total number of purchases and reviews each product will receive in the next 3 months.
rel-stack	post-votes	This task is to predict how many votes this user's post will receive in the next 3 months.
rel-stack	user-engagement	This task is to predict if a user will make any votes, posts, or comments in the next 3 months or not.
rel-stack	user-badge	This task is to predict if a user will receive a new badge in the next 3 months or not.
rel-avito	user-visits	This task is to predict whether this customer will visit more than one Ad in the next 4 days or not.
rel-avito	user-clicks	This task is to predict whether this customer will click on more than one Ads in the next 4 days or not.
rel-avito	ad-ctr	Assuming the Ad will be clicked in the next 4 days, this task is to predict the Click-Through-Rate (CTR) for each Ad.
rel-f1	driver-position	This task is to predict the average finishing position of each driver across all races in the next 2 months.
rel-f1	driver-dnf	This task is to predict if this driver will finish a race in the next 1 month or not.
rel-f1	driver-top3	This task is to predict if this driver will qualify in the top-3 for a race in the next 1 month or not.
rel-trial	study-outcome	This task is to predict if the trial in the next 1 year will achieve its primary outcome or not.
rel-trial	study-adverse	This task is to predict the number of affected patients with severe adverse events/deaths for the trial in the next 1 year.
rel-trial	site-success	This task is to predict the success rate of a trial site in the next 1 year.
rel-hm	item-sales	This task is to predict the total sales for an article in the next week.
rel-hm	user-churn	This task is to predict the churn for a customer (no transactions) in the next week.

Dataset	**Task**	**Question Prompt**
rel-event	user-attendance	What is the attendance of user? Give an integer as an answer.
rel-event	user-ignore	Given recent activity and event history, will this user ignore more than 2 event invitations in the next 7 days? Give Yes or No as an answer.
rel-event	user-repeat	Given recent activity and event history, will this user attend an event in the next 7 days? Give Yes or No as an answer.
rel-amazon	user-churn	Based on the customer data provided, will this customer review any product in the next 3 months? Give Yes or No as an answer.
rel-amazon	item-churn	Based on the product data provided, will the product receive any reviews in the next 3 months? Give Yes or No as an answer.
rel-amazon	user-ltv	What is the total dollar value of products this user will buy and review in the next 3 months? Provide a float numerical answer.
rel-amazon	item-ltv	What is the total dollar value of purchases this product will receive in the next 3 months? Provide a float numerical answer.
rel-stack	post-votes	Based on records of activity, how many votes will this user's post receive in the next 3 months? Give an integer as an answer.
rel-stack	user-engagement	Based on records of activity, will this user make any votes, posts, or comments in the next 3 months? Give Yes or No as an answer.
rel-stack	user-badge	Based on records of activity, will this user receive a new badge in the next 3 months? Give Yes or No as an answer.
rel-avito	user-visits	Will this customer visit more than one Ad in the next 4 days? Give Yes or No as an answer.
rel-avito	user-clicks	Will this customer click on more than one Ads in the next 4 days? Give Yes or No as an answer.

Dataset	**Task**	**Question Prompt**
rel-avito	ad-ctr	What is the Click-Through-Rate (CTR) for this Ad?
rel-f1	driver-position	What is the average finishing position of this driver across all races in the next 2 months? Provide a float numerical answer.
rel-f1	driver-dnf	Will this driver finish a race in the next 1 month? Give Yes or No as an answer.
rel-f1	driver-top3	Will this driver qualify in the top-3 for a race in the next 1 month? Give Yes or No as an answer.

B.3 Hyperparameyter Space

The hidden dimension of Llama d_l is 2048. If a weight decay is applied, then we use the AdamW (Loshchilov, 2017) as the optimizer. Otherwise, we leverage Adam (Kingma, 2014) as the optimizer. A Plateau learning rate scheduler is involved with a patience of 100 and a factor of 0.8. As for the other important hyperparameters, we use a random search mechanism to find the optimal combination. The entire hyperparameter search space is depicted in Table 5. The best hyperparameter is selected based on the supervised learning performance on the validation set without pretraining. Then we implement the pretraining and fine-tuning with this fixed set of hyperparameters.

Table 5: Hyperparameters setup for Rel-LLM.

Hyperparameters Search Space	Symbol	Value
Training Setup		
Epochs	–	[10, 20, 50, 100]
Batch size	–	[16, 32, 128, 256, 512]
Validation steps	–	[20, 200, 1000, 2000]
Learning rate	–	[1e-4, 5e-5, 1e-6]
Weight decay	–	[0, 1.5e-4]
Temporal Strategy	–	[Uniform, Last]
Focusing parameter for focal loss	γ	2.0
Class balance weight for focal coss	α_t	[0.1, 0.2, 0.4, 0.8, 0.9]
GNN Architecture		
Dropout rate	–	[0.1, 0.2, 0.3, 0.4, 0.6, 0.8]
Number of GNN layers	L	[2]
Text Embedder	–	[Glove, MPNet]
The output dimension of graph encoder	d_g	[128]
Graph Prompt		
Number of in-context examples	n_{inc}	[0, 1]
Number of nested entities	n_{nest}	[0, 128]
Recursion depth	ζ	[0, 1]
Pretraining Technique		
Pretraining epochs	–	[10, 50, 100]
Ratio of masked entities	$\frac{\lvert\mathcal{V}_{\text{mask}}\rvert}{\lvert\mathcal{V}\rvert}$	[0.5]
Attribute permutation	π	[True, False]
LLM Architecture		
Fine-tuning strategy	–	[Freeze, LoRA]

B.4 GNN Implementation Details

Here we provide more details of graph encoder architecture. Message-passing Graph Neural Networks (MP-GNNs) (Gilmer et al., 2017) are a generic computational framework to define DL architectures on graph-structured data. Given a heterogeneous graph $G = (\mathcal{V}, \mathcal{E}, \phi, \psi)$ with initial node embeddings $\left\{\mathbf{h}_v^{(0)}\right\}_{v\in\mathcal{V}}$, a single message passing iteration computes updated features $\left\{\mathbf{h}_v^{(i+1)}\right\}_{v\in\mathcal{V}}$ from features

$\left\{\mathbf{h}_v^{(i)}\right\}_{v \in \mathcal{V}}$ given by the previous iteration. One iteration takes the form:

$$\mathbf{h}_v^{(i+1)} = f\left(\mathbf{h}_v^{(i)}, \left\{\left\{g\left(\mathbf{h}_w^{(i)}\right) \mid w \in \mathcal{N}(v)\right\}\right\}\right) \tag{9}$$

where f and g are arbitrary differentiable functions with optimizable parameters and $\{\{\cdot\}\}$ a permutation invariant set aggregator, such as mean, max, sum, or a combination. Heterogeneous message passing (Schlichtkrull et al., 2018; Hu et al., 2020)is a nested version of Equation 9, adding an aggregation over all incoming edge types to learn distinct message types:

$$\mathbf{h}_v^{(i+1)} = f_{\phi(v)}\left(\mathbf{h}_v^{(i)}, \left\{\left\{f_R\left(\left\{\left\{g_R\left(\mathbf{h}_w^{(i)}\right) \mid w \in \mathcal{N}_R(v)\right\}\right\}\right) \mid \forall R = (T, \phi(v)) \in \mathcal{R}\right\}\right\}\right) \tag{10}$$

where $\mathcal{N}_R(v) = \{w \in \mathcal{V} \mid (w, v) \in \mathcal{E} \text{ and } \psi(w, v) = R\}$ denotes the R-specific neighborhood of node $v \in \mathcal{V}$. This formulation supports a wide range of different graph neural network operators, which define the specific form of functions $f_{\phi(v)}, f_R, g_R$ and $\{\{\cdot\}\}$.

Given a relational entity graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$ with attached mapping functions ψ, ϕ, τ and initial node embeddings $\left\{\mathbf{h}_v^{(0)}\right\}_{v \in \mathcal{V}}$ and an example specific seed time $t \in \mathbb{R}$, we obtain a set of deep node embeddings $\left\{\mathbf{h}_v^{(L)}\right\}_{v \in \mathcal{V}}$ by L consecutive applications, where we additionally filter R-specific neighborhoods based on their timestamp, *i.e.*, replace $\mathcal{N}_R(v)$ with

$$\mathcal{N}_{\bar{\bar{R}}}^{\leq t^*}(v) = \{w \in \mathcal{V} \mid (w, v) \in \mathcal{E}, \psi(w, v) = R, \text{ and } \tau(w) \leq t^*\}, \tag{11}$$

which can be realized by the temporal sampling procedure. The formulation naturally respects time by only aggregating messages from nodes that were available before the given seed time t^*. The given formulation is agnostic to specific implementations of message passing and supports a wide range of different operators.

B.5 Class Imbalance Training

We notice a severe class imbalance in some classification sub-tasks. For instance, in DRIVER-DNF of REF-F1, the ratios of positives and negatives are 11.96%/88.04%, 22.08%/77.92%, and 29.49%/70.51% for train, validation, and test, respectively. While in USER-ENGAGEMENT of REL-STACK, the ratios of positives and negatives are 5.0%/95.0%, 2.81%/97.19%, and 2.74%/97.26% for train, validation, and test, respectively. In USER-CLICKS of REL-AVITO, the ratios of positives and negatives are 3.87%/96.13%, 3.52%/96.48%, and 1.54%/98.46% for train, validation, and test, respectively. Due to the over-parameterization of LLMs on these tabular data, LLMs usually attain a very high accuracy but a pretty low AUC-ROC. To avoid that, we employ the classic focal loss (Lin, 2017), which can be written as:

$$\mathrm{FL}\left(p_\theta\right) = -\alpha_t \left(1 - p_\theta\right)^\gamma \log\left(p_\theta\right), \tag{12}$$

where p_θ is the predicted probability of the true (positive) class. α_t is the weighting factor and γ is a focusing parameter that adjusts the rate at which easy examples are down-weighted.

C Unsuccessful Attempts

In the early stage of developing Rel-LLM, we encounter failures and setbacks along the way. We provide our failure experiences here to provide insights and explanations.

C.1 Few-shot In-Context Examples

To facilitate in-context learning, we include n_{inc} labeled examples, denoted as x_{context}. Each example consists of an entity v_i from the training set and its corresponding label y_i. We enforce a temporal constraint $t_{v_i} < t^*$ for all v_i to maintain chronological validity. Stratified sampling ensures balanced positive and negative examples in binary classification settings where $y_i \in 0, 1$. These examples $\{v_i, y_i\}_{i=1}^{n_{\text{inc}}}$ remain consistent across all documents in a given task to enhance model stability.

However, we observe that Rel-LLM does not benefit from the inclusion of in-context examples. To elucidate this phenomenon, we refer to the framework proposed by Pan (2023), which decomposes the effect of in-context learning into two distinct roles: task recognition (TR) and task learning (TL). Their findings indicate that TR does not improve with increasing model size or additional demonstrations, whereas TL is acquired as model capacity scales. In our case, given that Llama 3.2-1B is relatively small, the provided demonstrations primarily contribute to TR rather than TL. However, following fine-tuning and the explicit inclusion of the task description x_{task} and question prompt x_{quest} , Rel-LLM has already fully internalized the task. Consequently, the addition of in-context examples has minimal impact. Nonetheless, this conclusion may not hold when employing significantly larger LLMs.

C.2 Many-shot In-context Learning with Graph Neural Prompt

Recent studies (Agarwal et al., 2025) have shown that LLMs tend to benefit from an increased number of in-context examples, leading to better generalization and task adaptation. However, a major limitation of conventional in-context learning arises when these examples are represented as raw text, as the total context length grows rapidly with the number of demonstrations. This issue becomes particularly problematic given the quadratic computational complexity of self-attention in transformer-based models, which can make many-shot in-context learning infeasible in practice. To mitigate this problem and investigate the effectiveness of many-shot in-context learning in a more scalable manner, we propose replacing textual in-context examples with graph neural prompts (see Figure 5). Instead of encoding each example as a sequence of tokens, we represent each demonstration input as a corresponding node embedding. This embedding is then paired with the ground truth label, significantly reducing the number of tokens required to store in-context examples. By leveraging this approach, we can incorporate a much larger number of examples into the model's context window, facilitating extensive in-context learning.

During the pretraining phase, the input context is structured as:

$$< \text{SeedNodeEmbedding} > + < \text{DatasetDescription} > + < \text{TaskDescription} >. \tag{13}$$

After pretraining, inference is performed using the following prompt template:

$$\begin{aligned}
&< \text{SeedNodeEmebedding} > + < \text{DatasetDescription} > + < \text{TaskDesription} > \\
&+ < \text{NodeEmbeddingofExample1} > + < \text{LabelofExample1} > + ... \\
&+ < \text{NodeEmbeddingofExampleN} > + < \text{LabelofExampleN} > + < \text{SeedNodeEmebedding} >.
\end{aligned} \tag{14}$$

Furthermore, to ensure a balanced representation across different classes, we consider a many-shot setting where the number of in-context examples per class is approximately equal. Despite these improvements in memory efficiency and scalability, we observe an unexpected decline in the classification performance of Rel-Zero across all tasks when adopting many-shot in-context learning. Specifically, when comparing few-shot settings ($N \leq 16$) to many-shot settings ($N > 16$), we find no significant improvements, contradicting previous findings that suggest LLMs benefit from a higher number of demonstrations.

Upon further investigation, we identify that the model's predictions are highly sensitive to the choice of in-context examples. The performance varies considerably depending on the specific selection of examples, with different random seeds leading to substantial fluctuations. This instability aligns with observations from prior studies (Wydmuch et al., 2024), which report that the order and composition of in-context demonstrations can disproportionately influence LLMs. We hypothesize that this sensitivity may stem from the model's inability to effectively aggregate information across many node embeddings, potentially due to limitations in how LLMs process structured representations compared to textual descriptions. Further research is required to disentangle whether this issue arises from suboptimal prompt construction, insufficient adaptation of GNN-derived embeddings, or inherent limitations in LLMs' ability to generalize from structured input representations.

C.3 Cell-level Masking

Beyond masking entire entities during pretraining, we also explore a more granular approach by selectively masking only certain attributes while keeping others accessible to the large language models (LLMs).

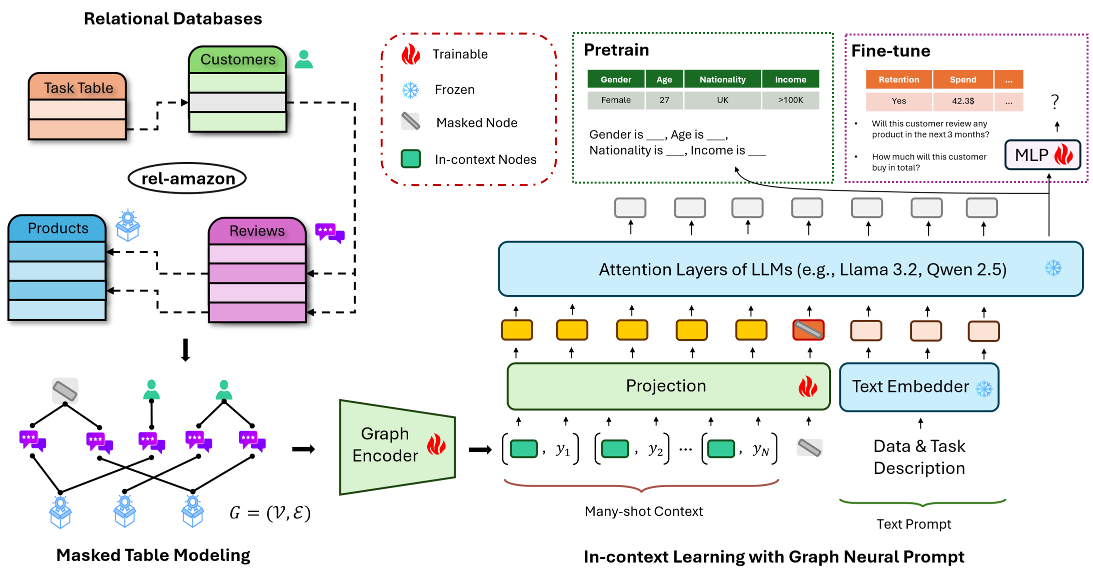


Figure 5: Illustration of many-shot ICL with graph neural prompt.

This alternative, referred to as cell-level masking, contrasts with the standard row-level masking strategy (see Figure 6). However, empirical results indicate a notable decline in performance under this approach. Specifically, the AUC-ROC of REL-F1 drops from 71.74 to 58.68, while the AUC-ROC of REL-AMAZON decreases from 64.10 to 55.24.

We hypothesize that this performance degradation stems from the inherent dependencies between different attributes within an entity. When only a subset of attributes is masked while others remain visible, the LLMs can often reconstruct the missing values by leveraging correlations across known attributes. This unintended leakage reduces the need for deeper structural reasoning and instead encourages naive inference based on column-wise dependencies. In contrast, entity-level masking disrupts entire relational structures, compelling the model to develop a more holistic understanding of relational graphs rather than relying on shallow pattern matching. Thus, our findings suggest that entity-level masking poses a greater challenge for LLMs, fostering a more robust representation of relational structures.

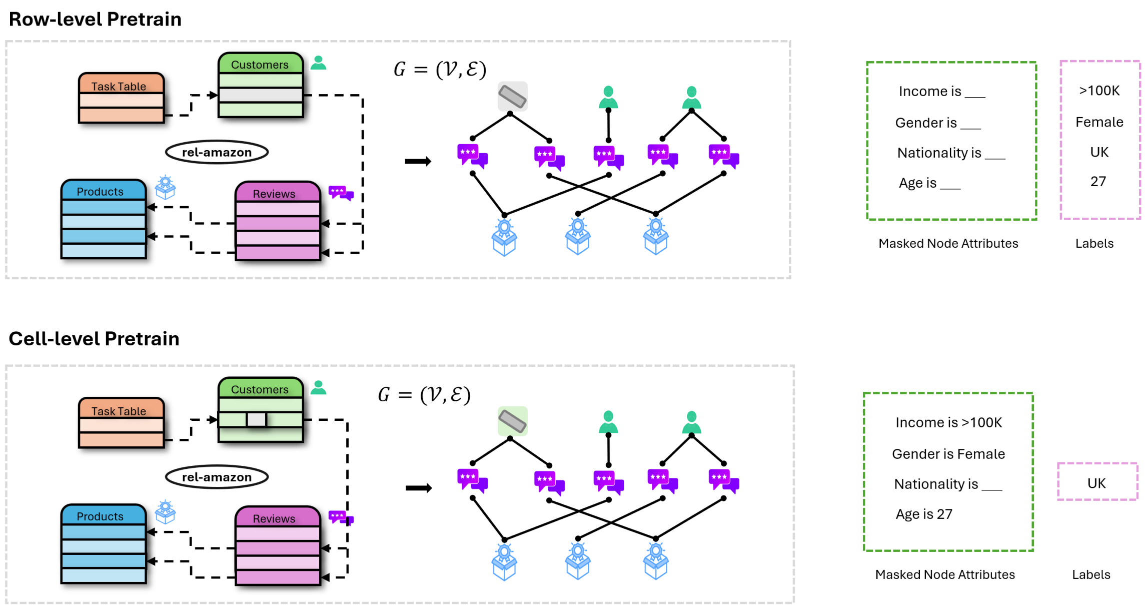


Figure 6: Difference between entity-level and cell-level masking strategies for pretraining.

D Additional Results

Here, we provide the standard deviations over repeated runs for both the entity classification and regression tasks, where the variance is slight and acceptable.

Table 6: Standard deviation of performance on the entity regression tasks.

Dataset	rel-amazon		rel-avito	rel-event	rel-f1	rel-hm	rel-stack	rel-trial	
Task	user-ltv	item-ltv	ad-ctr	user-attendance	driver-position	item-sales	post-votes	study-adverse	site-success
val	0.009	0.083	0.000	0.009	0.025	0.001	0.002	0.389	0.018
test	0.014	0.244	0.001	0.008	0.132	0.002	0.002	0.274	0.026

Table 7: Standard deviation of performance on the entity classification tasks.

Dataset	rel-amazon		rel-avito		rel-event		rel-f1	
Task	user-churn	item-churn	user-visits	user-clicks	user-repeat	user-ignore	driver-position	item-sales
val	0.012	0.08	0.09	0.47	3.08	0.42	2.09	4.77
test	0.012	0.10	0.15	3.51	2.76	1.86	0.94	2.85

Dataset	rel-hm	rel-stack		rel-trial
Task	user-churn	user-engagement	user-bagde	study-outcome
val	0.017	0.16	0.14	0.96
test	0.23	0.15	0.14	2.47