SEMANTICALLY ANCHORED PROMPTS FOR RELATIONAL DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work injects GNN embeddings of relational databases into frozen large language models (LLMs) as soft prompts through a learned projection. We show that such prompts are *semantically blind*: the projection places them off the manifold of pre-trained token embeddings, so the LLM cannot tell a customer from a transaction and its world knowledge is never activated. We propose **RelAnchor**, which learns an assignment instead of a projection. *Schema Anchoring* embeds the database's table and column descriptions with the frozen input embedding layer, yielding an on-manifold anchor basis that spans exactly the semantics of the database. *Adaptive Composition* expresses each row as a row-specific convex mixture over these anchors, gated with its graph representation, so that every prompt preserves relational distinctions while staying anchored to named schema concepts near the manifold. We prove that anchoring costs at most $O(1/\sqrt{k})$ in prompt fidelity with k active anchors, governed by the LLM's Lipschitz constant on the compact on-manifold anchor hull rather than the unbounded ambient space. On RelBench, RelAnchor outperforms GNN-only baselines and the state-of-the-art graph-prompted LLM on the aggregate classification and regression metrics.[1]

1 INTRODUCTION

Relational databases hold the majority of enterprise data, organizing information across multiple tables connected through primary and foreign keys. Predictive tasks over such data, such as forecasting customer churn, estimating item sales, and detecting fraud, require reasoning over multi-hop, heterogeneous, and time-stamped dependencies that span many tables. Relational deep learning (RDL) (Fey et al., 2024) addresses this setting by viewing a database as a temporal heterogeneous graph, where rows become nodes and key links become edges, and by training graph neural networks (GNNs) (Kipf & Welling, 2017; Hamilton et al., 2017; Schlichtkrull et al., 2018) directly on this graph.

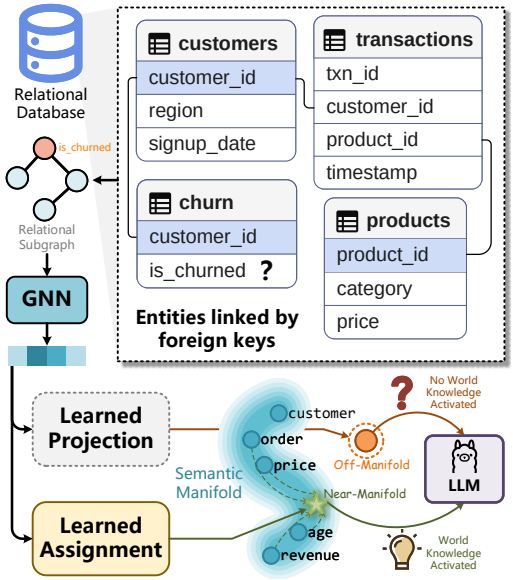

Figure 1: A learned projection places the graph prompt off the semantic manifold; our learned assignment anchors it to named schema concepts near it, activating LLM world knowledge.

Large language models (LLMs) have achieved remarkable success across natural language (Brown et al., 2020) and structured domains such as code (Chen et al., 2021), healthcare (Singhal et al., 2022), and finance (Wu et al., 2023). Through large-scale pre-training, LLMs acquire broad semantic and domain knowledge that can serve as transferable priors for relational learning, complementing database-specific supervision and improving the generalization of RDL models. A natural question is therefore whether the rich semantic space of LLMs can yield more informative

[1]Code is available at https://github.com/nmksjx/RelAnchor.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

representations for relational prediction. Existing efforts to connect LLMs with relational data fall into two lines. The first serializes the database around a target entity into flat text and relies on in-context learning (Wydmuch et al., 2024; Hegselmann et al., 2023), but it is bounded by the context window, discards structure beyond one or two hops, and cannot learn from labeled data at scale. The second, exemplified by Rel-LLM (Wu et al., 2025), first encodes a temporally sampled subgraph with a heterogeneous GNN (Hu et al., 2020) and then injects the resulting node embeddings into a frozen LLM as soft prompts via a learned projection (He et al., 2024; Lester et al., 2021; Li & Liang, 2021), achieving state-of-the-art results on the RelBench benchmark (Robinson et al., 2024).

This graph prompt, however, is *semantically blind*. Pre-trained token embeddings occupy a structured semantic manifold (Ethayarajh, 2019; Gao et al., 2019), the only region on which the LLM was trained to read input. A learned projection optimized solely for the downstream loss can place its outputs *anywhere*, and in practice lands them off this manifold, matching no word, concept, or schema element. The LLM must then read them through fine-tuned associations alone, leaving its pre-trained knowledge, indexed by lexical meaning, inactive: it cannot tell whether a token denotes a *customer* or a *transaction*. The GNN and the LLM are wired together but do not share a language.

Our key insight is a paradigm shift from learning the projection to learning the assignment (Figure 1). Rather than projecting freely into the ambient space, we learn how each row *distributes its meaning* over a fixed set of reference points, called *anchors*, that reside on the LLM's semantic manifold. Prompt construction thereby becomes anchored composition: whatever assignment is learned, every prompt is expressed relative to named schema concepts, so it stays semantically anchored near the region the LLM can interpret with its existing knowledge and without adaptation. We instantiate this idea in **RelAnchor**, which must overcome two challenges:

Semantic Dilution. An assignment requires reference points, but the LLM's full vocabulary spans hundreds of thousands of tokens covering all of natural language. Most of these tokens, words for cooking, astronomy, or sports, are irrelevant to any given database. Aligning GNN embeddings to this entire space introduces massive semantic noise from meanings that carry no relational signal.

Composition Bottleneck. A GNN embedding is a dense, continuous mixture of relational context: a customer's purchase frequency, return patterns, and spending trajectory. A prompt fully detached from the manifold is semantically blind, yet collapsing it exactly onto a few discrete anchors loses the fine-grained detail distinguishing rows: it must be anchored to the manifold, not confined to it.

To tackle Semantic Dilution, we introduce **Schema Anchoring**. Every database ships with a human-written dictionary of itself, its schema metadata, and we turn each table name, column name, and description into a *semantic anchor* by embedding it with the frozen input embedding layer of the LLM. Since this embedding layer defines the manifold on which the LLM reads, these anchors are on-manifold by construction, carry the associated world knowledge, and are immune to drift during training. Crucially, because they are drawn from the schema rather than the full vocabulary, they span exactly the semantic subspace that the database occupies.

To address Composition Bottleneck, we present **Adaptive Composition**. Rather than assigning each row to a single anchor, we represent it as a row-specific weighted mixture: the row's GNN embedding determines each anchor's contribution, and a learned gate blends this on-manifold composition with the row's graph vector. Navigating between the two extremes preserves distinctiveness: rows with different relational contexts weight anchors differently and retain instance-specific detail, while every prompt stays anchored near the manifold to meanings the LLM already understands.

Extensive experiments on RelBench show that RelAnchor outperforms GNN-only RDL baselines and Rel-LLM on the majority of individual classification and regression tasks, as well as on every aggregate metric, with ablations and visualizations verifying the effectiveness of each component.

2 Related Work

Relational Deep Learning. Classical pipelines flatten relational databases into a single table via feature engineering (Kanter & Veeramachaneni, 2015) and apply tabular learners such as TabNet (Arik & Pfister, 2021), FT-Transformer (Gorishniy et al., 2021), SAINT (Somepalli et al., 2021), and TabPFN (Hollmann et al., 2023); yet gradient-boosted trees remain competitive on single tables (Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022), suggesting the untapped signal lies in

cross-table structure. RDL (Fey et al., 2024) instead casts a database as a temporal heterogeneous graph, building on message-passing (Kipf & Welling, 2017; Hamilton et al., 2017), heterogeneous (Schlichtkrull et al., 2018; Wang et al., 2019; Hu et al., 2020), and temporal GNNs (Xu et al., 2020; Rossi et al., 2020). The RelBench benchmark (Robinson et al., 2024) standardizes leakage-safe evaluation, spurring RelGNN (Chen et al., 2025), ContextGNN (Yuan et al., 2025), the Relational Graph Transformer (Dwivedi et al., 2025), and foundation models Griffin (Wang et al., 2025) and the Relational Transformer (Ranjan et al., 2025); all train from scratch on each database, without semantic priors from pre-trained language models.

LLMs for Structured and Relational Data. One line *serializes* structured data into text, as in TabLLM (Hegselmann et al., 2023), LIFT (Dinh et al., 2022), UniPredict (Wang et al., 2023), table-pretraining models TaPas (Herzig et al., 2020), TaBERT (Yin et al., 2020), and TURL (Deng et al., 2020), and serialized entity neighborhoods as a strong RelBench baseline (Wydmuch et al., 2024); however, serialization is bounded by the context window and cannot scale to large label sets. A second line injects GNN embeddings as soft prompts via a learned projection: the GNN-as-prefix pattern of G-Retriever (He et al., 2024), GraphToken (Perozzi et al., 2024), GraphGPT (Tang et al., 2024), LLaGA (Chen et al., 2024), GraphLLM (Chai et al., 2023), and GraphPrompter (Liu et al., 2024b), surveyed in (Chen et al., 2023; Ren et al., 2024; Jin et al., 2024), and purely textual encodings in Fatemi et al. (2024). Rel-LLM (Wu et al., 2025) brings this pattern to RDL with a heterogeneous GNN over temporally sampled subgraphs, attaining state-of-the-art RelBench results. In all these methods the projection is *free*, trained on downstream loss alone, so prompts carry no explicit schema semantics; RelAnchor replaces it by an assignment over on-manifold anchors.

Prompt Learning and Semantic Grounding. RelAnchor builds on prompt learning, including prefix-tuning (Li & Liang, 2021), prompt tuning (Lester et al., 2021), and P-tuning (Liu et al., 2024a; 2022), and on instance-conditional prompts as in CoOp (Zhou et al., 2022b) and CoCoOp (Zhou et al., 2022a). Freely learned continuous prompts are semantically opaque: pre-trained representations concentrate on a narrow manifold (Ethayarajh, 2019; Mu & Viswanath, 2018; Gao et al., 2019), so arbitrary vectors fall off-manifold, and learned prompts admit arbitrary discrete readings (Khashabi et al., 2022). Discrete methods such as AutoPrompt (Shin et al., 2020), RLPrompt (Deng et al., 2022), and hard-prompt projection (Wen et al., 2023) share our semantic-anchoring motivation but target single textual tasks, while fixed-basis representations have precedent in vector quantization (van den Oord et al., 2017), codebook features (Tamkin et al., 2023), and concept bottleneck models (Koh et al., 2020). RelAnchor unifies these threads: anchors derived from schema metadata in the frozen token space give every prompt a named, on-manifold semantic reference, and row-specific mixtures over them preserve the distinctiveness a discrete assignment would collapse.

3 PRELIMINARIES

Relational databases. A relational database is a pair $\mathcal{D} = (\mathcal{T}, \mathcal{L})$, where $\mathcal{T} = \{T_1, \dots, T_n\}$ is a set of tables and $\mathcal{L} \subseteq \mathcal{T} \times \mathcal{T}$ is a set of inter-table links. Each table $T_i = \{r_1^{(i)}, \dots, r_{|T_i|}^{(i)}\}$ is a collection of rows, and each row $r \in T_i$ is a tuple $r = (k_r, \mathcal{K}_r, x_r, t_r)$, where k_r is the primary key, $\mathcal{K}_r$ is the set of foreign keys referencing rows in other tables, x_r denotes the attribute values of the row (numerical, categorical, or textual), and $t_r \in \mathbb{R} \cup \{-\infty\}$ is an optional timestamp indicating when the row became valid ($t_r = -\infty$ for static rows). A link $(T_i, T_j) \in \mathcal{L}$ exists whenever a foreign key of T_i references the primary key of T_j.

Relational entity graphs and graph-prompted LLMs. Following the relational deep learning formulation (Fey et al., 2024), we cast $\mathcal{D}$ as a temporal heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi)$, where each node $v \in \mathcal{V}$ corresponds to a row, each edge $(u, v) \in \mathcal{E}$ to a primary–foreign key relation, and $\phi : \mathcal{V} \to \mathcal{T}$, $\psi : \mathcal{E} \to \mathcal{L}$ assign node and edge types. Every node v inherits the attributes x_v and timestamp t_v of its row. A predictive task supplies training triples $\{(v_s, t_s, y_s)\}_{s=1}^{S}$, where $v_s \in \mathcal{V}$ is a seed entity, t_s a seed time, and $y_s \in \mathcal{Y}$ the label to predict using only information available before t_s. To avoid temporal leakage, a subgraph $\mathcal{G}_{v_s,t_s} \subseteq \mathcal{G}$ is sampled by temporal neighbor sampling (Hamilton et al., 2017; Robinson et al., 2024), retaining only nodes u with $t_u < t_s$. A heterogeneous GNN encoder f_θ (Hu et al., 2020) then produces node representations

$$\mathbf{H}_{v_s,t_s} = f_\theta(\mathcal{G}_{v_s,t_s}) \in \mathbb{R}^{N \times d_g}, \qquad \mathbf{h}_u = [\mathbf{H}_{v_s,t_s}]_u, \tag{1}$$

where $N = |\mathcal{G}_{v_s,t_s}|$ and d_g is the GNN hidden dimension. Let $\mathcal{M}$ denote a frozen decoder-only LLM with vocabulary $\mathcal{V}_{\text{tok}}$ and input token embedding table $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}_{\text{tok}}| \times d_\ell}$. Existing GNN-as-

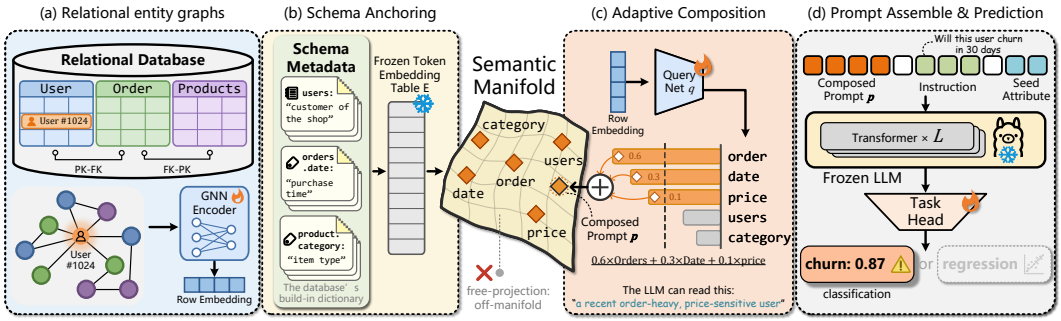

Figure 2: **Overview of RelAnchor:** (a) a GNN encodes the relational entity graph into row embeddings; (b) Schema Anchoring embeds automatically generated schema metadata with the frozen token embedding table into semantic anchors; (c) Adaptive Composition assigns and integrates an anchor mixture for each graph token; (d) the frozen LLM reads the assembled prompt for prediction.

prefix methods (Wu et al., 2025; He et al., 2024) inject graph information via a *free projection* $g_\omega : \mathbb{R}^{d_g} \to \mathbb{R}^{d_\ell}$, producing soft prompt vectors $\mathbf{p}_u = g_\omega(\mathbf{h}_u)$ prepended to the textual task instruction. As noted in Section 1, this projection yields unconstrained points in $\mathbb{R}^{d_\ell}$ that need not lie on the LLM's semantic manifold and therefore carry no lexical meaning the model can recognize. RelAnchor replaces it with a learned *assignment* over fixed on-manifold reference points, described next.

4 Methodology

We now present **RelAnchor**, a semantic-space-informed prompt learning framework for relational deep learning (Figure 2): *Schema Anchoring* builds a database-specific anchor vocabulary from schema metadata (Section 4.1); lightweight *Query-Conditioned Retention* adapts the graph prompt to the task and instance (Section 4.2); *Adaptive Composition* assigns and integrates row-specific anchor mixtures (Section 4.3); and Section 4.4 covers prompt assembly, prediction, and the loss.

4.1 Schema Anchoring

Schema metadata as a native vocabulary. Our key observation is that a relational database exposes a structured vocabulary through its schema and train-visible statistics. For each table $T_i \in \mathcal{T}$, we automatically construct descriptions for the table, its columns, keys, joins, and column statistics, and we collect these textual fragments into a metadata set

$$\mathcal{A}_{\text{meta}} = \underbrace{\{\, d(T_i) : T_i \in \mathcal{T} \,\}}_{\text{table descriptions}} \cup \underbrace{\{\, d(c) : c \in \text{cols}(T_i),\ T_i \in \mathcal{T} \,\}}_{\text{column descriptions}}, \tag{2}$$

where $d(\cdot)$ denotes a deterministic template and $\text{cols}(T_i)$ is the column set of table T_i. The resulting database-specific vocabulary scales with schema elements and relations rather than the row number.

Anchor construction. Each metadata entry $m_j \in \mathcal{A}_{\text{meta}}$ is tokenized into $(w_{j,1}, \ldots, w_{j,L_j})$ and embedded with the *frozen input token embedding table* $\mathbf{E}$, yielding the j-th *semantic anchor*

$$\mathbf{a}_j = \frac{1}{L_j} \sum_{l=1}^{L_j} \mathbf{E}[w_{j,l}] \ \in \ \mathbb{R}^{d_\ell}, \qquad j = 1, \ldots, M, \tag{3}$$

which we stack into the anchor matrix $\mathbf{A} = [\mathbf{a}_1; \ldots; \mathbf{a}_M] \in \mathbb{R}^{M \times d_\ell}$. *As the embedding of a named schema description, each anchor lies on the semantic manifold by construction, carries the world knowledge the LLM associates with the corresponding schema concept, and, being derived from frozen parameters, cannot drift during training.* The anchor matrix is computed once per database and shared across all tasks defined on that database.

4.2 Query-Conditioned Retention

Before anchor assignment, the graph prompt $\mathbf{G} = [\mathbf{g}_0; \ldots; \mathbf{g}_{N-1}]$ from the GNN and projector, with the seed at position zero followed by sampled one-hop neighbors, is adapted to the prediction without changing its length. We form three views: the local prompt $\mathbf{G}$ and the broadcast global and

neighbor means $\bar{\mathbf{g}} = N^{-1}\sum_i \mathbf{g}_i$ and $\bar{\mathbf{g}}_{\text{nbr}} = (N-1)^{-1}\sum_{i>0}\mathbf{g}_i$. A conditioning vector combines the mean task-description and question embedding with $\mathbf{g}_0$ and $\bar{\mathbf{g}}$. Two trainable projection networks score each view against it, and a softmax yields instance-dependent weights β_v. The retained vectors $\widetilde{\mathbf{g}}_u$ are the weighted sum of the views at non-seed positions, with position zero restored to $\mathbf{g}_0$.

4.3 Adaptive Composition

From projection to assignment. Given the anchor basis $\mathbf{A}$, RelAnchor no longer regresses a free prompt vector for each node. Instead, it learns *how each row distributes its meaning over the anchors*. Concretely, a lightweight query network $q_\omega : \mathbb{R}^{d_g} \to \mathbb{R}^{d_\ell}$ (a two-layer MLP) maps the GNN embedding $\mathbf{h}_u$ of node u into the LLM space, and compatibility scores against all anchors are computed as scaled dot products, $\mathbf{z}_u = \mathbf{A}\, q_\omega(\mathbf{h}_u)/\sqrt{d_\ell} \in \mathbb{R}^M$, where $[\mathbf{z}_u]_j$ measures how strongly the relational context of row u resonates with the j-th schema concept.

Soft anchor assignment. The assignment weights are obtained by a softmax over the database-specific anchor dictionary,

$$[\boldsymbol{\alpha}_u]_j = \frac{\exp\big([\mathbf{z}_u]_j/\tau\big)}{\sum_{j'=1}^{M}\exp\big([\mathbf{z}_u]_{j'}/\tau\big)}, \qquad j = 1,\dots,M, \tag{4}$$

where $\tau > 0$ is a temperature hyperparameter and $\boldsymbol{\alpha}_u \in \Delta^{M-1}$ lies on the probability simplex. The semantic vector of node u is then $\mathbf{s}_u = \sum_{j=1}^{M}[\boldsymbol{\alpha}_u]_j\,\mathbf{a}_j = \boldsymbol{\alpha}_u^\top \mathbf{A} \in \mathbb{R}^{d_\ell}$. *This composition gives each row a differentiable semantic reference conditioned on its relational representation.*

Semantic integration. To balance semantic grounding with instance-specific graph information, we integrate each anchor composition with its retained graph vector through a learnable scalar gate $\alpha = \sigma(a)$ shared across nodes, yielding the graph prompt token of node u, $\mathbf{p}_u = (1-\alpha)\widetilde{\mathbf{g}}_u + \alpha\mathbf{s}_u$. Because $\mathbf{s}_u$ lies in the *anchor hull* $\mathcal{S} = \text{conv}(\{\mathbf{a}_j\}_{j=1}^{M})$, the gate contracts each token towards $\mathcal{S}$ by a factor $1-\alpha$ (Proposition 4, Appendix F.6): every prompt stays anchored to its schema concepts near the manifold while retaining instance-specific graph detail.

4.4 Prompt Assembly, Prediction, and Training

Prompt assembly. For a task sample (v_s, t_s), we compose the LLM input from the graph-prompt vectors $\mathbf{P}_{v_s,t_s} = [\mathbf{p}_{u_1};\dots;\mathbf{p}_{u_N}] \in \mathbb{R}^{N\times d_\ell}$, one neighborhood-level semantic token $\mathbf{s}_{\text{G}}$, and the token embeddings of a textual task instruction $\mathcal{I}$ describing the prediction target in natural language. Denoting by $\text{emb}(\cdot)$ the lookup of textual tokens in $\mathbf{E}$, the full input sequence is

$$\mathbf{X}_{v_s,t_s} = \big[\,\mathbf{e}_{\text{BOS}} \,\|\, \mathbf{P}_{v_s,t_s} \,\|\, \mathbf{s}_{\text{G}} \,\|\, \text{emb}(\mathcal{I}) \,\|\, \mathbf{e}_{\text{EOT}}\,\big], \tag{5}$$

where $\|$ denotes sequence concatenation. Entity attributes enter through the graph encoder, and variable-length inputs are left-padded with attention masks.

Prediction. The frozen LLM $\mathcal{M}$ processes $\mathbf{X}_{v_s,t_s}$, and the final hidden state $\mathbf{o}_{v_s,t_s} \in \mathbb{R}^{d_\ell}$ at the last position is passed to a task head ρ_ξ: a linear-sigmoid head producing $\hat{y}_s = \sigma(\mathbf{w}^\top \mathbf{o}_{v_s,t_s} + b)$ for binary classification, and a linear head $\hat{y}_s = \mathbf{w}^\top \mathbf{o}_{v_s,t_s} + b$ for regression, with $\mathbf{w} \in \mathbb{R}^{d_\ell}$ and $b \in \mathbb{R}$.

Overall objective. Training minimizes the task loss alone, matching the RelBench protocol (Robinson et al., 2024). For binary classification, $\mathcal{L}_{\text{task}}$ is the binary cross-entropy over the training triples, and for regression it is the mean absolute (L_1) error:

$$\mathcal{L}_{\text{task}} = -\frac{1}{S}\sum_{s=1}^{S}\Big[\, y_s \log \hat{y}_s + (1-y_s)\log\big(1-\hat{y}_s\big)\,\Big] \;\text{(cls.)}, \qquad \mathcal{L}_{\text{task}} = \frac{1}{S}\sum_{s=1}^{S}\big|\,\hat{y}_s - y_s\,\big| \;\text{(reg.)}. \tag{6}$$

All trainable parameters, namely the GNN encoder θ, query and retention networks ω, gate parameter a, and task head ξ, are optimized jointly, $\min_{\theta,\omega,a,\xi}\mathcal{L}_{\text{task}}$, while the LLM $\mathcal{M}$, its embedding table $\mathbf{E}$, and the anchor matrix $\mathbf{A}$ remain frozen. No auxiliary regularizers are required: the fixed anchors supply a stable semantic reference, while the task loss shapes the graph representations, assignments, gate, and task heads. Algorithm 1 in the appendix summarizes the end-to-end procedure.

5 Theoretical Analysis

We quantify the cost of anchoring prompts as convex compositions of frozen anchors: for any target prompt in the anchor hull, some k-sparse composition shifts the LLM's output by only $O(1/\sqrt{k})$,

Table 1: **Classification results on RelBenchV1.** AUROC in percent (↑); Avg is the mean over the 12 tasks. **Bold** / underline : best / second-best per column; : ours.

Method	f1		avito		event		trial	amazon		stack		hm	Avg
	dnf	top3	click	visit	repeat	ignore	out	user	item	eng	badge	churn	↑
Zero-shot													
Rel-Zero	71.84	70.64	62.28	56.17	68.12	61.32	59.02	60.07	64.10	69.46	62.12	55.95	63.42
Griffin	57.70	82.50	45.90	60.70	71.88	83.27	51.00	62.30	69.00	77.50	73.50	60.20	66.29
RT_{zero}	81.20	89.30	59.50	61.80	73.22	77.47	51.80	64.00	70.90	75.70	80.10	62.80	70.65
$\text{RELANCHOR}_{\text{zero}}$	77.05	84.62	65.34	68.42	**88.96**	88.35	62.75	63.78	73.79	84.30	81.77	67.31	75.54
Few-shot / ICL													
TabPFN-2.5	61.21	77.35	54.44	51.45	69.10	70.91	71.74	57.14	59.30	59.66	52.26	54.99	61.63
RDBLearn	70.87	79.69	**69.04**	65.49	75.04	82.52	71.58	67.57	82.07	89.39	85.26	68.05	75.55
Task-specific													
LightGBM	68.56	73.92	53.60	53.05	68.04	79.93	70.09	52.22	62.54	63.39	63.43	55.21	63.67
DS+LightGBM	69.80	82.40	64.27	64.46	75.42	84.23	72.00	67.60	81.80	90.30	86.20	69.00	75.62
GraphSAGE	72.62	75.54	65.90	66.20	76.89	81.62	68.60	70.42	82.81	90.59	88.86	69.88	75.83
HGT	70.77	70.75	63.76	64.32	64.96	82.47	58.37	66.43	77.97	88.47	86.08	66.95	71.78
HGT_{PE}	71.17	76.27	64.57	64.95	65.36	81.61	59.21	66.19	78.03	88.17	85.66	65.69	72.24
RelGNN	75.29	85.69	68.23	66.18	79.61	86.18	71.24	70.99	82.64	90.75	88.98	**70.93**	78.06
RelGT	75.87	83.52	68.30	66.78	76.09	81.57	68.61	70.39	82.55	90.53	86.32	69.27	76.65
Rel-LLM	77.15	82.22	66.74	67.01	79.26	83.74	71.04	**71.89**	**83.37**	91.21	89.64	70.55	77.82
RELANCHOR	**81.49**	**90.77**	66.39	**70.34**	88.31	**92.92**	**75.74**	70.02	81.72	**91.29**	**90.26**	67.68	**80.58**

with a constant set by its Lipschitz behavior on the compact, on-manifold anchor hull, not the unbounded ambient space.

Throughout, $\mathbf{E} = [\mathbf{e}_1; \ldots; \mathbf{e}_{|\mathcal{V}_{\text{tok}}|}] \in \mathbb{R}^{|\mathcal{V}_{\text{tok}}| \times d_\ell}$ is the frozen token embedding table, and the anchor matrix $\mathbf{A} = [\mathbf{a}_1; \ldots; \mathbf{a}_M] \in \mathbb{R}^{M \times d_\ell}$ is that of Eq. 3, with diameter $D_{\mathcal{A}} = \max_{i,j} \|\mathbf{a}_i - \mathbf{a}_j\|_2$. $\mathcal{S} = \text{conv}(\{\mathbf{a}_j\}_{j=1}^M)$ is the anchor hull of Section 4.3, a compact on-manifold region fixed before training (Appendix F.4 fixes the formal status of the semantic manifold and why $\mathcal{S}$, not the token-embedding hull, is the right formal object). Holding the textual segments of the LLM input fixed, we write $F : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_o}$ for the dependence of the frozen LLM's output representation on a single prompt-token position, and assume F is $L_{\mathcal{S}}$-Lipschitz on $\mathcal{S}$.

Theorem 1 (On-Manifold Prompt Fidelity)**.** *For any target prompt $\mathbf{p}^* \in \mathcal{S}$ and any integer $k \geq 1$, there exists $\boldsymbol{\alpha} \in \Delta^{M-1}$ with $\|\boldsymbol{\alpha}\|_0 \leq k$ such that the composed prompt $\mathbf{p} = \boldsymbol{\alpha}^\top \mathbf{A}$ lies in $\mathcal{S}$ and*

$$\left\|F(\mathbf{p}) - F(\mathbf{p}^*)\right\|_2 \;\leq\; \frac{L_{\mathcal{S}}\, D_{\mathcal{A}}}{\sqrt{2k}}. \tag{7}$$

Proof. Every composition $\boldsymbol{\alpha}^\top \mathbf{A}$ with $\boldsymbol{\alpha} \in \Delta^{M-1}$ lies in $\mathcal{S} = \text{conv}(\mathbf{A})$ by definition of the anchor hull. This settles the membership claim and reduces the theorem to exhibiting a k-sparse $\boldsymbol{\alpha}$ with $\|\boldsymbol{\alpha}^\top \mathbf{A} - \mathbf{p}^*\|_2 \leq D_{\mathcal{A}}/\sqrt{2k}$, since the Lipschitz property of F on $\mathcal{S}$ then yields Eq. 7.

We obtain such an $\boldsymbol{\alpha}$ by Maurey's empirical method. Write $\mathbf{p}^* = \sum_{j=1}^M \beta_j \, \mathbf{a}_j$ with $\boldsymbol{\beta} \in \Delta^{M-1}$, let $J_1, \ldots, J_k$ be i.i.d. indices with $\Pr(J_i = j) = \beta_j$, and set $\hat{\mathbf{p}} = \frac{1}{k} \sum_{i=1}^k \mathbf{a}_{J_i}$. The random anchor $\mathbf{X} = \mathbf{a}_{J_1}$ has mean $\mathbf{p}^*$, and the summands $\mathbf{a}_{J_i} - \mathbf{p}^*$ are independent with mean zero, so

$$\mathbb{E}\,\|\hat{\mathbf{p}} - \mathbf{p}^*\|_2^2 = \frac{1}{k}\,\mathbb{E}\,\|\mathbf{X} - \mathbf{p}^*\|_2^2 = \frac{1}{2k}\,\mathbb{E}\,\|\mathbf{X} - \mathbf{X}'\|_2^2 \leq \frac{D_{\mathcal{A}}^2}{2k},$$

where $\mathbf{X}'$ is an independent copy of $\mathbf{X}$. The middle equality is the identity $\mathbb{E}\|\mathbf{X} - \mathbb{E}\mathbf{X}\|^2 = \frac{1}{2}\mathbb{E}\|\mathbf{X} - \mathbf{X}'\|^2$, and the inequality holds because $\mathbf{X}$ and $\mathbf{X}'$ are anchors and hence at distance at most $D_{\mathcal{A}}$. Some realization $(j_1, \ldots, j_k)$ attains at most the expectation, and for it $\|\hat{\mathbf{p}} - \mathbf{p}^*\|_2 \leq D_{\mathcal{A}}/\sqrt{2k}$. Setting $[\boldsymbol{\alpha}]_j = |\{i : j_i = j\}|/k$ gives $\boldsymbol{\alpha} \in \Delta^{M-1}$ with at most k nonzero entries and $\boldsymbol{\alpha}^\top \mathbf{A} = \hat{\mathbf{p}}$. Because $\hat{\mathbf{p}}$ and $\mathbf{p}^*$ both lie in $\mathcal{S}$,

$$\left\|F(\hat{\mathbf{p}}) - F(\mathbf{p}^*)\right\|_2 \leq L_{\mathcal{S}}\,\|\hat{\mathbf{p}} - \mathbf{p}^*\|_2 \leq \frac{L_{\mathcal{S}}\, D_{\mathcal{A}}}{\sqrt{2k}}.$$

□

Since $\mathbf{p} \in \mathcal{S}$ holds for every composition the model can express, the manifold-restricted constant $L_{\mathcal{S}}$ applies unconditionally, whereas a freely projected prompt may fall outside $\mathcal{S}$, where only a global

Table 2: **Regression results on RelBenchV1.** MAE in each task's original units (↓); Raw Avg. is the mean of the nine task MAEs, and the unit-invariant μ_n normalizes each MAE by LightGBM's before averaging. **Bold** / underline : best / second-best per column; : ours.

Method	f1	avito	event	trial		amazon		stack	hm	Raw Avg.	μ_n
	pos	ctr	attend	adverse	success	user	item	votes	sales	↓	↓
Zero-shot											
Griffin	4.460	0.050	0.461	78.232	0.463	35.590	53.214	0.092	0.151	19.1903	1.4712
RT_{zero}	2.901	0.058	0.379	73.999	0.455	18.802	57.996	0.110	0.089	17.1988	1.2405
RELANCHOR$_{zero}$	4.326	0.043	0.285	43.464	0.466	15.447	67.442	0.064	0.086	14.6248	1.0396
Few-shot / ICL											
TabPFN-2.5	4.446	0.044	0.527	55.187	0.469	17.513	60.996	0.104	0.096	15.4869	1.2596
RDBLearn	3.834	**0.034**	**0.237**	43.913	0.424	14.540	48.559	0.068	0.064	12.4081	0.9058
Task-specific											
Global Median	4.399	0.043	0.264	57.533	0.462	16.783	64.234	0.068	0.076	15.9847	1.0621
Entity Median	8.519	0.046	0.269	57.930	0.441	17.423	66.436	0.069	0.078	16.8012	1.1904
LightGBM	4.170	0.041	0.264	44.011	0.425	16.783	60.569	0.068	0.076	14.0452	1.0000
DS+LightGBM	3.963	0.044	0.284	40.581	0.407	13.928	**41.122**	0.065	0.036	11.1589	0.8797
GraphSAGE	4.022	0.041	0.258	44.473	0.400	14.313	50.053	0.065	0.056	12.6312	0.9184
HGT	4.226	0.046	0.264	45.169	0.443	15.412	55.868	0.068	0.064	13.5067	0.9874
HGT_{PE}	4.392	0.048	0.261	42.648	0.440	15.864	55.849	0.068	0.064	13.2927	0.9918
RelGNN	3.798	0.037	0.238	44.461	**0.301**	14.230	48.767	0.065	0.054	12.4390	0.8614
RelGT	3.917	0.035	0.250	43.992	0.326	14.267	48.922	0.065	0.054	12.4253	0.8701
Rel-LLM	3.967	0.037	0.251	43.682	0.397	14.087	48.224	0.062	0.052	12.3066	0.8847
RELANCHOR	**2.785**	0.037	0.282	**30.038**	0.352	**13.745**	47.283	**0.058**	**0.035**	**10.5128**	**0.7847**

constant $L_{\text{glob}} \geq L_{\mathcal{S}}$ is available and the gap can be large (Remark 1, Appendix F.3); this gap is the quantitative face of semantic blindness. Moreover, sparsity does not sacrifice distinctiveness: under a mild general-position condition on the anchors, distinct sparse assignments yield distinct prompts (Proposition 5, Appendix F.7). The gated prompt tokens of Section 4.3 inherit this control in graded form: the gate contracts each token towards $\mathcal{S}$ by a factor $1 - \alpha$ (Proposition 4, Appendix F.6), so every prompt RelAnchor emits stays anchored near the manifold. Appendix F further gives the underlying sparsification lemma and a multi-token extension.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

We evaluate on 21 RelBenchV1 tasks (12 classification, 9 regression) across seven databases and six RelBenchV2 tasks from RateBeer and ArXiv (Robinson et al., 2024). Classification uses AUROC (0–100); regression, MAE and the LightGBM-normalized mean μ_n. We use leave-one-database-out pretraining: all non-target databases form the source pool, with no target-task labels seen. Results are grouped into *zero-shot*, *few-shot / ICL*, and *task-specific* regimes. We compare against tabular (LightGBM, DS+LightGBM/AutoGluon/TabPFN-2.5), GNN (GraphSAGE, HGT, RelGNN, RelGT), foundation-model (Griffin, RT_{zero}, TabPFN-2.5, RDBLearn), and LLM (Rel-LLM, Llama-3.2-1B/3B + MLP) baselines. RelAnchor uses a frozen Llama 3.2–1B backbone with trainable encoder, query and retention networks, gate, and task head; the validation-best checkpoint is evaluated on the test split. Appendix C details benchmarks, pretraining, baselines, and implementation.

6.2 PREDICTIVE PERFORMANCE

On RelBenchV1 (Tables 1 and 2), RelAnchor$_{zero}$ leads all zero-shot methods on both aggregates (75.54 mean AUROC, $\mu_n = 1.0396$), surpassing the runner-up RT_{zero} by 4.89 AUROC points and 16.2% in μ_n, and coming within 0.29 points of task-specific GraphSAGE on classification. Task-specific, RelAnchor again leads both aggregates (80.58 mean AUROC, $\mu_n = 0.7847$), ahead of RelGNN (78.06, 0.8614) and Rel-LLM (77.82, 0.8847). Gains over Rel-LLM are largest where semantic grounding plausibly matters most: 9.05 and 9.18 AUROC points on Event user-repeat and user-ignore, and 31.2% lower MAE on Trial study-adverse (43.682 to 30.038). On RelBenchV2

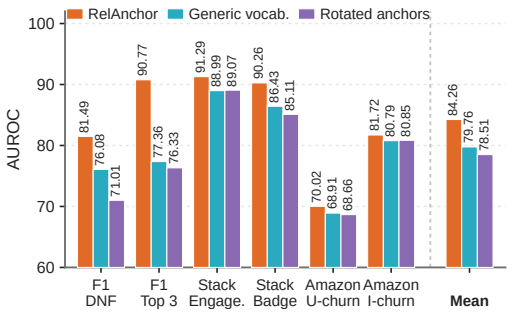

Figure 3: **Schema Anchoring.** AUROC per classification task and mean per dictionary.

Table 3: **Schema names and statistical descriptors.** Per-task/mean AUROC; mean MAE.

Task	Full	W/o names	W/o statistics
F1 DNF	**81.49**	73.67	72.18
F1 Top 3	**90.77**	74.28	74.43
Stack Engage.	**91.29**	88.17	88.89
Stack Badge	**90.26**	85.51	85.31
Amazon U-churn	**70.02**	69.43	67.44
Amazon I-churn	**81.72**	79.18	79.29
Mean AUROC (6 cls.) ↑	**84.26**	78.37	77.92
Mean MAE (4 reg.) ↓	**15.9678**	17.2803	17.3370

(Appendix D), zero-shot RelAnchor already exceeds task-specific LightGBM on entirely unseen schemas, and task-specific RelAnchor attains the best mean AUROC (86.43) and raw MAE (3.6061).

6.3 Ablation Studies

Schema Anchoring. We replace the schema anchors with two frozen, size-matched controls: *generic vocabulary embeddings* (M sampled non-special alphabetic tokens, norm-matched to the replaced anchors), which are on-manifold but database-agnostic, and *orthogonally rotated anchors* $\mathbf{A}_{\text{rot}} = \mathbf{AQ}$, which preserve all norms and pairwise inner products but reorient the dictionary relative to the LLM's coordinates. Figure 3 shows RelAnchor ahead on all six classification tasks: mean AUROC drops from 84.26 to 79.76 (generic) and 78.51 (rotated), with the damage concentrated on F1 (up to 14.44 points on driver-top3) versus 0.87–5.15 points on Stack and Amazon. Since rotation alters only the dictionary's orientation, its cost isolates on-manifold alignment; both database specificity and expression in the LLM's native embedding coordinates are therefore necessary.

Schema names and statistical descriptors. Anonymizing table and column identifiers, or removing the statistical-description phrases (with anchor slots and eligibility masks preserved), tests which metadata the anchors draw on. In Table 3, each control loses on all six classification tasks, lowering mean AUROC to 78.37 and 77.92 and raising mean MAE to 17.2803 and 17.3370, with the largest drops on F1 driver-top3 (16.49 and 16.34 points). The two controls are nearly tied, so names and statistics carry comparable, complementary signal.

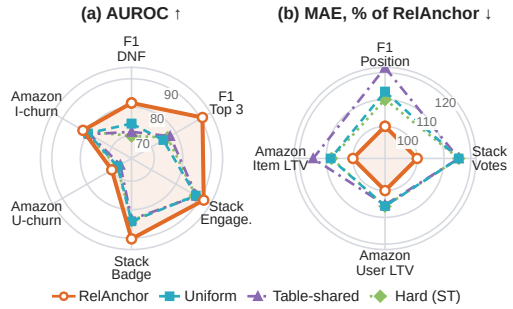

Figure 4: **Adaptive Composition.** (a) AUROC per classification task; (b) MAE per regression task as a percentage of RelAnchor.

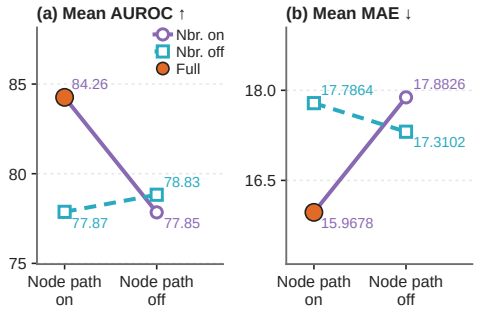

Figure 5: **Semantic pathways.** Mean (a) AUROC and (b) MAE; node path on/off, neighborhood token kept (solid) or removed (dashed).

Adaptive Composition. We replace learned, row-specific soft assignment with three rules sharing all other components: a *uniform mixture*, a *table-shared mixture* (one learned logit vector per table), and *hard assignment (ST)* (top-scoring anchor with a straight-through gradient). In Figure 4a the RelAnchor polygon encloses all controls on every classification axis, with the widest gaps on F1 (up to 15.90 points on driver-top3) and the narrowest on Amazon item-churn (0.75 to hard assignment); in (b) every control has higher MAE on all four regression tasks, up to 15.5% on Stack post-votes. In mean AUROC, the controls trail by 6.17–6.53 points (78.09, 77.73, 77.85 versus 84.26), and RelAnchor has the lowest mean MAE (Appendix E). Both row specificity and softness are needed.

Node and neighborhood semantic pathways. Figure 5 arranges as a 2×2 factorial the effect of zeroing the node-injection gate, zeroing the neighborhood semantic token, or both. The lines cross rather than run parallel: removing either pathway alone costs about 6.4 mean AUROC points (77.85 and 77.87 versus 84.26) and raises mean MAE to 17.8826 and 17.7864, whereas removing both recovers part of the loss (78.83, 17.3102). This non-monotonic ordering shows the pathways act as a coupled prompt design rather than contributing independent additive gains.

Shared-gate sensitivity. Retraining with α fixed to $\{0, 0.10, 0.25, 0.50, 0.75, 1.00\}$ reveals task-dependent optima (Figure 6): F1 driver-dnf peaks at 0.75 but driver-top3 at 0.25, and Amazon item-churn favors 0 while item-ltv is best at 0.50; a wrong fixed value is costly, with item-ltv MAE rising to 61.03 at $\alpha = 1$ versus 47.28 with the learned gate. The learned gate beats all six fixed settings on every task shown, justifying a learnable α that sets the semantic injection strength per model at the cost of a single shared scalar.

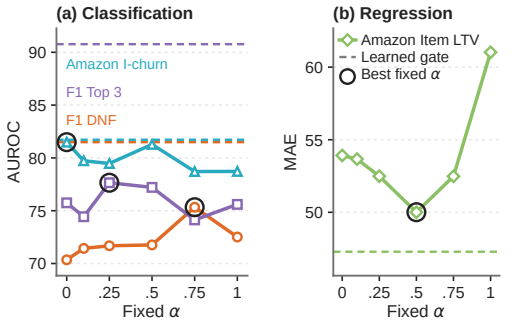

Figure 6: **Shared-gate sensitivity.** (a) AUROC of three classification tasks and (b) MAE of Amazon item-ltv against fixed α; dashed: learned gate.

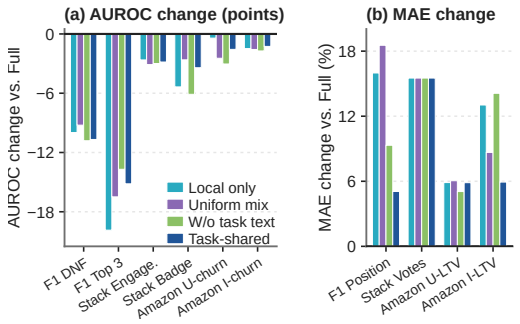

Figure 7: **Query-Conditioned Retention.** (a) AUROC change of each control vs. full RelAnchor per classification task; (b) MAE change per regression task.

Query-Conditioned Retention. We ablate retention with four controls: *local prompt only* (no view mixing), *uniform mixed views*, *remove task text* from the conditioning vector (the LLM's instruction is unchanged), and *task-shared weights* (per-task logits replacing instance-dependent ones). In Figure 7, all bars are negative: every control loses on every task. In the mean, the four variants trail by 6.62, 5.91, 6.40, and 5.82 AUROC points and raise MAE from 15.9678 to 17.8240, 17.3305, 17.8768, and 16.9075 (Appendix E). View mixing is the single most valuable ingredient, and retention requires learned weights conditioned on both task semantics and instance-level graph context.

7 Conclusion and Limitations

We identified semantic blindness as a fundamental limitation of graph-prompted LLMs for relational data: a free projection places graph prompts off the semantic manifold, leaving the LLM's pretrained knowledge inactive. We proposed RelAnchor, which replaces the learned projection with a learned assignment via Schema Anchoring, deriving on-manifold anchors from schema metadata, and Adaptive Composition, expressing each row as a mixture over these anchors gated with its graph representation, so every prompt is anchored to named schema concepts near the manifold. Our analysis shows this anchoring costs only $O(1/\sqrt{k})$ in prompt fidelity while enabling the tighter on-manifold Lipschitz constant, and experiments on RelBench show gains over GNN-only RDL baselines and Rel-LLM on most classification and regression tasks and on every aggregate metric.

Two directions remain open. First, the expressiveness of the anchor basis is bounded by the descriptiveness of the schema metadata, so databases with terse or auto-generated column names may yield less informative anchors; enriching the basis with LLM-generated or statistics-driven descriptions is a promising extension. Second, our evaluation focuses on entity-level classification and regression with a single frozen backbone, and extending semantically anchored composition to relational recommendation tasks and studying its behavior as the LLM scales are natural next steps.

AI USE STATEMENT

In this work, we used generative AI tools to formulate mathematical equations and to implement methods. We also used them to polish the paper for readability. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including any text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made the following efforts to ensure reproducibility. The RelAnchor architecture, including Schema Anchoring, Query-Conditioned Retention, Adaptive Composition, and the training objective, is fully specified in Section 4. All experiments use the public RelBenchV1 and RelBenchV2 benchmarks; Appendix C.1 describes the tasks and metrics, Appendix C.2 the leave-one-database-out pretraining protocol, Appendix C.4 the evaluation regimes, Appendix C.5 the baselines, and Appendix C.6 the backbone, training protocol, update budget, and model-selection procedure. Complete per-task ablation results are reported in Appendix E. For the theoretical results, the assumptions of Theorem 1 are stated in Section 5, and the supporting lemma, the complete proofs, and the multi-token extension are given in Appendix F. Our source code is available in an anonymous repository at https://github.com/nmksjx/RelAnchor. It contains the complete RelAnchor pipeline, including the anchor-construction code, the model, and a single entry point for both pretraining and fine-tuning, together with the per-task hyperparameter configurations of all 27 tasks, the task instructions, and the evaluation protocols used in our experiments. The base LLM, the text encoder, and the RelBench databases are obtained from their public sources.

REFERENCES

Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 6679–6687. AAAI Press, 2021. doi: 10.1609/AAAI.V35I8.16826. URL https://doi.org/10.1609/aaai.v35i8.16826.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. Graphllm: Boosting graph reasoning ability of large language model. *CoRR*, abs/2310.05845, 2023. doi: 10.48550/ARXIV.2310.05845. URL https://doi.org/10.48550/arXiv.2310.05845.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech

Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7809–7823. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/chen24bh.html.

Tianlang Chen, Charilaos I. Kanatsoulis, and Jure Leskovec. Relgnn: Composite message passing for relational deep learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/chen25ad.html.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor.*, 25(2):42–61, 2023. doi: 10.1145/3655103.3655110. URL https://doi.org/10.1145/3655103.3655110.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 3369–3391. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.222. URL https://doi.org/10.18653/v1/2022.emnlp-main.222.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. TURL: table understanding through representation learning. *Proc. VLDB Endow.*, 14(3):307–319, 2020. doi: 10.14778/3430915.3430921. URL http://www.vldb.org/pvldb/vol14/p307-deng.pdf.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris S. Papailiopoulos, and Kangwook Lee. LIFT: language-interfaced fine-tuning for non-language machine learning tasks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/4ce7fe1d2730f53cb3857032952cd1b8-Abstract-Conference.html.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer. *CoRR*, abs/2505.10960, 2025. doi: 10.48550/ARXIV.2505.10960. URL https://doi.org/10.48550/arXiv.2505.10960.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander J. Smola. Autogluon-tabular: Robust and accurate automl for structured data. *CoRR*, abs/2003.06505, 2020. URL https://arxiv.org/abs/2003.06505.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 55–65. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1006. URL https://doi.org/10.18653/v1/D19-1006.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=IuXR1CCrSi.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Position: Relational deep learning - graph representation learning on relational databases. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13592–13607. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/fey24a.html.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkEYojRqtm.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 18932–18943, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/0378c7692da36807bdec87ab043cdadc-Abstract-Datasets_and_Benchmarks.html.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-2.5: Advancing the state of the art in tabular foundation models. *CoRR*, abs/2511.08667, 2025. doi: 10.48550/ARXIV.2511.08667. URL https://doi.org/10.48550/arXiv.2511.08667.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/efaf1c9726648c8ba363a5c927440529-Abstract-Conference.html.

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

J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent (eds.), *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5549–5581. PMLR, 2023. URL https://proceedings.mlr.press/v206/hegselmann23a.html.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 4320–4333. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.398. URL https://doi.org/10.18653/v1/2020.acl-main.398.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (eds.), *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pp. 2704–2710. ACM / IW3C2, 2020. doi: 10.1145/3366423.3380027. URL https://doi.org/10.1145/3366423.3380027.

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. Kumorfm-2: Scaling foundation models for relational learning. *CoRR*, abs/2604.12596, 2026. doi: 10.48550/ARXIV.2604.12596. URL https://doi.org/10.48550/arXiv.2604.12596.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE Trans. Knowl. Data Eng.*, 36(12):8622–8642, 2024. doi: 10.1109/TKDE.2024.3469578. URL https://doi.org/10.1109/TKDE.2024.3469578.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015*, pp. 1–10. IEEE, 2015. doi: 10.1109/DSAA.2015.7344858. URL https://doi.org/10.1109/DSAA.2015.7344858.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 3146–3154, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

Daniel Khashabi, Xinxi Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, Sameer Singh, and Yejin Choi. Prompt waywardness: The curious case of discretized interpretation of continuous prompts. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 3631–3643. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.NAACL-MAIN.266. URL https://doi.org/10.18653/v1/2022.naacl-main.266.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5562–5571. PMLR, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348. PMLR, 2020. URL http://proceedings.mlr.press/v119/koh20a.html.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 3045–3059. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.243. URL https://doi.org/10.18653/v1/2021.emnlp-main.243.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.353. URL https://doi.org/10.18653/v1/2021.acl-long.353.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 61–68. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-SHORT.8. URL https://doi.org/10.18653/v1/2022.acl-short.8.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *AI Open*, 5:208–215, 2024a. doi: 10.1016/J.AIOPEN.2023.08.012. URL https://doi.org/10.1016/j.aiopen.2023.08.012.

Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V. Chawla. Can we soft prompt llms for graph learning tasks? In Tat-Seng Chua, Chong-Wah Ngo, Roy Ka-Wei Lee, Ravi Kumar, and Hady W. Lauw (eds.), *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, pp. 481–484. ACM, 2024b. doi: 10.1145/3589335.3651476. URL https://doi.org/10.1145/3589335.3651476.

Jiaqi Mu and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HkuGJ3kCb.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Seyed Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *CoRR*, abs/2402.05862, 2024. doi: 10.48550/ARXIV.2402.05862. URL https://doi.org/10.48550/arXiv.2402.05862.

Gilles Pisier. Remarques sur un résultat non publié de b. maurey. *Séminaire d'analyse fonctionnelle*, pp. 1–12, 1980.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. *CoRR*, abs/2510.06377, 2025. doi: 10.48550/ARXIV.2510.06377. URL https://doi.org/10.48550/arXiv.2510.06377.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh V. Chawla, and Chao Huang. A survey of large language models for graphs. In Ricardo Baeza-Yates and Francesco Bonchi (eds.), *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pp. 6616–6626. ACM, 2024. doi: 10.1145/3637528.3671460. URL https://doi.org/10.1145/3637528.3671460.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. Relbench: A benchmark for deep learning on relational databases. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/25cd345233c65fac1fec0ce61d0f7836-Abstract-Datasets_and_Benchmarks_Track.html.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *CoRR*, abs/2006.10637, 2020. URL https://arxiv.org/abs/2006.10637.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam (eds.), *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pp. 593–607. Springer, 2018. doi: 10.1007/978-3-319-93417-4_38. URL https://doi.org/10.1007/978-3-319-93417-4_38.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 4222–4235. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.346. URL https://doi.org/10.18653/v1/2020.emnlp-main.346.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Inf. Fusion*, 81:84–90, 2022. doi: 10.1016/J.INFFUS.2021.11.011. URL https://doi.org/10.1016/j.inffus.2021.11.011.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Kumar Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Schärli, Aakanksha Chowdhery, Philip Andrew Mansfield, Blaise Agüera y Arcas, Dale R. Webster, Gregory S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle K. Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *CoRR*, abs/2212.13138, 2022. doi: 10.48550/ARXIV.2212.13138. URL https://doi.org/10.48550/arXiv.2212.13138.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. SAINT: improved neural networks for tabular data via row attention and contrastive pre-training. *CoRR*, abs/2106.01342, 2021. URL https://arxiv.org/abs/2106.01342.

Alex Tamkin, Mohammad Taufeeque, and Noah D. Goodman. Codebook features: Sparse and discrete interpretability for neural networks. *CoRR*, abs/2310.17230, 2023. doi: 10.48550/ARXIV.2310.17230. URL https://doi.org/10.48550/arXiv.2310.17230.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (eds.), *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pp. 491–500. ACM, 2024. doi: 10.1145/3626772.3657775. URL https://doi.org/10.1145/3626772.3657775.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances*

in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 6306–6315, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

Ruiyu Wang, Zifeng Wang, and Jimeng Sun. Unipredict: Large language models are universal tabular predictors. *CoRR*, abs/2310.03266, 2023. doi: 10.48550/ARXIV.2310.03266. URL https://doi.org/10.48550/arXiv.2310.03266.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous graph attention network. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 2022–2032. ACM, 2019. doi: 10.1145/3308558.3313562. URL https://doi.org/10.1145/3308558.3313562.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a graph-centric relational database foundation model. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/wang25da.html.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a00548031e4647b13042c97c922fadf1-Abstract-Conference.html.

Fang Wu, Vijay Prakash Dwivedi, and Jure Leskovec. Large language models are good relational learners. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 7835–7854. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.386. URL https://doi.org/10.18653/v1/2025.acl-long.386.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David S. Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *CoRR*, abs/2303.17564, 2023. doi: 10.48550/ARXIV.2303.17564. URL https://doi.org/10.48550/arXiv.2303.17564.

Marek Wydmuch, Lukasz Borchmann, and Filip Gralinski. Tackling prediction tasks in relational databases with llms. *CoRR*, abs/2411.11829, 2024. doi: 10.48550/ARXIV.2411.11829. URL https://doi.org/10.48550/arXiv.2411.11829.

Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rJeW1yHYwH.

Linjie Xu and David Wipf. Parameter-free encoders remain viable for RDB foundation models. *CoRR*, abs/2607.05476, 2026. doi: 10.48550/ARXIV.2607.05476. URL https://doi.org/10.48550/arXiv.2607.05476.

Linjie Xu, Yanlin Zhang, Quan Gan, Minjie Wang, and David Wipf. No need to train your RDB foundation model. *CoRR*, abs/2602.13697, 2026. doi: 10.48550/ARXIV.2602.13697. URL https://doi.org/10.48550/arXiv.2602.13697.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and

Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8413–8426. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.745. URL https://doi.org/10.18653/v1/2020.acl-main.745.

Yiwen Yuan, Zecheng Zhang, Xinwei He, Akihiro Nitta, Weihua Hu, Manan Shah, Blaz Stojanovic, Shenyang Huang, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. Contextgnn: Beyond two-tower recommendation systems. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=nzOD1we8Z4.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 16795–16804. IEEE, 2022a. doi: 10.1109/CVPR52688.2022.01631. URL https://doi.org/10.1109/CVPR52688.2022.01631.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *Int. J. Comput. Vis.*, 130(9):2337–2348, 2022b. doi: 10.1007/S11263-022-01653-1. URL https://doi.org/10.1007/s11263-022-01653-1.

APPENDIX

A NOTATION

Table 4 collects the symbols of Sections 3–5 together with the few additional symbols introduced in this appendix (marked with †).

Table 4: **Notation.** Symbols marked † appear only in the appendix; others follow the main text.

Symbol	Meaning
Relational data and graph encoder	
$\mathcal{D} = (\mathcal{T}, \mathcal{L})$	relational database with tables $\mathcal{T} = \{T_1, \dots, T_n\}$ and primary–foreign-key links $\mathcal{L}$
$r = (k_r, \mathcal{K}_r, x_r, t_r)$	row with primary key, foreign keys, attribute values, and timestamp
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi)$	relational entity graph; ϕ maps a node to its table, ψ an edge to its link
(v_s, t_s, y_s), S	training triple (seed entity, seed time, label) and number of triples
$\mathcal{G}_{v_s,t_s}$, N	temporally sampled subgraph of the seed and the number of graph-prompt positions
f_θ, $\mathbf{h}_u \in \mathbb{R}^{d_g}$	heterogeneous GNN encoder and the representation of node u
Frozen LLM	
$\mathcal{M}$, $\mathcal{V}_{\text{tok}}$	frozen decoder-only LLM and its vocabulary
$\mathbf{E} \in \mathbb{R}^{\lvert\mathcal{V}_{\text{tok}}\rvert \times d_\ell}$	frozen input token-embedding table; d_ℓ is the LLM width
$\text{emb}(\cdot)$	lookup of textual tokens in $\mathbf{E}$
Schema Anchoring	
$\mathcal{A}_{\text{meta}}$, $d(\cdot)$	schema-metadata set and deterministic description template
m_j, $(w_{j,1}, \dots, w_{j,L_j})$	j-th metadata entry and its tokenisation
$\mathbf{a}_j$, $\mathbf{A} \in \mathbb{R}^{M \times d_\ell}$	j-th semantic anchor (mean token embedding) and the stacked anchor matrix
$\mathcal{E}(T) \subseteq [M]^\dagger$	eligible anchors of table T (anchors whose metadata mention T; Appendix B.2)
Query-Conditioned Retention	
$\mathbf{G} = [\mathbf{g}_0; \dots; \mathbf{g}_{N-1}]$	graph prompt from the GNN and projector; $\mathbf{g}_0$ is the seed
$\bar{\mathbf{g}}$, $\bar{\mathbf{g}}_{\text{nbr}}$	global mean and neighbour mean of $\mathbf{G}$
$\mathbf{c}^\dagger$, $\bar{\mathbf{e}}^\dagger_{\mathcal{I}}$	conditioning vector and mean embedding of the task description and question
$\eta_{\text{c}}, \eta_{\text{v}}^\dagger$	the two trainable projection networks that score views against $\mathbf{c}$
β_v, $\widetilde{\mathbf{g}}_u$	instance-dependent view weights and retained graph vector of position u
Adaptive Composition and prediction	
q_ω, $\mathbf{z}_u \in \mathbb{R}^M$	query network and anchor-compatibility scores of node u
τ, $\boldsymbol{\alpha}_u \in \Delta^{M-1}$	assignment temperature and soft anchor assignment
$\mathbf{s}_u = \boldsymbol{\alpha}_u^\top \mathbf{A}$	semantic vector (anchor composition) of node u
$\alpha = \sigma(a)$	learnable scalar gate shared across nodes
$\mathbf{p}_u$, $\mathbf{P}_{v_s,t_s}$	graph prompt token $(1-\alpha)\widetilde{\mathbf{g}}_u + \alpha \mathbf{s}_u$ and the stacked prompt
$\mathbf{s}_{\text{G}}$	neighbourhood-level semantic token (Appendix B.4)
$\mathcal{I}$, $\mathbf{X}_{v_s,t_s}$	textual task instruction and the full LLM input sequence
$\mathbf{o}_{v_s,t_s}$, $\rho_\xi = (\mathbf{w}, b)$	last-position hidden state and task head
Theory	
$\mathcal{S} = \text{conv}(\{\mathbf{a}_j\})$	anchor hull: convex hull of the schema anchors, spanned by embedded schema text (Appendix F.4)
$D_{\mathcal{A}}$	anchor diameter $\max_{i,j} \lVert\mathbf{a}_i - \mathbf{a}_j\rVert_2$
F, $L_{\mathcal{S}}$, L_{glob}	prompt-to-output map and its Lipschitz constants on $\mathcal{S}$ and on a bounded region containing all reachable prompts
k, $\lVert\cdot\rVert_0$	number of active anchors and the number of non-zero entries
$\text{dist}(\mathbf{x}, \mathcal{S})^\dagger$	Euclidean distance from $\mathbf{x}$ to the (closed, convex) set $\mathcal{S}$

B Additional Method Details

At a glance. This section spells out every component of RelAnchor at the level of detail needed to re-implement it: the end-to-end procedure (Algorithms 1 and 2), the exact schema-metadata templates and the resulting anchor dictionaries (Appendix B.2), the equations of Query-Conditioned Retention (Appendix B.3), the table-aware anchor assignment and the neighbourhood semantic token $\mathbf{s}_G$ (Appendix B.4), the layout of the LLM input (Appendix B.5), the computational overhead (Appendix B.6), and how RelAnchor relates to other ways of building prompts from structured data (Appendix B.7).

B.1 End-to-End Procedure

Algorithm 1 summarizes how RelAnchor is built, pretrained, and deployed, and Algorithm 2 details the forward pass for a single training triple. Three properties of the procedure are worth emphasising. *(i)* The only database-specific, non-learned object is the anchor matrix $\mathbf{A}$, which is computed once per database from its schema, in a single embedding lookup and without any forward pass through the LLM. *(ii)* Every learned component (the GNN, the query and retention networks, the gate, and the task head) is agnostic to the number M of anchors, because assignment is a softmax over whichever dictionary is supplied; the same weights can therefore be applied to a database whose schema was never seen during training simply by swapping in that database's anchor matrix. *(iii)* The LLM and its embedding table are never updated, yet gradients flow *through* the frozen LLM into the prompt tokens and from there into all trainable modules.

Algorithm 1 RelAnchor: anchor construction, pretraining, and deployment

Require: frozen LLM $\mathcal{M}$ with embedding table $\mathbf{E}$; source databases $\mathcal{D}_{\text{src}}$ with their training tasks; target database $\mathcal{D}^\star$ with target task; temperature τ; update budget U; validation interval Δ
Ensure: trained parameters $\Theta = (\theta, \omega, a, \xi)$ and test predictions on the target task

// Stage 0: Schema Anchoring (once per database, no training)
for each database $\mathcal{D} \in \mathcal{D}_{\text{src}} \cup \{\mathcal{D}^\star\}$ **do**
  build the train-visible snapshot of $\mathcal{D}$ (rows up to the validation cutoff) and the metadata set $\mathcal{A}_{\text{meta}}(\mathcal{D})$ with the templates of Table 5
  $\mathbf{a}_j \leftarrow \frac{1}{L_j}\sum_{l=1}^{L_j} \mathbf{E}[w_{j,l}]$ for every entry m_j (Eq. 3); stack into $\mathbf{A}_{\mathcal{D}}$
  record the eligible sets $\mathcal{E}(T)$ for every table T of $\mathcal{D}$ (Eq. 8)
end for
// Stage 1: leave-one-database-out pretraining on $\mathcal{D}_{\text{src}}$ (no target labels)
initialise Θ; set a such that $\sigma(a) = 0.1$
for $u = 1, \ldots, U$ **do**
  draw a source task on some $\mathcal{D} \in \mathcal{D}_{\text{src}}$ and a mini-batch $\mathcal{B}$ of its training triples
  $\mathcal{L} \leftarrow \frac{1}{|\mathcal{B}|}\sum_{(v_s,t_s,y_s)\in\mathcal{B}} \ell_{\text{task}}\big(\textsc{Forward}(v_s, t_s; \mathbf{A}_{\mathcal{D}}),\, y_s\big)$ (Eq. 6)
  update Θ with AdamW on $\nabla_\Theta \mathcal{L}$ $\{\mathcal{M}, \mathbf{E}, \mathbf{A}$ stay frozen$\}$
  every Δ updates: evaluate on validation data, keep the best checkpoint, stop early on stagnation
end for
// Stage 2: deployment on the target task
if zero-shot regime **then**
  predict $\hat{y} = \textsc{Forward}(v, t; \mathbf{A}_{\mathcal{D}^\star})$ for every test pair with the pretrained Θ
else
  fine-tune Θ on the target training triples as in Stage 1 (with $\mathbf{A}_{\mathcal{D}^\star}$); select the validation-best checkpoint; predict on the test split once
end if

B.2 Schema Anchoring in Detail

Metadata templates. The template $d(\cdot)$ of Eq. 2 is instantiated for six kinds of schema elements, listed in Table 5. Table and column entries name the schema element; primary-key and foreign-key entries additionally state its role in the relational structure; join entries name each pair of tables connected by at least one key link; and statistic entries describe each column's data type

Algorithm 2 FORWARD$(v_s, t_s; \mathbf{A})$: one RelAnchor prediction

Require: seed entity v_s, seed time t_s, anchor matrix $\mathbf{A}$ of the seed's database, task instruction $\mathcal{I}$

1: sample $\mathcal{G}_{v_s,t_s}$ by temporal neighbour sampling, keeping only nodes with $t_u < t_s$
2: $\mathbf{H}_{v_s,t_s} \leftarrow f_\theta(\mathcal{G}_{v_s,t_s})$ (Eq. 1)
3: $\mathbf{G} = [\mathbf{g}_0; \ldots; \mathbf{g}_{N-1}]$: projected embeddings of the seed ($u = 0$) and its sampled one-hop neighbours
4: **// Query-Conditioned Retention (Appendix B.3)**
5: $\mathbf{c} \leftarrow \frac{1}{3}(\bar{\mathbf{e}}_{\mathcal{I}} + \mathbf{g}_0 + \bar{\mathbf{g}})$; $\boldsymbol{\beta} \leftarrow \mathrm{softmax}_v\big(\langle \eta_{\mathrm{v}}(\boldsymbol{\sigma}^{(v)}), \eta_{\mathrm{c}}(\mathbf{c})\rangle / \sqrt{d_r}\big)$
6: $\widetilde{\mathbf{g}}_u \leftarrow \beta_1 \mathbf{g}_u + \beta_2 \bar{\mathbf{g}} + \beta_3 \bar{\mathbf{g}}_{\mathrm{nbr}}$ for $u \geq 1$; $\widetilde{\mathbf{g}}_0 \leftarrow \mathbf{g}_0$
7: **// Adaptive Composition (Section 4.3)**
8: **for** each prompt position $u = 0, \ldots, N-1$ **do**
9: $\quad \mathbf{z}_u \leftarrow \mathbf{A}\, q_\omega(\mathbf{h}_u)/\sqrt{d_\ell}$; $\boldsymbol{\alpha}_u \leftarrow$ softmax of $\mathbf{z}_u/\tau$ over $\mathcal{E}(\phi(u))$ (Eq. 12)
10: $\quad \mathbf{s}_u \leftarrow \boldsymbol{\alpha}_u^\top \mathbf{A}$; $\mathbf{p}_u \leftarrow (1 - \sigma(a))\, \widetilde{\mathbf{g}}_u + \sigma(a)\, \mathbf{s}_u$
11: **end for**
12: $\mathbf{s}_{\mathrm{G}} \leftarrow \boldsymbol{\alpha}_{\mathrm{G}}^\top \mathbf{A}$ with $\boldsymbol{\alpha}_{\mathrm{G}} = \mathrm{softmax}\big(\mathbf{A}\, q_\omega^{\mathrm{G}}(\bar{\mathbf{g}})/(\sqrt{d_\ell}\,\tau)\big)$ (Eq. 13)
13: **// Prompt assembly and prediction (Section 4.4)**
14: $\mathbf{X} \leftarrow [\mathbf{e}_{\mathrm{BOS}} \,\|\, \mathbf{p}_0, \ldots, \mathbf{p}_{N-1} \,\|\, \mathbf{s}_{\mathrm{G}} \,\|\, \mathrm{emb}(\mathcal{I}) \,\|\, \mathbf{e}_{\mathrm{EOT}}]$ (Eq. 5)
15: $\mathbf{o} \leftarrow$ last-position final hidden state of $\mathcal{M}(\mathbf{X})$
16: **return** $\hat{y} = \sigma(\mathbf{w}^\top \mathbf{o} + b)$ (classification) or $\hat{y} = \mathbf{w}^\top \mathbf{o} + b$ (regression)

and its uniqueness, missingness, and (for numeric columns) quantile profile. All descriptions are generated deterministically from the schema and a train-visible snapshot of the data, without any manual writing and without any language model. Each entry receives a unique identifier (e.g. `fk::results::driverId::drivers::driverId`), so that duplicate schema elements yield a single anchor.

Table 5: **Schema-metadata templates.** ⟨*T*⟩, ⟨*c*⟩: table and column names as they appear in the database; ⟨*T'*⟩, ⟨*c'*⟩: referenced table and its primary key. The last column gives the number of entries each element contributes.

Anchor type	Template	Example (F1)	Count
Table descriptions $\{d(T_i)\}$			
Table	`table ⟨T⟩`	`table drivers`	1 per table
Foreign key	`foreign key from ⟨T⟩ ⟨c⟩ to ⟨T'⟩ ⟨c'⟩`	`foreign key from results driverId to drivers driverId`	1 per key
Join path	`join path ⟨T⟩ to ⟨T'⟩`	`join path drivers to results`	1 per linked pair
Column descriptions $\{d(c)\}$			
Column	`column ⟨T⟩ ⟨c⟩`	`column drivers nationality`	1 per column
Primary key	`primary key ⟨T⟩ ⟨c⟩`	`primary key drivers driverId`	1 per table with a key
Statistic: type	`column ⟨T⟩ ⟨c⟩ has dtype ⟨type⟩`	`column drivers dob has dtype time`	1 per column
Statistic: uniqueness	`column ⟨T⟩ ⟨c⟩ has bucketed uniqueness`	`column drivers code has bucketed uniqueness`	1 per column
Statistic: missingness	`column ⟨T⟩ ⟨c⟩ has bucketed missingness`	`column drivers code has bucketed missingness`	1 per column
Statistic: quantiles	`column ⟨T⟩ ⟨c⟩ has bucketed quantiles`	`column results points has bucketed quantiles`	1 per numeric column

Train-visible statistics. Statistic entries are computed on a snapshot of the database that contains only rows with timestamps up to the validation cutoff of the benchmark, so no validation or test rows,

and no task labels, enter the anchors. The data type is one of *integer, float, boolean, categorical, string, time*. The uniqueness ratio (distinct non-null values over rows) and the missingness ratio (null values over rows) are discretised with the thresholds 0.01, 0.05, 0.2, and 0.5 into five buckets (*very low* to *very high*), and the quantile profile of a numeric column codes its 10th, 50th, and 90th percentiles by sign and order of magnitude (negative, zero, $(0, 1)$, $[1, 10)$, ≥ 10). These buckets enter the entry identifier; the description itself names the column and the statistic, so that a statistic anchor lies close to, but is distinct from, the anchor of its column. Because only coarse, schema-level descriptors are used and no cell value is ever written into a description, the anchors cannot reveal individual records.

Anchor embedding. Each description is tokenised with the LLM tokenizer without special tokens and embedded with the frozen input-embedding table; the anchor is the mean over its tokens (Eq. 3) and is stored in single precision. Each anchor is thus the mean-pooled embedding of a real schema description, a point on the semantic manifold in the operative sense of Appendix F.4, and together the anchors span the anchor hull $\mathcal{S}$ on which Theorem 1 operates. The embedding is a lookup, not a forward pass: building the anchors of the largest database in our study (RateBeer, $M = 1{,}054$) takes one batched lookup over roughly 10^4 tokens.

Table-aware eligibility. A row of table T should be described by the concepts of its own table and of the relations it participates in, not by the columns of unrelated tables. We therefore restrict the assignment of a node u with $\phi(u) = T$ to the eligible set

$$\mathcal{E}(T) = \{\, j \;:\; m_j \text{ describes } T, \text{ a column, key, or statistic of } T, \text{ or a foreign key or join path incident to } T \,\}, \tag{8}$$

and fall back to all M anchors if $\mathcal{E}(T)$ is empty. Eligibility is a fixed, parameter-free mask derived from the entry identifiers; it is kept unchanged in every anchor-dictionary control of the ablation study (Appendix E.1), so that those controls change only the anchor *vectors* and never which slots a row may use.

Worked example: the 37 eligible anchors of a `drivers` row in F1

The F1 database has nine tables, 67 columns, and 13 foreign keys, which yield $M = 358$ anchors (Table 6). A row of the entity table `drivers`, the seed of all three F1 tasks, may compose over the following $|\mathcal{E}(\texttt{drivers})| = 37$ of them:

Table (1)	`table drivers`
Columns (7)	`column drivers {driverId, driverRef, code, forename, surname, dob, nationality}`
Primary key (1)	`primary key drivers driverId`
Statistics (22)	`column drivers ⟨c⟩ has dtype ⟨type⟩`, `... has bucketed uniqueness`, `... has bucketed missingness` for all seven columns, and `column drivers driverId has bucketed quantiles`
Foreign keys (3)	`foreign key from {qualifying, results, standings} driverId to drivers driverId`
Join paths (3)	`join path drivers to {qualifying, results, standings}`

A `results` neighbour of the same seed composes over a different set, which includes the result-specific columns such as `grid`, `points`, `laps`, and `position`, and its foreign keys to `races` and `constructors`. One seed's prompt thus mixes several table-specific vocabularies, one per position.

Dictionary sizes. Table 6 and Figure 8 report the anchor dictionaries of all nine databases. M ranges from 75 (Amazon) to 1,054 (RateBeer) and grows linearly with the number of schema elements, not with the number of rows, as stated in Section 4.1. Statistic anchors form the largest group (64–77% of each dictionary), followed by column anchors. Eligibility shrinks the effective dictionary of an entity row to between 13 (Amazon `customer`) and 181 (RateBeer `beers`) anchors.

B.3 Query-Conditioned Retention in Detail

Section 4.2 describes retention in words; we give the exact computation here. Let $\mathbf{G} = [\mathbf{g}_0; \ldots; \mathbf{g}_{N-1}] \in \mathbb{R}^{N \times d_\ell}$ be the graph prompt, with the seed at position zero and its sampled one-hop neighbours at positions $1, \ldots, N-1$.

Table 6: **Anchor dictionaries of the nine databases**, computed from the released database schemas with the templates of Table 5. The number of table and column anchors equals the number of tables and columns. The last column gives $|\mathcal{E}(T)|$ for the entity tables of the evaluated tasks.

	Database	Table	Column	PK	Stat.	FK	Join	M	Eligible anchors of entity tables
RelBenchV1	Amazon	3	15	2	51	2	2	75	`customer` 13, `product` 30
	Avito	8	42	5	162	11	11	239	`UserInfo` 33, `AdsInfo` 41
	Event	5	131	2	512	7	6	663	`users` 42
	F1	9	67	9	247	13	13	358	`drivers` 37
	H&M	3	37	2	130	2	2	176	`customer` 36, `article` 115
	Stack	7	52	7	182	12	10	270	`users` 47, `posts` 65
	Trial	15	140	15	466	15	15	666	`studies` 143, `facilities` 29
V2	ArXiv	6	21	3	74	6	5	115	`papers` 39, `authors` 17
	RateBeer	13	221	12	766	21	21	1054	`beers` 181, `users` 93, `brewers` 158

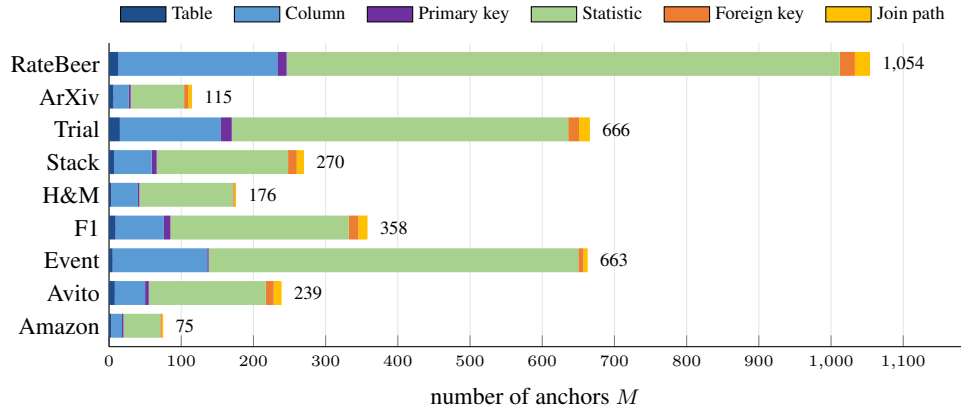

Figure 8: **Composition of the anchor dictionaries.** Number of anchors of each type per database; the total M is printed at the end of each bar. The dictionary grows with the schema, not the data.

Views and view summaries. Retention mixes three views of the same length N, each paired with a single summary vector that is used for scoring:

$$\underbrace{\mathbf{V}^{(1)} = \mathbf{G}}_{\text{local}},\ \boldsymbol{\sigma}^{(1)} = \mathbf{g}_0; \qquad \underbrace{\mathbf{V}^{(2)} = \mathbf{1}_N \bar{\mathbf{g}}^\top}_{\text{global}},\ \boldsymbol{\sigma}^{(2)} = \bar{\mathbf{g}}; \qquad \underbrace{\mathbf{V}^{(3)} = \mathbf{1}_N \bar{\mathbf{g}}_{\text{nbr}}^\top}_{\text{neighbour}},\ \boldsymbol{\sigma}^{(3)} = \bar{\mathbf{g}}_{\text{nbr}}, \tag{9}$$

with $\bar{\mathbf{g}} = N^{-1} \sum_i \mathbf{g}_i$ and $\bar{\mathbf{g}}_{\text{nbr}} = (N-1)^{-1} \sum_{i>0} \mathbf{g}_i$.

Conditioning vector. Let $\bar{\mathbf{e}}_{\mathcal{I}}$ be the mean input embedding of the tokens of the task description and question (the same text that forms the instruction $\mathcal{I}$). The conditioning vector combines task semantics with instance-level graph context,

$$\mathbf{c} = \tfrac{1}{3}\left(\bar{\mathbf{e}}_{\mathcal{I}} + \mathbf{g}_0 + \bar{\mathbf{g}}\right) \in \mathbb{R}^{d_\ell}. \tag{10}$$

Scoring and mixing. Two trainable projection networks $\eta_{\text{c}}, \eta_{\text{v}} : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_r}$ (two-layer perceptrons with a sigmoid non-linearity and $d_r = 256$) embed the conditioning vector and the view summaries in a common scoring space; a scaled dot product and a softmax yield the view weights:

$$\beta_v = \frac{\exp(\lambda_v)}{\sum_{v'=1}^{3} \exp(\lambda_{v'})}, \qquad \lambda_v = \frac{\langle \eta_{\text{v}}(\boldsymbol{\sigma}^{(v)}),\, \eta_{\text{c}}(\mathbf{c}) \rangle}{\sqrt{d_r}}, \qquad \widetilde{\mathbf{g}}_u = \begin{cases} \mathbf{g}_0, & u = 0, \\ \beta_1 \mathbf{g}_u + \beta_2 \bar{\mathbf{g}} + \beta_3 \bar{\mathbf{g}}_{\text{nbr}}, & u \geq 1. \end{cases} \tag{11}$$

Retention is the identity when the seed has no sampled neighbour ($N = 1$).

Interpretation. Eq. 11 is an instance- and task-adaptive *shrinkage* of every neighbour token towards two neighbourhood means: $\beta_1 \to 1$ keeps the tokens local and distinct, whereas large β_2 or β_3 smooths them towards the context shared by the neighbourhood. Because the weights depend on the task text, the same subgraph can be read locally for one task and contextually for another. The seed token is always kept intact, so the identity of the entity being predicted is never averaged away. The four retention controls of Section 6.3 correspond to fixing $\boldsymbol{\beta} = (1, 0, 0)$ (local only), fixing $\boldsymbol{\beta} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ (uniform), dropping $\bar{\mathbf{e}}_{\mathcal{I}}$ from Eq. 10 (no task text), and replacing $(\lambda_v)_v$ by one learned vector per task (task-shared).

B.4 Adaptive Composition and the Neighbourhood Semantic Token

Table-aware soft assignment. With the eligible sets of Eq. 8, the soft assignment of Eq. 4 is computed over the anchors that the node's table may use,

$$[\boldsymbol{\alpha}_u]_j = \frac{\mathbb{1}[j \in \mathcal{E}(\phi(u))] \exp\big([\mathbf{z}_u]_j/\tau\big)}{\sum_{j' \in \mathcal{E}(\phi(u))} \exp\big([\mathbf{z}_u]_{j'}/\tau\big)}, \qquad \mathbf{z}_u = \frac{\mathbf{A}\, q_\omega(\mathbf{h}_u)}{\sqrt{d_\ell}}, \tag{12}$$

which is Eq. 4 with the scores of ineligible anchors set to $-\infty$. The assignment therefore still lies on the probability simplex Δ^{M-1}, and $\mathbf{s}_u = \boldsymbol{\alpha}_u^\top \mathbf{A}$ still lies in the anchor hull $\mathcal{S} = \mathrm{conv}(\mathbf{A})$, so all results of Section 5 apply unchanged. The query network q_ω is a two-layer perceptron (hidden width 1,024, sigmoid activation, dropout). A small temperature ($\tau \in [0.03, 0.2]$ in our experiments) makes the assignment effectively sparse; Proposition 3 quantifies how closely such a soft assignment is approximated by a k-sparse one.

Gate. The shared gate $\alpha = \sigma(a)$ is initialised at $\sigma(a) = 0.1$, so training starts close to the retained graph prompt and learns how much semantic grounding to inject. Because $\mathbf{s}_u \in \mathcal{S}$, the gate contracts every prompt token towards the anchor hull by a factor $1 - \alpha$ (Proposition 4).

Neighbourhood semantic token. In addition to the N per-node tokens, the prompt contains one neighbourhood-level token $\mathbf{s}_{\mathrm{G}}$ (Eq. 5) that summarises the whole sampled neighbourhood in the anchor vocabulary. It is computed from the mean $\bar{\mathbf{g}}$ of the graph prompt with a separate lightweight query network $q_\omega^{\mathrm{G}} : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_\ell}$ (a layer-normalised linear map) and, since a neighbourhood spans several tables, over the full dictionary:

$$\mathbf{s}_{\mathrm{G}} = \boldsymbol{\alpha}_{\mathrm{G}}^\top \mathbf{A}, \qquad \boldsymbol{\alpha}_{\mathrm{G}} = \mathrm{softmax}\left(\frac{\mathbf{A}\, q_\omega^{\mathrm{G}}(\bar{\mathbf{g}})}{\sqrt{d_\ell}\,\tau}\right) \in \Delta^{M-1}. \tag{13}$$

Unlike the node tokens, $\mathbf{s}_{\mathrm{G}}$ is not gated, so it is an exact point of the anchor hull $\mathcal{S}$: a fully on-manifold token that tells the LLM which schema concepts the neighbourhood as a whole is about. The "neighbourhood token off" ablation of Section 6.3 replaces it by the zero vector.

Gradient flow. The anchor matrix is a constant, so no gradient ever changes it. The loss reaches q_ω and q_ω^{G} through the assignment weights, the gate parameter a through the convex mixture, the retention networks through the view weights, and the GNN through all of these paths and through the retained graph vectors. The hard-assignment control of the ablation study uses the straight-through estimator $\boldsymbol{\alpha}_u^{\mathrm{hard}} = \mathrm{onehot}(\arg\max_j [\boldsymbol{\alpha}_u]_j) - \mathrm{sg}(\boldsymbol{\alpha}_u) + \boldsymbol{\alpha}_u$, where sg stops gradients.

B.5 Prompt Assembly

Figure 9 shows the complete LLM input of Eq. 5. The instruction $\mathcal{I}$ is the concatenation of a one-sentence task description and a question, in the fixed format

⟨*task description*⟩ `Question:` ⟨*question*⟩ `Answer:`

followed by the end-of-turn token; the begin-of-text and end-of-turn tokens are the native special tokens of the backbone (`<|begin_of_text|>` and `<|eot_id|>`). Representative instructions are shown in the box at the end of this subsection. The instruction contains no label, no example, and no cell value: all information about the entity enters through the graph prompt. Sequences in a mini-batch are left-padded, and padded positions are masked out of attention, so the last position is always $\mathbf{e}_{\mathrm{EOT}}$ and its final hidden state is the input of the task head.

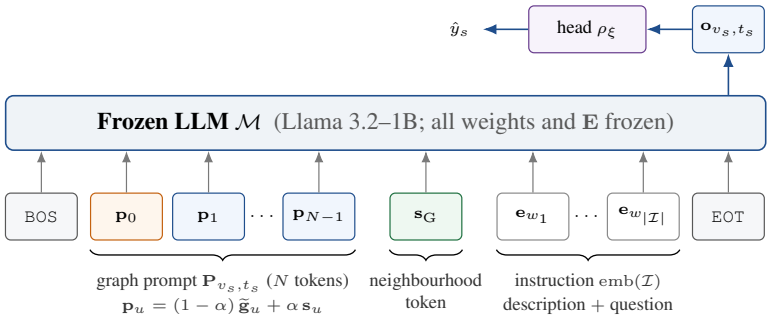

Figure 9: **Layout of the LLM input** (Eq. 5). Seed and neighbour prompt tokens are gated mixtures of retained graph vectors and on-manifold anchor compositions; the neighbourhood token $\mathbf{s}_G$ is a pure anchor composition; the instruction consists of ordinary token embeddings. Only the last-position hidden state $\mathbf{o}_{v_s,t_s}$ is read out, by a linear(-sigmoid) head trained with BCE (classification) or the L_1 loss (regression).

Representative task instructions $\mathcal{I}$ (description + question)

rel-stack/user-badge
```
This task is to predict if a user will receive a new badge in the
next 3 months or not.  Question:  Based on records of activity, will
this user receive a new badge in the next 3 months?  Give Yes or No
as an answer.  Answer:
```
rel-amazon/item-ltv
```
This task is to predict the $ value of the total number purchases and
reviews each product will receive in the next 3 months.  Question:
What is the total dollar value of purchases this product will receive
in the next 3 months?  Provide a float numerical answer.  Answer:
```
rel-arxiv/paper-citation
```
This task is to predict whether a paper will receive at least one
citation in the next 6 months.  Question:  Will this paper receive
at least one citation in the next 6 months?  Give Yes or No as an
answer.  Answer:
```
rel-ratebeer/user-count
```
This task is to predict the number of ratings a user will give in the
next 90 days.  Question:  How many ratings will this user give in the
next 90 days?  Give an integer as an answer.  Answer:
```
Every task uses one fixed instruction of this form; the instructions of all tasks are part of the released code. The answer-format sentence is kept from the instruction format of prior work; since predictions are read out by the task head rather than decoded, it only serves as task context.

B.6 Computational Cost

Anchors. The anchor dictionary is built once per database and reused by every task on it. Its construction is a single embedding lookup over the tokens of M short descriptions, and its storage is $M \times d_\ell$ single-precision numbers, at most $1{,}054 \times 2{,}048 \times 4$ bytes ≈ 8.6 MB for the largest dictionary.

Per-prediction overhead. For each of the N graph-prompt positions, the assignment scores $\mathbf{A}q_\omega(\mathbf{h}_u)$ and the composition $\boldsymbol{\alpha}_u^\top \mathbf{A}$ each cost $2Md_\ell$ floating-point operations, i.e. at most $4 \cdot 1{,}054 \cdot 2{,}048 \approx 8.6$ MFLOPs per position. A forward pass of the 1B-parameter backbone costs about two FLOPs per non-embedding parameter and token, i.e. roughly 2 GFLOPs per position. Composition therefore adds less than 0.5% to the LLM cost of the graph-prompt positions even for the largest dictionary, and about 0.15% for F1. Retention adds a constant number of small perceptron evaluations per prediction, independent of N except for the $O(Nd_\ell)$ averaging and mixing.

Sequence length and memory. The LLM input has $N + |\mathcal{I}| + 3$ positions (BOS, N graph tokens, $\mathbf{s}_G$, $|\mathcal{I}|$ instruction tokens, EOT), exactly one more than the corresponding GNN-as-prefix input

with the same neighbourhood and instruction. Because the LLM is frozen, no gradients or optimizer states are stored for its parameters; the backward pass through the LLM is needed only to propagate gradients to the prompt tokens.

B.7 Relation to Other Prompt-Construction Paradigms

Table 7 positions RelAnchor against the three families of prompt construction discussed in Section 2. Text serialization is on-manifold and schema-aware because it literally writes the data as text, but it is bounded by the context window and cannot be trained on large label sets. Free projections (GNN-as-prefix) scale and are row-specific, but their outputs are unconstrained points of $\mathbb{R}^{d_\ell}$. Hard assignments to tokens or codebook entries are on-manifold but discrete, so rows that share a nearest entry collapse to the same prompt. RelAnchor is the only paradigm that satisfies all five properties at once: its anchors are on-manifold and database-specific by construction, and its gated, continuous, row-specific mixtures keep every prompt anchored to named schema concepts near the manifold while keeping distinct rows distinct (Proposition 5).

Table 7: **Prompt-construction paradigms for LLMs over relational data.** ✓: satisfied by construction; ∼: only partially or not by design; ✗: not satisfied.

Paradigm	Anchored prompts	Database-specific semantics	Continuous, row-specific	Multi-hop context beyond window	Trainable on large label sets
Text serialization (e.g. TabLLM, serialized RelBench context)	✓	✓	∼	✗	✗
Free projection (e.g. G-Retriever, GraphToken, Rel-LLM)	✗	✗	✓	✓	✓
Hard assignment (e.g. nearest token, codebook)	✓	✗	✗	✓	✓
RelAnchor (learned assignment over schema anchors)	✓	✓	✓	✓	✓

C Experimental Details

At a glance. This section gives the full experimental setup behind Section 6: the 27 tasks and their metrics (Appendix C.1), the leave-one-database-out pretraining protocol and why it admits no target-task leakage (Appendix C.2), the controls that disentangle the anchoring mechanism from pretraining and from Retention (Appendix C.3), the three evaluation regimes (Appendix C.4), the baselines and the provenance of their numbers (Appendix C.5), and all implementation details and hyperparameters (Appendix C.6).

C.1 Benchmarks and Metrics

We evaluate on all 21 entity-level tasks of RelBenchV1 (Robinson et al., 2024), 12 binary classification tasks and nine regression tasks across the seven databases F1, Avito, Event, Trial, Amazon, Stack Overflow, and H&M, and on six tasks from RelBenchV2: four classification tasks (RateBeer beer, user, and brewer-dormant; ArXiv paper-citation) and two regression tasks (RateBeer user-count; ArXiv author-publication). Table 8 lists every task with its entity table and prediction horizon. Each database is converted into a temporal heterogeneous graph, and all training, validation, and test examples are constructed with the benchmark's temporal splits and leakage-safe temporal sampling, so that no information from after the seed time t_s is ever visible to the model. The tasks span a wide range of schema sizes (3 to 15 tables), interaction patterns, horizons (4 days to one year), and label distributions.

Table 8: **Evaluated tasks.** "Label in tables" gives the column headers of Tables 1, 2, 11, and 12. Cls.: binary classification (AUROC ↑); Reg.: regression (MAE ↓).

	Database	Task	Type	Entity table	Horizon	Label in tables
RelBenchV1	F1	driver-dnf	Cls.	drivers	30 d	f1 / dnf
		driver-top3	Cls.	drivers	30 d	f1 / top3
		driver-position	Reg.	drivers	60 d	f1 / pos
	Avito	user-clicks	Cls.	UserInfo	4 d	avito / click
		user-visits	Cls.	UserInfo	4 d	avito / visit
		ad-ctr	Reg.	AdsInfo	4 d	avito / ctr
	Event	user-repeat	Cls.	users	7 d	event / repeat
		user-ignore	Cls.	users	7 d	event / ignore
		user-attendance	Reg.	users	7 d	event / attend
	Trial	study-outcome	Cls.	studies	365 d	trial / out
		study-adverse	Reg.	studies	365 d	trial / adverse
		site-success	Reg.	facilities	365 d	trial / success
	Amazon	user-churn	Cls.	customer	91 d	amazon / user
		item-churn	Cls.	product	91 d	amazon / item
		user-ltv	Reg.	customer	91 d	amazon / user
		item-ltv	Reg.	product	91 d	amazon / item
	Stack	user-engagement	Cls.	users	91 d	stack / eng
		user-badge	Cls.	users	91 d	stack / badge
		post-votes	Reg.	posts	91 d	stack / votes
	H&M	user-churn	Cls.	customer	7 d	hm / churn
		item-sales	Reg.	article	7 d	hm / sales
RelBenchV2	RateBeer	beer-churn	Cls.	beers	90 d	ratebeer / beer
		user-churn	Cls.	users	90 d	ratebeer / user
		brewer-dormant	Cls.	brewers	365 d	ratebeer / dormant
		user-count	Reg.	users	90 d	ratebeer / user-count
	ArXiv	paper-citation	Cls.	papers	182 d	arxiv / citation
		author-publication	Reg.	authors	182 d	arxiv / publication

Classification metric. Binary classification is measured by the area under the receiver operating characteristic curve (AUROC), reported on a 0–100 scale. The aggregate is the unweighted mean AUROC over the classification tasks of a benchmark.

Regression metrics. Regression is measured by the mean absolute error (MAE) in each task's original units. Because task scales differ by orders of magnitude (e.g. 0.035 on H&M item-sales versus 30.038 on Trial study-adverse for RelAnchor), comparing raw MAE across tasks is not meaningful, and we report two aggregates: (1) Raw Avg., the unweighted mean of the per-task MAEs, which is dominated by the large-scale tasks, and (2) the LightGBM-normalized mean

$$\mu_n = \frac{1}{T}\sum_{t=1}^{T} \frac{\mathrm{MAE}_t}{\mathrm{MAE}_{\mathrm{LightGBM},t}}, \tag{14}$$

which divides each task's MAE by the LightGBM MAE of the same task before averaging and therefore weights all tasks equally. A value $\mu_n < 1$ indicates that a method outperforms LightGBM on average in this scale-invariant sense; LightGBM itself has $\mu_n = 1$ by definition.

C.2 Leave-One-Database-Out Pretraining

RelAnchor uses supervised cross-database pretraining. For each RelBenchV1 target database D, the source pool is $\mathcal{D}_{\mathrm{src}} = \mathcal{D}_{\mathrm{V1}} \setminus \{D\}$, comprising the other six RelBenchV1 databases (Figure 10). All prediction tasks defined on the target database are excluded from gradient-based pretraining, so that no target-task labels, target task instructions, or target-database subgraphs contribute to the pretrained weights. Source tasks of both types are interleaved during pretraining, with each mini-batch drawn from a single source task and processed with the anchor dictionary of that task's database.

For RelBenchV2 targets (RateBeer and ArXiv), we pretrain exclusively on all seven RelBenchV1 databases, and no RelBenchV2 database is included in the pretraining pool. This is a stricter transfer setting than leave-one-out within a single benchmark, because the schema and data distribution of the RelBenchV2 databases are entirely unseen during pretraining.

In both cases, a single pretrained checkpoint per source pool is used. The pretrained model is either evaluated in the zero-shot regime or fine-tuned on the target task in the task-specific regime.

	Pretraining (source) databases							
Target database	F1	Avito	Event	Trial	Amazon	Stack	H&M	**#sources**
RelBenchV1: leave one database out								
F1	×	•	•	•	•	•	•	6
Avito	•	×	•	•	•	•	•	6
Event	•	•	×	•	•	•	•	6
Trial	•	•	•	×	•	•	•	6
Amazon	•	•	•	•	×	•	•	6
Stack	•	•	•	•	•	×	•	6
H&M	•	•	•	•	•	•	×	6
RelBenchV2: unseen benchmark								
RateBeer	•	•	•	•	•	•	•	7
ArXiv	•	•	•	•	•	•	•	7

Figure 10: **Source pools of leave-one-database-out pretraining.** •: pretraining database; ×: held-out target database, none of whose tasks, labels, instructions, or subgraphs are used for pretraining.

How a pretrained model is applied to an unseen database. Three design choices make the pretrained weights directly applicable to a target database whose schema was never seen. First, the anchor matrix $\mathbf{A}$ is recomputed from the target schema (Appendix B.2) and swapped in; since assignment is a softmax over whichever dictionary is supplied, the query network needs no change when M changes. Second, the task is specified by its natural-language instruction $\mathcal{I}$, which the frozen LLM reads with its pre-trained knowledge. Third, the task head is shared by all tasks of the same type (one head for binary classification, one for regression), so a head exists for every target task. The zero-shot regime evaluates exactly this model; the task-specific regime starts fine-tuning from it.

Leakage safeguards. *(i)* No target-database task, label, instruction, or subgraph is used for pretraining. *(ii)* Anchors are built from a train-visible snapshot of the database (rows up to the validation cutoff) and contain only schema names and coarse statistic descriptors, never labels or cell values. *(iii)* Temporal neighbour sampling keeps only rows with $t_u < t_s$. *(iv)* Checkpoints are selected on validation data only, and the selected checkpoint is evaluated on the test split once.

C.3 Disentangling Anchoring from Pretraining and from Retention

The task-specific results of Tables 1 and 2 fine-tune RelAnchor from a leave-one-database-out checkpoint (Appendix C.2), whereas Rel-LLM and the GNN baselines are trained from scratch on the single target task with their published numbers. One might therefore ask whether the margin over Rel-LLM reflects the cross-database pretraining, or the Query-Conditioned Retention module, rather than the anchoring mechanism itself. Two controls address the two candidate confounds directly.

RelAnchor without pretraining. Our experiments show that RelAnchor trained directly on the target task achieves comparable final performance to RelAnchor initialized by LODO pretraining, while pretraining accelerates convergence during target-task training. The performance advantage over Rel-LLM is preserved under training from scratch. These results indicate that the gains remain attainable using target-task supervision alone, with cross-database initialization primarily improving the efficiency of target-task optimization in the settings examined.

Rel-LLM with Retention. Conversely, grafting the Retention module of Section 4.2 onto Rel-LLM yields no performance gain. A plausible explanation follows from the geometry of view mix-

ing. Retention combines each graph token with shared neighbourhood summaries; for two non-seed tokens of the same neighbourhood, their difference after mixing is scaled by the local-view weight (Eq. 11),

$$\widetilde{\mathbf{g}}_i - \widetilde{\mathbf{g}}_j = \beta_1 \, (\mathbf{g}_i - \mathbf{g}_j), \tag{15}$$

so the operation aggregates context while contracting distinctions between tokens. In Rel-LLM, such smoothing provides no explicit constraint linking the resulting representations to schema concepts in the frozen LLM embedding space, so it can attenuate useful variation without a compensating alignment benefit. In RelAnchor, the retained context also guides the assignments over the fixed schema anchors, allowing neighbourhood information to influence the relative emphasis on identifiable semantic concepts. The contrasting results are consistent with a benefit from coupling context aggregation with this semantic reference.

Anchoring remains consequential under the same architecture. The dictionary and composition controls of the ablation study (Tables 15 and 16) provide converging evidence: those variants keep Retention, the pretraining protocol, and every other component fixed, and degrading the anchor dictionary or replacing the learned soft assignment still reduces performance throughout. The anchoring choices therefore remain consequential within the same Retention architecture, independently of the two factors above.

C.4 Evaluation Regimes

We organize baselines and our own results into three evaluation regimes, following the conventions of prior work on RelBench; Table 9 summarizes which supervision each regime permits.

Zero-shot. The target database is excluded from relational model training, and no labeled target-task examples are used at inference. The model must generalize to unseen schemas and tasks using only the pretrained representations. This regime tests the transferability of the learned relational and semantic representations.

Few-shot / ICL. The target database is also excluded from training, but labeled target-task examples are provided as inference-time context. Model parameters remain fixed; the labeled examples guide the prediction through in-context learning or a nearest-neighbour mechanism. This regime tests whether labeled examples can compensate for the lack of task-specific training.

Task-specific. Predictors are fitted, or task-specific statistics are computed, using the target-task training labels. This includes fine-tuning a pretrained model on the target task, training a GNN from scratch, and fitting a gradient-boosted tree on engineered features. This is the standard supervised regime and provides the strongest baselines.

Table 9: **Supervision available in each evaluation regime.**

Regime	Trained on target database	Target labels for training	Target labels at inference	Methods in Tables 1–2 and 11–12
Zero-shot	✗	✗	✗	Rel-Zero, Griffin, RT_{zero}, $\text{RELANCHOR}_{\text{zero}}$
Few-shot / ICL	✗	✗	✓	TabPFN-2.5, RDBLearn (v1.1 on RelBenchV2)
Task-specific	✓	✓	–	Global / Entity Median, LightGBM, DS+LightGBM, GraphSAGE, HGT, HGT_{PE}, RelGNN, RelGT, Rel-LLM, RELANCHOR

C.5 Baselines

We compare against four baseline families, retaining the supervision and evaluation settings reported in their source studies.

Tabular and statistical baselines. Global Median and Entity Median provide non-learned reference points for regression: the former predicts the median training label of the task, the latter the median training label of the same entity. LightGBM (Ke et al., 2017) is a gradient-boosted tree trained on single-table features. The "DS+" prefix denotes human-engineered feature pipelines using deep feature synthesis (Kanter & Veeramachaneni, 2015), paired with LightGBM, AutoGluon (Erickson et al., 2020), or TabPFN-2.5 (Grinsztajn et al., 2025), following Hudovernik et al. (2026). These pipelines represent the state of the art for flat tabular prediction on relational data and serve as strong baselines that do not model cross-table structure end-to-end.

Relational deep learning. GraphSAGE (Hamilton et al., 2017) is the standard GNN baseline on RelBench. HGT and HGT_{PE} (Hu et al., 2020) are heterogeneous graph transformers without and with positional encodings. RelGNN (Chen et al., 2025) introduces composite message passing over atomic routes. RelGT (Dwivedi et al., 2025) is a relational graph transformer. These methods learn structural representations end-to-end but do not incorporate pre-trained language-model knowledge.

Foundation models and ICL pipelines. Griffin (Wang et al., 2025) and RT_{zero} (Ranjan et al., 2025) are relational foundation models. TabPFN-2.5 (Grinsztajn et al., 2025) is a tabular prior-data fitted network. RDBLearn (Xu et al., 2026; Xu & Wipf, 2026), including v1.1 on RelBenchV2, is an automated relational prediction pipeline that consumes labeled target examples at inference time.

Language-model baselines. Llama-3.2-1B and Llama-3.2-3B with an MLP head (Wydmuch et al., 2024; Wu et al., 2025) evaluate the LLM backbone used by RelAnchor with an alternative integration strategy. Rel-Zero and Rel-LLM (Wu et al., 2025) couple a heterogeneous GNN over temporally sampled subgraphs with a frozen LLM via a learned projection; Rel-LLM is the closest prior work and the paradigm RelAnchor departs from.

C.6 Implementation Details

Model architecture. We use Llama 3.2–1B as the frozen LLM backbone. The schema anchors are computed once per database from the frozen input-embedding table and remain fixed throughout training. The trainable components are: (1) the GNN relational encoder and MLP projector, (2) the assignment query networks, (3) the retention scoring networks, (4) the shared scalar gate, and (5) the task prediction heads. Table 10 lists all architectural and optimisation settings.

Training. Training uses binary cross-entropy for classification and the L_1 loss for regression, as in Eq. 6, with no auxiliary objective. Pretraining uses at most 128,000 optimizer updates; fine-tuning uses the same update budget. Evaluation subsets are fixed using sampling seed 48.

Model selection and evaluation. We evaluate validation performance every 2,000 optimizer updates for checkpoint selection and stop early after three consecutive evaluations without improvement. The selection metric is AUROC for classification and MAE for regression. The validation-best checkpoint for each task is evaluated on the test split to obtain the reported metrics. In the zero-shot regime, the pretrained checkpoint is evaluated directly without any task-specific fine-tuning. In the task-specific regime, the pretrained model is fine-tuned on the target task, and the best validation checkpoint from fine-tuning is used for test evaluation.

Reproducibility checklist. All benchmarks are public (RelBenchV1 and RelBenchV2). Every run is fully specified by the task, the regime, and the released per-task hyperparameter configuration. Anchors are deterministic functions of the schema and the train-visible snapshot, and the task instructions are fixed strings. Results in Tables 1–2 and 11–12 for RelAnchor report the test performance of the validation-selected checkpoint. The code, the anchor-construction script, the per-task configurations, and the task instructions are available at `https://github.com/nmksjx/RelAnchor`; its entry point runs both pretraining and fine-tuning and implements the validation-selected protocol used here.

Table 10: **Implementation details and hyperparameters.** Per-task values (learning rate, weight decay, batch size, dropout, temperature) are tuned on the validation split; the table gives the values or ranges used across tasks.

Component	Setting
Frozen backbone	
LLM $\mathcal{M}$	Llama 3.2–1B: 16 decoder layers, width $d_\ell = 2{,}048$, vocabulary of 128,256 tokens; weights stored in half precision, bfloat16 autocast; no parameter of $\mathcal{M}$ or $\mathbf{E}$ is updated
Relational encoder	
Column encoders	per-type tabular encoders (PyTorch Frame); text columns embedded with a frozen sentence encoder (all-mpnet-base-v2); missing numerical values mean-imputed
GNN f_θ	heterogeneous message passing (RelBench reference architecture) with relative temporal encoding; 2 layers, hidden width $d_g = 128$, sum aggregation
Neighbour sampling	temporal, $t_u < t_s$; 128 neighbours per relation and hop, most recent first; graph prompt: seed and its sampled one-hop neighbours
Projector	two-layer perceptron $d_g \to 1{,}024 \to d_\ell$, sigmoid, dropout
RelAnchor modules	
Anchors	templates of Table 5; mean input embedding; $M \in [75, 1{,}054]$ (Table 6); frozen
Query networks	q_ω: two-layer perceptron (hidden width 1,024, sigmoid); q_ω^{G}: layer-normalised linear map
Temperature τ	0.07 by default; tuned per task in $\{0.03, 0.07, 0.1, 0.2\}$
Retention networks	$\eta_{\mathrm{c}}, \eta_{\mathrm{v}}$: two-layer perceptrons $d_\ell \to 256 \to 256$, sigmoid; three views
Gate $\alpha = \sigma(a)$	one scalar per model, initialised at $\sigma(a) = 0.1$
Task heads	one binary-classification head and one regression head, shared across tasks of the same type
Optimisation and selection	
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.95)$; no weight decay on biases and normalisation parameters
Learning rate / weight decay	5×10^{-5} to 3×10^{-3} / 0.01 to 0.1, per task
Batch size / dropout	2 or 8 / 0.1 or 0.4, per task
Update budget	at most 128,000 optimizer updates (pretraining and fine-tuning)
Validation	every 2,000 updates; early stopping after 3 evaluations without improvement
Evaluation subsets	fixed with sampling seed 48
Software and hardware	
Libraries	PyTorch Geometric 2.6.1, Transformers 4.45.2, Sentence-Transformers 3.2.0, RelBench and PyTorch Frame (vendored)
Hardware	a single GPU per run

D Results on RelBenchV2

At a glance. RelBenchV2 contributes two databases, RateBeer and ArXiv, that are absent from the pretraining pool, so this section tests transfer to *entirely unseen schemas*. Zero-shot RelAnchor already exceeds task-specific LightGBM on the mean AUROC (79.63 versus 76.78) and on μ_n (0.7636 versus 1.0000); task-specific RelAnchor attains the best mean AUROC (86.43) and the best raw MAE (3.6061) of all displayed methods and improves over GraphSAGE on all six tasks.

This appendix reports the RelBenchV2 results that Section 6.2 refers to and provides a more detailed, per-task discussion than space permits in the main text. Tables 11 and 12 give the classification and regression results, respectively, and Figure 11 plots both tables per task.

Setup. RelBenchV2 contributes two databases that are absent from RelBenchV1: RateBeer, a beer-rating community with beers, brewers, users, and ratings, and ArXiv, a scholarly database with papers, authors, and citations. We evaluate four binary classification tasks (RateBeer beer, user, and brewer-dormant; ArXiv paper-citation) and two regression tasks (RateBeer user-count; ArXiv author-publication). The evaluation protocol, metrics, and baselines follow Section 6.1. Two points

deserve emphasis. First, RelBenchV2 targets are evaluated with models pretrained exclusively on the seven RelBenchV1 databases, so for RelAnchor$_{\text{zero}}$ both the schema and the data distribution of RateBeer and ArXiv are entirely unseen; this is a stricter transfer setting than the leave-one-database-out protocol of RelBenchV1, where the held-out database at least shares the benchmark's construction conventions. Second, the available baselines differ from RelBenchV1: RDBLearn v1.1 is the only published few-shot / ICL result on RelBenchV2, and it consumes labeled target-task examples at inference time, so comparisons across regimes should be read with these supervision differences in mind.

RelBenchV2. Tables 11 and 12 report results on RateBeer and ArXiv, neither of which appears in our pretraining pool. RelAnchor$_{\text{zero}}$ attains a mean classification AUROC of 79.63, exceeding task-specific LightGBM by 2.85 points, and its regression $\mu_n = 0.7636$ is a 23.6% reduction relative to LightGBM. Task-specific RelAnchor achieves the highest mean classification AUROC of 86.43, versus 86.24 for RDBLearn and 83.99 for GraphSAGE, with the best scores on brewer-dormant and paper-citation. On regression, it improves over GraphSAGE on both tasks, attains the lowest RateBeer user-count MAE of 6.7199, and achieves the lowest raw average MAE of 3.6061, compared with 3.6645 for RDBLearn and 3.9435 for GraphSAGE.

Table 11: **Classification results on RelBenchV2.** AUROC is reported in percent (higher is better), and Avg is the unweighted mean over the four tasks. **Bold** and underline mark the best and second-best distinct scores across all displayed methods; rows are ours.

Method	ratebeer			arxiv	**Avg**
	beer	user	dormant	citation	↑
Zero-shot					
RELANCHOR$_{\text{zero}}$	72.56	88.74	75.82	81.41	79.63
Few-shot / ICL					
RDBLearn (v1.1)	**83.56**	**97.84**	81.18	82.37	86.24
Task-specific					
LightGBM	76.21	83.92	75.79	71.21	76.78
GraphSAGE	78.67	94.27	80.51	82.50	83.99
RELANCHOR	83.24	94.96	**83.66**	**83.86**	**86.43**

Table 12: **Regression results on RelBenchV2.** MAE is reported in original units (lower is better). Raw Avg. is the unweighted mean over the two tasks, and μ_n is the mean LightGBM-normalized MAE. **Bold** and underline mark the best and second-best distinct scores across all displayed methods; rows are ours.

Method	ratebeer	arxiv	**Raw Avg.**	μ_n
	user-count	publication	↓	↓
Zero-shot				
RELANCHOR$_{\text{zero}}$	10.588	0.581	5.5845	0.7636
Few-shot / ICL				
RDBLearn (v1.1)	6.870	**0.459**	3.6645	**0.5665**
Task-specific				
Global Median	15.124	0.577	7.8505	0.8716
Entity Median	13.079	0.874	6.9765	1.0787
LightGBM	20.350	0.577	10.4635	1.0000
GraphSAGE	7.374	0.513	3.9435	0.6257
RELANCHOR	**6.7199**	0.49225	**3.6061**	0.5917

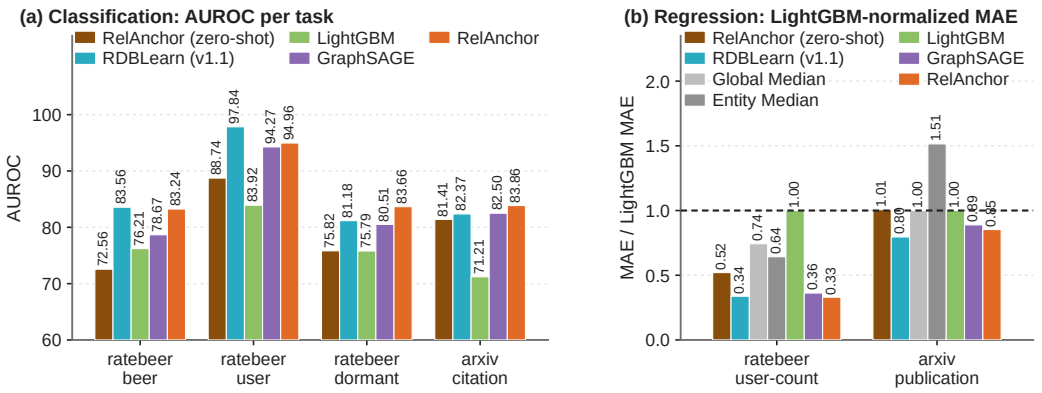

Figure 11: **RelBenchV2 per task.** (a) Test AUROC of every displayed method on the four classification tasks. (b) Test MAE divided by the LightGBM MAE of the same task on the two regression tasks; the dashed line is LightGBM.

Classification in detail. The zero-shot advantage over task-specific LightGBM is not uniform across tasks (Table 11). It is largest on ArXiv paper-citation, where RelAnchor$_{\text{zero}}$ reaches 81.41 AUROC against 71.21 for LightGBM, a gain of 10.20 points, and on RateBeer user, where it reaches 88.74 against 83.92, a gain of 4.82 points. On brewer-dormant the two are essentially tied (75.82 versus 75.79), and on RateBeer beer LightGBM retains a 3.65-point lead (76.21 versus 72.56). The two tasks on which the zero-shot model excels are precisely those whose labels depend on multi-hop relational context (citations accumulated through the paper–author–paper structure, and user activity aggregated over ratings), whereas the beer task is comparatively well served by flat per-entity features. Zero-shot RelAnchor also trails the task-specific graph baseline GraphSAGE on every task, by 6.11 points on beer, 5.53 on user, 4.69 on brewer-dormant, and 1.09 on paper-citation, which quantifies how much target-database supervision is still worth on these schemas.

Fitting RelAnchor with target labels closes this gap and reverses it. The zero-shot to task-specific improvement is 6.80 mean AUROC points (79.63 to 86.43), with the largest per-task gains on beer (10.68 points), brewer-dormant (7.84), and user (6.22), and the smallest on paper-citation (2.45), the task on which the zero-shot model was already strong. Against GraphSAGE, task-specific RelAnchor is better on all four tasks, by 4.57 points on beer, 0.69 on user, 3.15 on brewer-dormant, and 1.36 on paper-citation, for a mean improvement of 2.44 points. Against RDBLearn v1.1, the comparison is split: RDBLearn is ahead on beer (83.56 versus 83.24, a 0.32-point margin) and more clearly on user (97.84 versus 94.96), while RelAnchor is ahead on brewer-dormant (83.66 versus 81.18) and paper-citation (83.86 versus 82.37). RelAnchor wins the aggregate by 0.19 points (86.43 versus 86.24) and holds the best or second-best score in every column, which is the most consistent profile among the displayed methods.

Regression in detail. The regression tasks are on very different scales (Table 12): RateBeer user-count errors are in the tens, whereas ArXiv author-publication errors are below one. The normalized score μ_n therefore weights the two tasks equally, while the raw average is dominated by user-count. A notable feature of this benchmark is that task-specific LightGBM is a weak reference on user-count: its MAE of 20.350 is worse than both the Global Median (15.124) and the Entity Median (13.079), which is why μ_n values below one are easy to obtain on this task. The opposite holds on author-publication, where the Global Median (0.577) equals LightGBM and the Entity Median (0.874) is substantially worse, indicating that an author's own history is a poor predictor of future publication count.

RelAnchor$_{\text{zero}}$ reduces the user-count MAE from LightGBM's 20.350 to 10.588, a 48.0% reduction that also beats both median baselines, but on author-publication its 0.581 is marginally worse than the Global Median and LightGBM (0.577). Its μ_n of 0.7636 thus comes almost entirely from the user-count task. Task-specific RelAnchor attains the lowest user-count MAE among all displayed methods, 6.7199, which is 8.9% below GraphSAGE (7.374) and 2.2% below RDBLearn v1.1

(6.870). On author-publication it reaches 0.49225, 4.0% below GraphSAGE (0.513) and second only to RDBLearn (0.459). These two tasks explain the difference between the two aggregate columns: RelAnchor has the lowest raw average (3.6061, versus 3.6645 for RDBLearn and 3.9435 for GraphSAGE) because it is best on the large-scale user-count task, whereas RDBLearn has the lower μ_n (0.5665 versus 0.5917) because its advantage on author-publication, which carries equal weight after normalization, outweighs its deficit on user-count. Relative to LightGBM, task-specific RelAnchor reduces μ_n by 40.8%, and relative to its own zero-shot variant it reduces the user-count MAE by a further 36.5% (10.588 to 6.7199) and the author-publication MAE by 15.3% (0.581 to 0.49225).

Summary. Across the six RelBenchV2 tasks, RelAnchor transfers to two databases whose schemas were never seen during pretraining, matching or exceeding a task-specific tabular learner in the zero-shot regime on four of the six tasks, and, once fitted with target labels, it improves over the task-specific graph baseline on every task and reaches the best aggregate classification AUROC and the best raw regression MAE among all displayed methods. The remaining gaps to RDBLearn v1.1 on RateBeer user and ArXiv author-publication are the two cases in which inference-time labeled examples are most beneficial, and they identify a natural direction for combining schema-anchored prompts with in-context supervision.

E Complete Ablation Results

At a glance. This section reports every ablation variant of Section 6.3 on all ten tasks. Appendix E.1 defines each control precisely, Table 14 condenses all 19 controls into their mean scores, and the following subsections give, for each design question, a figure with the complete per-task picture, a table with the exact numbers, and an analysis. **Every one of the 19 controls is worse than full RelAnchor on both aggregates**: each loses between 4.50 and 6.96 mean AUROC points and raises mean MAE by between 5.9% and 23.3%.

All variants follow the training and model-selection protocol of the main experiments, and the RelAnchor column reuses the values of Tables 1 and 2. The ten tasks are the six classification tasks (F1 driver-dnf and driver-top3, Stack user-engagement and user-badge, Amazon user-churn and item-churn) and the four regression tasks (F1 driver-position, Stack post-votes, Amazon user-ltv and item-ltv) of F1, Stack, and Amazon. Figures 12 to 17 and Tables 15 to 21 follow the order of the six design questions of the main text; the retention figure covers the view-mixing and the conditioning controls, which have one table each.

E.1 Ablation Protocol

Each control changes exactly one mechanism and keeps every other component, the training recipe, the update budget, and the selection protocol unchanged. Table 13 lists what each control replaces, in the notation of Appendix B.

Table 13: **Definition of the ablation controls.** Everything not mentioned in the last column is identical to full RelAnchor.

Design question	Control	What is replaced
Schema Anchoring	Generic vocab.	$\mathbf{A} \to$ embeddings of M randomly sampled non-special alphabetic tokens, each rescaled to the norm of the anchor it replaces (on-manifold, database-agnostic)
	Rotated anchors	$\mathbf{A} \to \mathbf{A}_{\text{rot}} = \mathbf{AQ}$ with a random orthogonal $\mathbf{Q}$ (all norms and inner products preserved; Proposition 6)
Names & statistics	W/o names	table and column identifiers in all metadata descriptions anonymized
	W/o statistics	statistical-description phrases removed from the metadata
Adaptive Composition	Uniform mixture	$\boldsymbol{\alpha}_u \to$ uniform distribution over $\mathcal{E}(\phi(u))$
	Table-shared	$\mathbf{z}_u \to$ one learned logit vector per table, independent of $\mathbf{h}_u$
	Hard (ST)	$\boldsymbol{\alpha}_u \to$ one-hot top anchor with a straight-through gradient
Semantic pathways	Node off	gate fixed to $\alpha = 0$, i.e. $\mathbf{p}_u = \tilde{\mathbf{g}}_u$
	Nbr. token off	$\mathbf{s}_{\text{G}} \to \mathbf{0}$
	Both off	both of the above
Shared gate	$\alpha \in \{0, \dots, 1\}$	learnable $a \to$ constant with $\sigma(a)$ equal to the listed value ($\alpha = 0$ is "node off")
Query-Conditioned Retention	Local only	$\boldsymbol{\beta} \to (1, 0, 0)$, i.e. $\widetilde{\mathbf{G}} = \mathbf{G}$
	Uniform mix	$\boldsymbol{\beta} \to (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$
	W/o task text	$\mathbf{c} \to \frac{1}{2}(\mathbf{g}_0 + \bar{\mathbf{g}})$ in Eq. 10; the LLM's instruction is unchanged
	Task-shared	$(\lambda_v)_v \to$ one learned logit vector per task, independent of the instance

E.2 Schema Anchoring

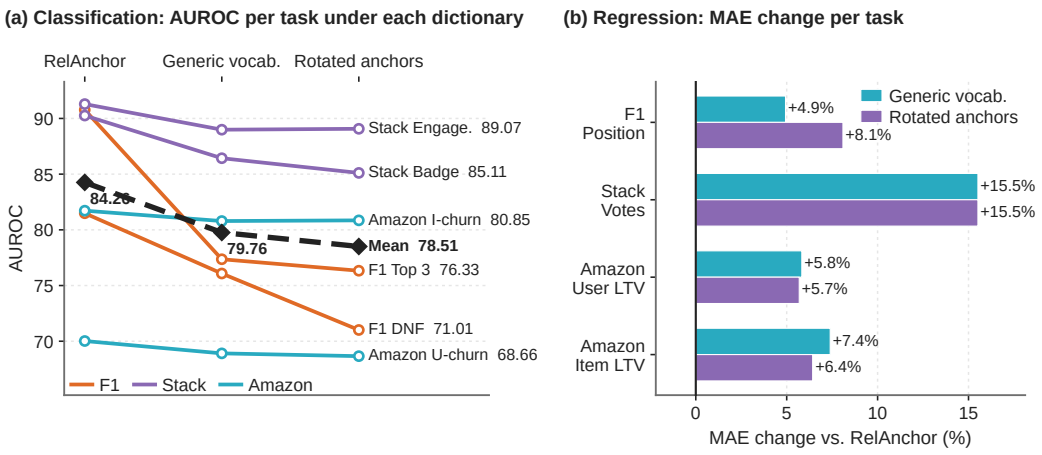

Figure 12: **Schema Anchoring, all tasks.** (a) Test AUROC of each classification task under the schema anchors, generic vocabulary embeddings, and rotated anchors; the dashed line is the mean over the six tasks. (b) Change of test MAE relative to RelAnchor on each regression task (positive: higher error).

Classification. Both frozen dictionaries lower AUROC on all six tasks (Figure 12a, Table 15). Generic vocabulary costs 4.50 mean points and rotated anchors 5.75, but the loss is far from uniform. The two F1 tasks account for most of it: driver-top3 loses 13.41 and 14.44 points under the two controls and driver-dnf loses 5.41 and 10.48, whereas the four Stack and Amazon tasks lose between 0.87 and 5.15 points. The contrast between the two controls is the informative part. Generic

Table 14: **All ablation controls at a glance.** Mean test AUROC over the six classification tasks and mean test MAE over the four regression tasks, with the change relative to full RelAnchor (bars: AUROC loss in points, MAE increase in percent). Per-task values are in Tables 15–21.

Design question	Variant	Mean AUROC ↑	Δ AUROC	Mean MAE ↓	Δ MAE
RELANCHOR (full)		**84.26**	–	**15.9678**	–
Schema Anchoring	Generic vocab.	79.76	−4.50	17.0784	+7.0%
	Rotated anchors	78.51	−5.75	16.9816	+6.3%
Names & statistics	W/o names	78.37	−5.89	17.2803	+8.2%
	W/o statistics	77.92	−6.34	17.3370	+8.6%
Adaptive Composition	Uniform mixture	78.09	−6.17	17.2410	+8.0%
	Table-shared	77.73	−6.53	18.0602	+13.1%
	Hard (ST)	77.85	−6.41	17.1816	+7.6%
Semantic pathways	Node off (α=0)	77.85	−6.41	17.8826	+12.0%
	Nbr. token off	77.87	−6.39	17.7864	+11.4%
	Both off	78.83	−5.43	17.3102	+8.4%
Shared gate	α=0.10	77.30	−6.96	17.8268	+11.6%
	α=0.25	79.08	−5.18	17.5047	+9.6%
	α=0.50	78.36	−5.90	16.9155	+5.9%
	α=0.75	78.38	−5.88	17.6126	+10.3%
	α=1.00	77.50	−6.76	19.6829	+23.3%
Retention	Local only	77.64	−6.62	17.8240	+11.6%
	Uniform mix	78.35	−5.91	17.3305	+8.5%
	W/o task text	77.86	−6.40	17.8768	+12.0%
	Task-shared	78.44	−5.82	16.9075	+5.9%

Table 15: **Schema Anchoring.** Test AUROC (in percent, higher is better) on the six classification tasks and test MAE (original units, lower is better) on the four regression tasks when the schema anchors are replaced by generic vocabulary embeddings or by orthogonally rotated anchors. Mean rows are unweighted over the tasks above them. **Bold** and underline mark the best and second-best distinct values in each row; a row with a single value is a tie.

Database	Task	RelAnchor	Generic vocab.	Rotated anchors
Classification (AUROC ↑)				
F1	driver-dnf	**81.49**	76.08	71.01
F1	driver-top3	**90.77**	77.36	76.33
Stack	user-engagement	**91.29**	88.99	89.07
Stack	user-badge	**90.26**	86.43	85.11
Amazon	user-churn	**70.02**	68.91	68.66
Amazon	item-churn	**81.72**	80.79	80.85
Mean		**84.26**	79.76	78.51
Regression (MAE ↓)				
F1	driver-position	**2.785**	2.922	3.010
Stack	post-votes	**0.058**	0.067	0.067
Amazon	user-ltv	**13.745**	14.547	14.527
Amazon	item-ltv	**47.283**	50.778	50.322
Mean		**15.9678**	17.0784	16.9816

vocabulary tokens lie on the embedding manifold but carry no database semantics; rotated anchors preserve every norm and pairwise inner product of the schema dictionary, so each row's mixture has exactly the geometry it has in RelAnchor, but the dictionary is no longer expressed in the coordinates the LLM has learned to read. Rotation is the more damaging control on four of the six tasks and in the mean, so alignment with the LLM's coordinate system matters at least as much as the distinctness of the anchors. F1 is the most sensitive database: its tables are described by short,

highly specific schema text, and the drops suggest that this text is exploited only when the anchors sit in the LLM's native embedding space.

Regression. The regression tasks move less than the F1 classification tasks but degrade consistently (Figure 12b). Both controls raise the Stack post-votes MAE from 0.058 to 0.067 (15.5%), F1 driver-position worsens by 4.9% with generic vocabulary and 8.1% under rotation, and the two Amazon LTV tasks are 5.7% to 7.4% worse under both controls. Mean MAE rises from 15.9678 to 17.0784 with generic vocabulary and to 16.9816 with rotated anchors, that is, by 7.0% and 6.3%. The regression targets on these databases depend more on the numeric neighborhood statistics that the GNN summarizes than on the identity of the entity, which is consistent with their smaller sensitivity to the anchor dictionary.

> **Take-away.** Both database specificity and on-manifold alignment are necessary: neither on-manifold but generic tokens nor a geometry-preserving rotation of the schema anchors recovers the full model.

E.3 Adaptive Composition

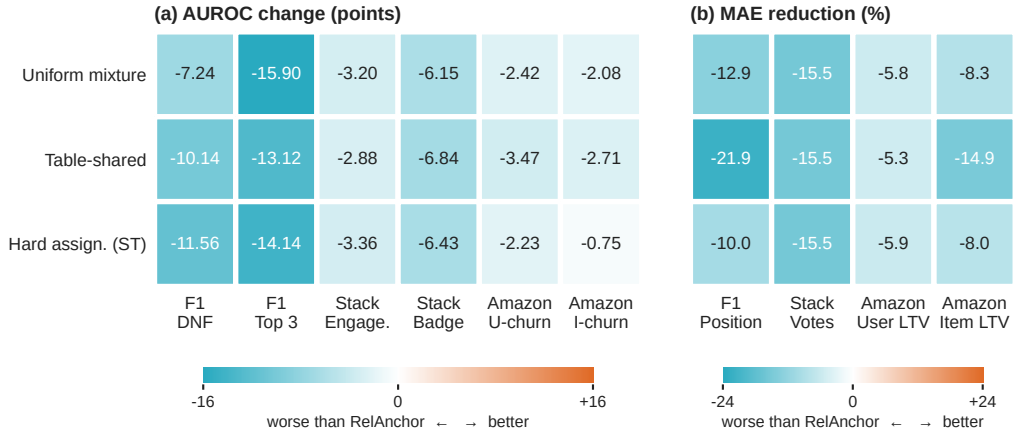

Figure 13: **Adaptive Composition, all tasks.** Change against RelAnchor under each assignment rule: (a) AUROC difference in points and (b) MAE reduction in percent, so that negative values mean a worse score in both panels; cell colors run from teal (worse) to orange (better).

Classification. All three fixed rules trail RelAnchor on every task (Figure 13a, Table 16): the uniform mixture by 2.08 to 15.90 points, the table-shared mixture by 2.71 to 13.12, and hard assignment by 0.75 to 14.14. Their mean losses, 6.17, 6.53, and 6.41 points, are close, but the rules fail on different tasks. The uniform mixture is the worst rule on driver-top3 (-15.90), where equal weight on every eligible anchor blurs the distinction between drivers; hard assignment is the worst rule on driver-dnf (-11.56), where a single anchor cannot represent a row that combines several attributes, yet it is the closest rule on Amazon item-churn (-0.75), where one dominant anchor evidently suffices; and the table-shared mixture is the worst rule on user-badge, user-churn, and item-churn, whose rows are heterogeneous enough that one mixture per table cannot follow them.

Regression. The picture is sharper (Figure 13b). Every rule raises the post-votes MAE from 0.058 to 0.067, and the table-shared mixture is by far the most costly, raising MAE by 21.9% on driver-position and 14.9% on item-ltv and the mean MAE by 13.1%, against 8.0% for the uniform mixture and 7.6% for hard assignment. Removing row specificity therefore hurts regression more than removing softness, whereas classification penalizes both about equally. A learned, row-specific, continuous mixture is the only rule that is best on every task.

Table 16: **Adaptive Composition.** Test AUROC (in percent, higher is better) on the six classification tasks and test MAE (original units, lower is better) on the four regression tasks when the learned row-specific assignment is replaced by a uniform mixture, a table-shared mixture, or hard assignment with a straight-through gradient. Mean rows are unweighted over the tasks above them. **Bold** and underline mark the best and second-best distinct values in each row; a row with a single value is a tie.

Database	Task	RelAnchor	Uniform	Table-shared	Hard (ST)
Classification (AUROC ↑)					
F1	driver-dnf	**81.49**	74.25	71.35	69.93
F1	driver-top3	**90.77**	74.87	77.65	76.63
Stack	user-engagement	**91.29**	88.09	88.41	87.93
Stack	user-badge	**90.26**	84.11	83.42	83.83
Amazon	user-churn	**70.02**	67.60	66.55	67.79
Amazon	item-churn	**81.72**	79.64	79.01	80.97
Mean		**84.26**	78.09	77.73	77.85
Regression (MAE ↓)					
F1	driver-position	**2.785**	3.146	3.394	3.065
Stack	post-votes	**0.058**	0.067	0.067	0.067
Amazon	user-ltv	**13.745**	14.540	14.474	14.549
Amazon	item-ltv	**47.283**	51.211	54.306	51.045
Mean		**15.9678**	17.2410	18.0602	17.1816

Take-away. Only a learned, row-specific, soft mixture is best on every task. Removing row specificity (table-shared) hurts regression most, whereas classification penalises the loss of row specificity and of softness about equally.

E.4 Schema names and statistical descriptors

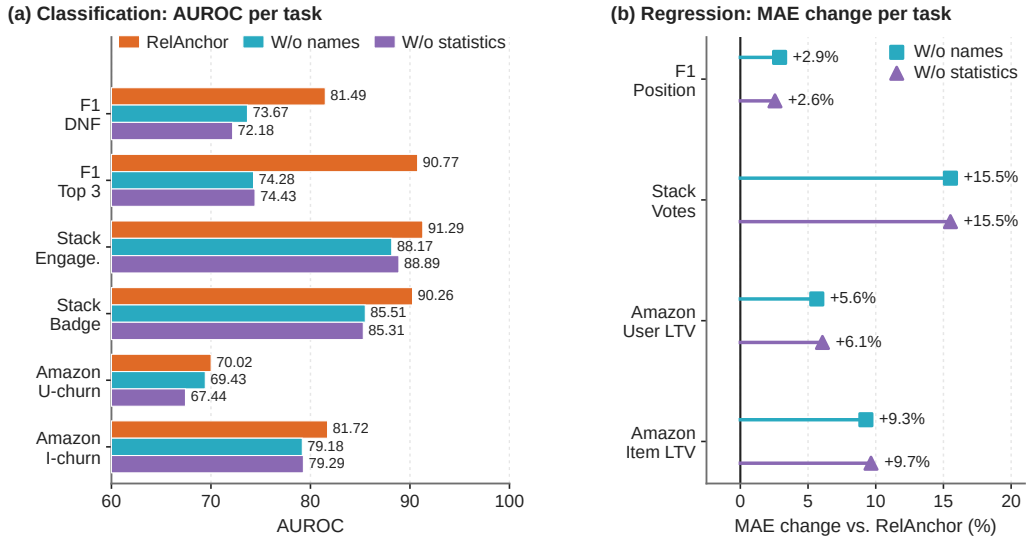

Figure 14: **Schema names and statistical descriptors, all tasks.** (a) Test AUROC of each classification task with the full metadata, anonymized identifiers, and no statistical phrases. (b) Change of test MAE relative to RelAnchor on each regression task.

Classification. Both controls lower AUROC on all six tasks (Figure 14a, Table 17). The losses are largest on F1 driver-top3 (16.49 and 16.34 points) and driver-dnf (7.82 and 9.31), between 2.40 and

Table 17: **Schema names and statistical descriptors.** Test AUROC (in percent, higher is better) on the six classification tasks and test MAE (original units, lower is better) on the four regression tasks when table and column identifiers are anonymized, or the statistical-description phrases are removed. Mean rows are unweighted over the tasks above them. **Bold** and underline mark the best and second-best distinct values in each row; a row with a single value is a tie.

Database	Task	RelAnchor	W/o names	W/o statistics
Classification (AUROC ↑)				
F1	driver-dnf	**81.49**	73.67	72.18
F1	driver-top3	**90.77**	74.28	74.43
Stack	user-engagement	**91.29**	88.17	88.89
Stack	user-badge	**90.26**	85.51	85.31
Amazon	user-churn	**70.02**	69.43	67.44
Amazon	item-churn	**81.72**	79.18	79.29
Mean		**84.26**	78.37	77.92
Regression (MAE ↓)				
F1	driver-position	**2.785**	2.866	2.856
Stack	post-votes	**0.058**	0.067	0.067
Amazon	user-ltv	**13.745**	14.521	14.579
Amazon	item-ltv	**47.283**	51.667	51.846
Mean		**15.9678**	17.2803	17.3370

4.95 points on the two Stack tasks, and moderate on Amazon item-churn (2.54 and 2.43); even on Amazon user-churn, the mildest case, anonymized names cost 0.59 points and dropping the statistics costs 2.58. The two controls are close in the mean, 78.37 against 77.92, which indicates that names and statistical descriptors contribute comparable amounts of signal rather than one dominating the other; both are needed to reach 84.26.

Regression. Both controls raise the post-votes MAE from 0.058 to 0.067, worsen driver-position by 2.9% and 2.6%, and raise the two Amazon LTV errors by 5.6% to 9.7% (Figure 14b); mean MAE grows by 8.2% and 8.6%. The effect is smallest on driver-position, an average of finishing ranks, and largest on the LTV tasks, whose labels aggregate many reviews per entity and thus benefit most clearly from the descriptors; the mean effect follows the LTV tasks.

Take-away. Schema names and statistical descriptors carry comparable and complementary signal; removing either one costs about six mean AUROC points and about 8% mean MAE.

E.5 NODE AND NEIGHBORHOOD SEMANTIC PATHWAYS

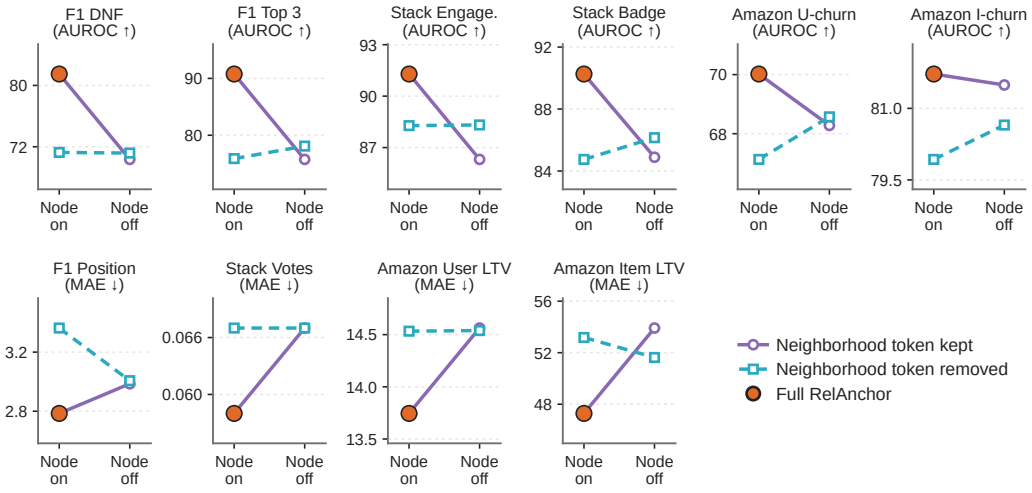

Figure 15: **Node and neighborhood semantic pathways, per task.** Each panel is the 2×2 factorial of one task: the node-injection gate on or off, with the neighborhood semantic token kept (solid) or replaced by zeros (dashed); the orange point is full RelAnchor.

Table 18: **Node and neighborhood semantic pathways.** Test AUROC (in percent, higher is better) on the six classification tasks and test MAE (original units, lower is better) on the four regression tasks when the node-injection gate is zeroed, the neighborhood semantic token is replaced by a zero vector, or both. Mean rows are unweighted over the tasks above them. **Bold** and underline mark the best and second-best distinct values in each row; a row with a single value is a tie.

Database	Task	RelAnchor	Node off	Nbr. off	Both off
Classification (AUROC ↑)					
F1	driver-dnf	**81.49**	70.35	71.26	71.18
F1	driver-top3	**90.77**	75.75	75.89	78.09
Stack	user-engagement	**91.29**	86.31	88.28	88.32
Stack	user-badge	**90.26**	84.89	84.75	86.15
Amazon	user-churn	**70.02**	68.28	67.13	68.57
Amazon	item-churn	**81.72**	81.49	79.93	80.65
Mean		**84.26**	77.85	77.87	78.83
Regression (MAE ↓)					
F1	driver-position	**2.785**	2.986	3.363	3.006
Stack	post-votes	**0.058**	0.067	0.067	0.067
Amazon	user-ltv	**13.745**	14.563	14.533	14.539
Amazon	item-ltv	**47.283**	53.915	53.182	51.629
Mean		**15.9678**	17.8826	17.7864	17.3102

Per-task interactions. Figure 15 shows the 2×2 factorial on each task and Table 18 lists the values. On every task except Stack post-votes, where all three variants tie at 0.067, disabling both pathways is better than the worse of the two single ablations, and on five tasks (driver-top3, user-engagement, user-badge, user-churn, and item-ltv) it is better than both. The two single ablations differ in which pathway matters more: the node path dominates on user-engagement (86.31 with the node path off against 88.28 with the neighborhood token off) and on driver-dnf, whereas the neighborhood token dominates on driver-position (20.7% higher MAE without it, 7.2% without the node path), user-churn, and item-churn. In the mean, the single ablations reach 77.85 and 77.87 AUROC and 17.8826 and 17.7864 MAE, while disabling both reaches 78.83 and 17.3102.

Interpretation. If the pathways contributed independently, removing both would cost roughly the sum of the two single losses; instead it costs less than either. Plausibly, the shared gate and the retention weights are trained to a balance between the two pathways: removing one leaves the other carrying an injection strength that was tuned for the pair, whereas removing both returns the model to a purely GNN-conditioned prompt that is internally consistent. Either way, the two pathways should be evaluated together, and their gain attributed to the pair rather than to each separately.

Take-away. The node and neighbourhood pathways act as a coupled prompt design, and their gain should be credited to the pair rather than to each pathway separately.

E.6 SHARED-GATE SENSITIVITY

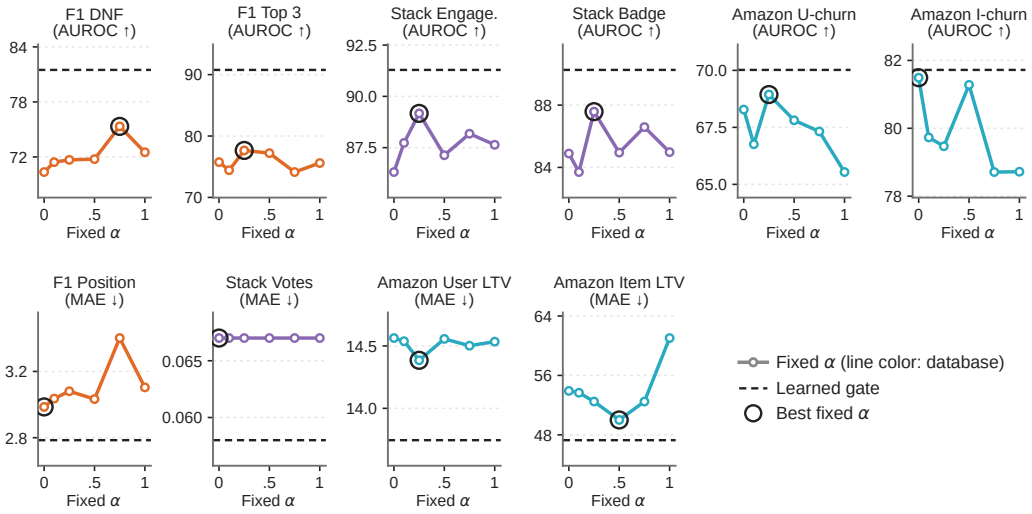

Figure 16: **Shared-gate sensitivity, per task.** Test score of each task with α fixed to 0, 0.10, 0.25, 0.50, 0.75, or 1.00; the dashed line is the learned gate and the ring marks the best fixed value.

Table 19: **Shared-gate sensitivity.** Test AUROC (in percent, higher is better) on the six classification tasks and test MAE (original units, lower is better) on the four regression tasks when the shared gate α is fixed to each listed value instead of being learned. Mean rows are unweighted over the tasks above them. **Bold** and underline mark the best and second-best distinct values in each row; a row with a single value is a tie.

Database	Task	Learned	α=0	0.10	0.25	0.50	0.75	1.00
Classification (AUROC ↑)								
F1	driver-dnf	**81.49**	70.35	71.43	71.68	71.76	75.33	72.51
F1	driver-top3	**90.77**	75.75	74.43	77.65	77.21	74.13	75.60
Stack	user-engagement	**91.29**	86.31	87.73	89.17	87.13	88.18	87.64
Stack	user-badge	**90.26**	84.89	83.70	87.58	84.95	86.59	84.98
Amazon	user-churn	**70.02**	68.28	66.76	68.94	67.81	67.32	65.54
Amazon	item-churn	**81.72**	81.49	79.73	79.47	81.28	78.71	78.72
Mean		**84.26**	77.85	77.30	79.08	78.36	78.38	77.50
Regression (MAE ↓)								
F1	driver-position	**2.785**	2.986	3.036	3.080	3.033	3.401	3.104
Stack	post-votes	**0.058**	0.067	0.067	0.067	0.067	0.067	0.067
Amazon	user-ltv	**13.745**	14.563	14.538	14.385	14.556	14.502	14.534
Amazon	item-ltv	**47.283**	53.915	53.666	52.487	50.006	52.480	61.027
Mean		**15.9678**	17.8826	17.8268	17.5047	16.9155	17.6126	19.6829

Per-task optima. The best fixed value differs across tasks (Figure 16, Table 19): $\alpha = 0.75$ on driver-dnf, $\alpha = 0.25$ on driver-top3, user-engagement, user-badge, user-churn, and user-ltv, $\alpha = 0$ on item-churn and driver-position, and $\alpha = 0.50$ on item-ltv. The curves are non-monotonic on every task: the spread between the best and the worst fixed value is 2.78 to 4.98 AUROC points on the classification tasks and 11.02 MAE on item-ltv, where $\alpha = 1.00$ raises the error from 50.01 (at 0.50) to 61.03. The learned gate beats every fixed value on all ten tasks.

Aggregate. The best fixed value in the mean is $\alpha = 0.25$ for classification (79.08, 5.18 points below the learned gate) and $\alpha = 0.50$ for regression (16.9155, 5.9% above it); the worst, $\alpha = 1.00$, raises mean MAE by 23.3%. Note that $\alpha = 0$ coincides with the node-off variant of Table 18. A single global value is thus always a compromise across tasks, whereas the learned scalar reaches the per-model optimum at no architectural cost.

> **Take-away.** No single fixed gate value is best across tasks, and a wrong value is costly; the learned scalar beats all six fixed values on every task at the cost of one parameter.

E.7 Query-Conditioned Retention

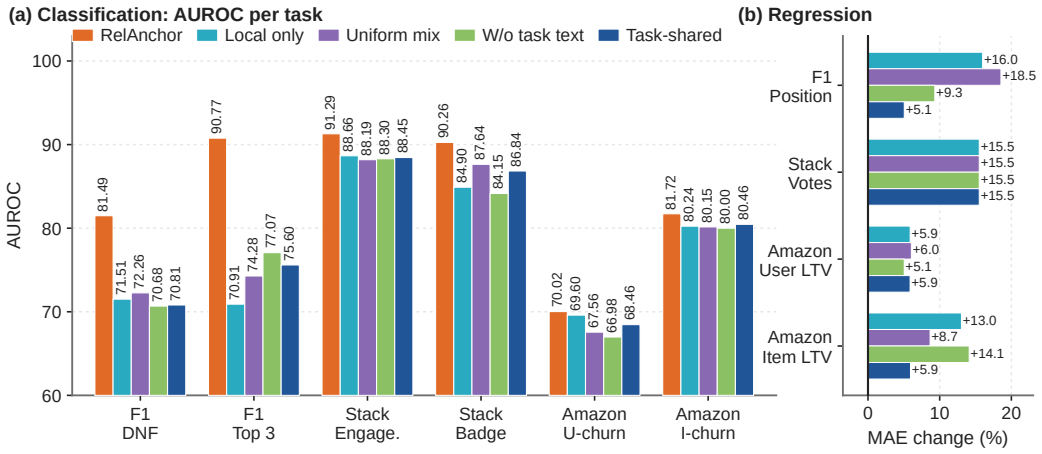

Figure 17: **Query-Conditioned Retention, all tasks.** (a) Test AUROC of each classification task for RelAnchor and the four retention controls. (b) Change of test MAE relative to RelAnchor on each regression task.

Classification. Figure 17 and Tables 20 and 21 cover the four retention controls. All four controls are worse than RelAnchor on all ten tasks. The local-only prompt has a mean loss of 6.62 AUROC points and the largest single drop of the whole ablation suite, 19.86 points on driver-top3. Uniform view weights lose 5.91 mean points. Dropping the task text loses 6.40 mean points, with 10.81 and 13.70 points on the two F1 tasks. Task-shared weights are the mildest control, with a mean loss of 5.82 points.

Regression and interpretation. On regression, task-shared weights lose least (mean MAE +5.9%, driver-position +5.1%), whereas the other three controls add 8.5% to 12.0% to the mean MAE, with individual increases of up to 18.5% (uniform weights on driver-position) and 14.1% (no task text on item-ltv). Two conclusions follow. View mixing itself is the most valuable ingredient: removing it (local only) is the worst control on classification and among the worst on regression. Within the conditioning vector, task semantics matter more than instance-level context: keeping task-level information but sharing it across instances (task-shared) loses least, while dropping the task text costs almost as much as removing the mixing altogether. Query-conditioned weights that see both signals are the best choice on every task.

Table 20: **Local and multi-view context.** Test AUROC (in percent, higher is better) on the six classification tasks and test MAE (original units, lower is better) on the four regression tasks when view mixing is removed (local prompt only) or replaced by equal view weights. Mean rows are unweighted over the tasks above them. **Bold** and underline mark the best and second-best distinct values in each row; a row with a single value is a tie.

Database	Task	RelAnchor	Local only	Uniform mix
Classification (AUROC ↑)				
F1	driver-dnf	**81.49**	71.51	72.26
F1	driver-top3	**90.77**	70.91	74.28
Stack	user-engagement	**91.29**	88.66	88.19
Stack	user-badge	**90.26**	84.90	87.64
Amazon	user-churn	**70.02**	69.60	67.56
Amazon	item-churn	**81.72**	80.24	80.15
Mean		**84.26**	77.64	78.35
Regression (MAE ↓)				
F1	driver-position	**2.785**	3.230	3.301
Stack	post-votes	**0.058**	0.067	0.067
Amazon	user-ltv	**13.745**	14.553	14.577
Amazon	item-ltv	**47.283**	53.446	51.377
Mean		**15.9678**	17.8240	17.3305

Table 21: **Task and instance conditioning.** Test AUROC (in percent, higher is better) on the six classification tasks and test MAE (original units, lower is better) on the four regression tasks when the task-text embedding is dropped from the conditioning vector, or instance-dependent view logits are replaced by a per-task learned vector. Mean rows are unweighted over the tasks above them. **Bold** and underline mark the best and second-best distinct values in each row; a row with a single value is a tie.

Database	Task	RelAnchor	W/o task text	Task-shared
Classification (AUROC ↑)				
F1	driver-dnf	**81.49**	70.68	70.81
F1	driver-top3	**90.77**	77.07	75.60
Stack	user-engagement	**91.29**	88.30	88.45
Stack	user-badge	**90.26**	84.15	86.84
Amazon	user-churn	**70.02**	66.98	68.46
Amazon	item-churn	**81.72**	80.00	80.46
Mean		**84.26**	77.86	78.44
Regression (MAE ↓)				
F1	driver-position	**2.785**	3.045	2.926
Stack	post-votes	**0.058**	0.067	0.067
Amazon	user-ltv	**13.745**	14.439	14.552
Amazon	item-ltv	**47.283**	53.956	50.085
Mean		**15.9678**	17.8768	16.9075

Take-away. View mixing is the single most valuable ingredient of retention, and its weights should be conditioned on both the task text and the instance: every simplification loses on every task.

F Proof Details and Additional Theoretical Results

At a glance. This section supplements Section 5. We state the sparsification lemma behind Theorem 1 in general form (Appendix F.1) and show that its $1/\sqrt{k}$ rate cannot be improved (Appendix F.2); we make the on-manifold Lipschitz constant on the anchor hull precise and explain why it is always finite (Appendix F.3); we fix the formal status of the semantic manifold, in particular why we do not identify it with the convex hull of the token embeddings, whose membership carries no information (Appendix F.4);

we connect the theory, which is stated for k-sparse assignments, to the dense softmax assignments RelAnchor actually learns (Appendix F.5) and to its gated prompt tokens (Appendix F.6); we prove that sparse composition keeps distinct rows distinct (Appendix F.7); we show what the rotated-anchor control of the ablation study isolates (Appendix F.8); and we extend the fidelity bound to multi-token prompts (Appendix F.9).

Throughout, $\|\cdot\|$ is the Euclidean norm, anchors are treated as points of $\mathbb{R}^{d_\ell}$, and for $\boldsymbol{\alpha} \in \Delta^{M-1}$ we write $\boldsymbol{\alpha}^\top \mathbf{A} = \sum_j \alpha_j \mathbf{a}_j$. We use repeatedly that the diameter of $\mathrm{conv}(\mathbf{A})$ equals the diameter $D_{\mathcal{A}}$ of the anchor set, since the distance between two convex combinations is at most the largest distance between the points they combine.

F.1 Maurey Sparsification

The construction in the proof of Theorem 1 is an instance of the following classical lemma, due to Maurey and first published by Pisier (1980).

Lemma 1 (Maurey). *Let $\mathbf{a}_1, \dots, \mathbf{a}_M$ be points in a Hilbert space with diameter $D = \max_{i,j} \|\mathbf{a}_i - \mathbf{a}_j\|$, and let $\mathbf{p}^* = \sum_{j=1}^M \beta_j \mathbf{a}_j$ with $\boldsymbol{\beta} \in \Delta^{M-1}$. For every integer $k \geq 1$ there exist indices $j_1, \dots, j_k$, not necessarily distinct, such that*

$$\Big\| \frac{1}{k} \sum_{i=1}^{k} \mathbf{a}_{j_i} - \mathbf{p}^* \Big\|^2 \;\leq\; \frac{D^2}{2k}. \tag{16}$$

Proof. Let $\mathbf{X}, \mathbf{X}'$ be independent with $\Pr(\mathbf{X} = \mathbf{a}_j) = \Pr(\mathbf{X}' = \mathbf{a}_j) = \beta_j$, so that $\mathbb{E}\mathbf{X} = \mathbf{p}^*$. Expanding $\|\mathbf{X} - \mathbf{X}'\|^2 = \|(\mathbf{X} - \mathbf{p}^*) - (\mathbf{X}' - \mathbf{p}^*)\|^2$ and using independence and the zero mean of the two terms gives $\mathbb{E}\|\mathbf{X} - \mathbf{X}'\|^2 = 2\,\mathbb{E}\|\mathbf{X} - \mathbf{p}^*\|^2$, and since $\|\mathbf{X} - \mathbf{X}'\| \leq D$ almost surely, $\mathbb{E}\|\mathbf{X} - \mathbf{p}^*\|^2 \leq D^2/2$. For i.i.d. copies $\mathbf{X}_1, \dots, \mathbf{X}_k$ of $\mathbf{X}$, the centred summands $\mathbf{X}_i - \mathbf{p}^*$ are independent with mean zero, so the cross terms vanish and

$$\mathbb{E}\Big\| \frac{1}{k} \sum_{i=1}^{k} \mathbf{X}_i - \mathbf{p}^* \Big\|^2 = \frac{1}{k}\, \mathbb{E}\|\mathbf{X} - \mathbf{p}^*\|^2 \leq \frac{D^2}{2k}. \tag{17}$$

A random variable cannot exceed its mean on every outcome, so some realization $(j_1, \dots, j_k)$ attains at most this expectation. □

F.2 The $1/\sqrt{k}$ Rate Is Tight

The approximation step of Theorem 1 cannot be improved beyond a factor of $\sqrt{2}$: there are anchor configurations and targets for which every k-sparse composition is $\Omega(D_{\mathcal{A}}/\sqrt{k})$ away from the target.

Proposition 1 (Tightness). *Let $M \leq d_\ell$ and $D > 0$. There exist anchors $\mathbf{a}_1, \dots, \mathbf{a}_M \in \mathbb{R}^{d_\ell}$ with diameter $D_{\mathcal{A}} = D$ and a target $\mathbf{p}^* \in \mathrm{conv}(\mathbf{A})$ such that, for every $k < M$ and every $\boldsymbol{\alpha} \in \Delta^{M-1}$ with $\|\boldsymbol{\alpha}\|_0 \leq k$,*

$$\|\boldsymbol{\alpha}^\top \mathbf{A} - \mathbf{p}^*\| \;\geq\; D_{\mathcal{A}} \sqrt{\frac{1}{2}\Big(\frac{1}{k} - \frac{1}{M}\Big)} \;\geq\; \frac{D_{\mathcal{A}}}{2\sqrt{k}} \quad \textit{whenever } M \geq 2k. \tag{18}$$

Proof. Let $\mathbf{a}_j = (D/\sqrt{2})\,\mathbf{u}_j$ for orthonormal $\mathbf{u}_1, \dots, \mathbf{u}_M$, so that $\|\mathbf{a}_i - \mathbf{a}_j\| = D$ for all $i \neq j$, and let $\mathbf{p}^* = \frac{1}{M}\sum_j \mathbf{a}_j$ be the centroid. By orthonormality, $\|\boldsymbol{\alpha}^\top \mathbf{A} - \mathbf{p}^*\|^2 = \frac{D^2}{2}\sum_j (\alpha_j - \frac{1}{M})^2$. Fix a support T of size $s \leq k$. The entries outside T contribute $(M - s)/M^2$, and, subject to $\sum_{j \in T} \alpha_j = 1$, the entries inside T contribute at least $s(\frac{1}{s} - \frac{1}{M})^2$, attained at $\alpha_j = 1/s$. The sum of both is $\frac{1}{s} - \frac{1}{M} \geq \frac{1}{k} - \frac{1}{M}$, which proves the first inequality; for $M \geq 2k$, $\frac{1}{k} - \frac{1}{M} \geq \frac{1}{2k}$ gives the second. □

Comparing Eq. 18 with the upper bound $D_{\mathcal{A}}/\sqrt{2k}$ of Lemma 1, the two differ by a factor of at most $\sqrt{2}$. The same holds for the fidelity bound itself: if F is a linear isometry (so that $L_{\mathcal{S}} = 1$ on the hull

of the anchors of Proposition 1), then $\|F(\mathbf{p}) - F(\mathbf{p}^*)\| = \|\mathbf{p} - \mathbf{p}^*\| \geq D_{\mathcal{A}}/(2\sqrt{k})$ for every k-sparse composition $\mathbf{p}$. No bound that holds for all anchor configurations and all $L_{\mathcal{S}}$-Lipschitz maps can therefore improve on the $O(1/\sqrt{k})$ dependence of Theorem 1 by more than a constant factor.

F.3 On-Manifold versus Global Lipschitz Constants

The constant $L_{\mathcal{S}}$ always exists. Holding the textual segments of the input fixed, the map F of Section 5 is a composition of linear maps, rotary position embeddings, RMS normalisation with a positive stabiliser, SiLU activations, and softmax attention, all of which are continuously differentiable. The anchor hull $\mathcal{S}$ is compact and convex, being the convex hull of finitely many anchors. By the mean-value inequality along the segment $[\mathbf{x}, \mathbf{y}] \subseteq \mathcal{S}$,

$$\|F(\mathbf{x}) - F(\mathbf{y})\| \leq L_{\mathcal{S}} \|\mathbf{x} - \mathbf{y}\| \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathcal{S}, \qquad L_{\mathcal{S}} = \max_{\mathbf{z} \in \mathcal{S}} \|J_F(\mathbf{z})\|_{\mathrm{op}} < \infty, \tag{19}$$

where J_F is the Jacobian of F. The assumption of Theorem 1 is thus always satisfied; its content lies in the *size* of $L_{\mathcal{S}}$, which is governed by the LLM's Jacobian on a region spanned by embedded schema text, inputs of the kind pre-training controls.

Remark 1. *For a prompt that may leave $\mathcal{S}$, the relevant constant is $L_{\mathrm{glob}} = \sup_{\mathbf{z} \in \mathcal{B}} \|J_F(\mathbf{z})\|_{\mathrm{op}} \geq L_{\mathcal{S}}$, where $\mathcal{B} \supseteq \mathcal{S}$ is a bounded convex region containing every prompt the projection can produce. Two effects can make L_{glob} much larger than $L_{\mathcal{S}}$. First, worst-case bounds compound across layers: if each of the n_{L} blocks is σ-Lipschitz on the relevant region, only $L_{\mathrm{glob}} \leq \sigma^{n_{\mathrm{L}}}$ can be guaranteed, which is exponential in depth. Second, dot-product self-attention is not globally Lipschitz at all (Kim et al., 2021): its local constant grows with the norm of the input, and a free projection is not constrained in norm, so the region $\mathcal{B}$ it can reach, and with it L_{glob}, is not controlled by pre-training. On $\mathcal{S}$, in contrast, pre-training drives the network to map nearby embedded text to nearby representations at every layer. A free projection that places a prompt $\tilde{\mathbf{p}}$ outside $\mathcal{S}$ therefore cannot invoke $L_{\mathcal{S}}$; the only available control is*

$$\|F(\tilde{\mathbf{p}}) - F(\mathbf{p}^*)\| \leq L_{\mathrm{glob}} \|\tilde{\mathbf{p}} - \mathbf{p}^*\|, \tag{20}$$

which may be vacuous even when $\|\tilde{\mathbf{p}} - \mathbf{p}^*\|$ is moderate. This gap is the formal counterpart of the semantic blindness described in Section 1: nothing in pre-training constrains the LLM's behaviour on inputs outside its training distribution.

F.4 The Semantic Manifold Is Not the Token-Embedding Hull

The main text speaks of the LLM's *semantic manifold*: the structured region of $\mathbb{R}^{d_\ell}$ occupied by the pre-trained token embeddings and by the embeddings of natural-language content built from them, on which pre-training shapes how the model reads its input. This section fixes the formal status of that notion: what it refers to, which of our claims rest on it, and why we do not formalise it as the convex hull of the token embeddings.

Terminology. *Manifold* is used in the informal sense standard in representation learning, as in the manifold hypothesis, not in the differential-geometric sense. Empirically, the token and hidden representations of pre-trained language models concentrate in a narrow, anisotropic region of the ambient space (Ethayarajh, 2019; Mu & Viswanath, 2018; Gao et al., 2019); we cite this phenomenon as motivation that LLM inputs are far from generic points of $\mathbb{R}^{d_\ell}$, and no formal result in this paper depends on its exact geometry. Every formal statement is made on the anchor hull $\mathcal{S} = \mathrm{conv}(\{\mathbf{a}_j\}_{j=1}^{M})$, a concrete compact polytope whose vertices are embedded schema descriptions, that is, points on the manifold in the only sense our arguments use: vectors built exclusively from the embeddings of real text, on which pre-training fixes the model's behaviour.

Why not the convex hull of the vocabulary. A tempting formalisation of the manifold would be $\mathcal{H} = \mathrm{conv}(\{\mathbf{e}_i\}_{i=1}^{|\mathcal{V}_{\mathrm{tok}}|})$, the convex hull of the token-embedding table. We avoid it, for a reason worth recording: $\mathcal{H}$ is far too large for membership in it to carry information.

Proposition 2 (The token-embedding hull is full-dimensional)**.** *Suppose some $d_\ell + 1$ of the token embeddings $\{\mathbf{e}_i\}_{i=1}^{|\mathcal{V}_{\mathrm{tok}}|}$ are affinely independent. Then $\mathcal{H}$ has nonempty interior, and*

the embedding centroid $\bar{\mathbf{e}} = |\mathcal{V}_{\text{tok}}|^{-1} \sum_i \mathbf{e}_i$ *is an interior point: there exists* $r > 0$ *such that* $\{\mathbf{x} : \|\mathbf{x} - \bar{\mathbf{e}}\| \le r\} \subseteq \mathcal{H}$.

Proof. Let T index $d_\ell + 1$ affinely independent embeddings, let $\Delta_T = \text{conv}(\{\mathbf{e}_i\}_{i \in T})$ be their simplex, and let $\mathbf{c} = (d_\ell + 1)^{-1} \sum_{i \in T} \mathbf{e}_i$ be its barycentre. A non-degenerate simplex has nonempty interior with its barycentre in the interior, so there is $\rho > 0$ with $B(\mathbf{c}, \rho) \subseteq \Delta_T$. Write $N_{\text{tok}} = |\mathcal{V}_{\text{tok}}|$ and $m = (d_\ell + 1)/N_{\text{tok}}$, and split the centroid as $\bar{\mathbf{e}} = m\,\mathbf{c} + (1 - m)\,\mathbf{c}'$ with $\mathbf{c}' = (N_{\text{tok}} - d_\ell - 1)^{-1} \sum_{i \notin T} \mathbf{e}_i \in \mathcal{H}$. For any $\mathbf{v}$ with $\|\mathbf{v}\| \le m\rho$,

$$\bar{\mathbf{e}} + \mathbf{v} = m\Big(\mathbf{c} + \frac{\mathbf{v}}{m}\Big) + (1 - m)\,\mathbf{c}', \tag{21}$$

and $\mathbf{c} + \mathbf{v}/m \in \Delta_T \subseteq \mathcal{H}$, so $\bar{\mathbf{e}} + \mathbf{v} \in \mathcal{H}$ by convexity. Taking $r = m\rho$ completes the proof. □

For the backbone used in this paper, $|\mathcal{V}_{\text{tok}}| = 128{,}256$ embeddings span $d_\ell = 2{,}048$ dimensions, so the hypothesis of Proposition 2 fails only if the entire vocabulary lies in a proper affine subspace, which a single rank computation on the centred embedding table refutes. $\mathcal{H}$ therefore contains a full-dimensional ball around the embedding centroid, and a freely projected prompt of moderate norm can perfectly well lie *inside* it.

Consequences for how our claims are to be read. *(i) No claim of off-hull placement.* We do not claim that the prompts of a trained free projection lie outside $\mathcal{H}$, or at any particular distance from it. For a given prompt $\tilde{\mathbf{p}}$, the distance $\text{dist}(\tilde{\mathbf{p}}, \mathcal{H})$ is computable by a quadratic program, but we deliberately base no argument on its value, because neither outcome would be informative: a prompt outside $\mathcal{H}$ has no expression over tokens, and a prompt inside $\mathcal{H}$ still has none that is distinguished or available. By Carathéodory's theorem, an interior point of $\mathcal{H}$ can be written as a convex combination of at most $d_\ell + 1$ token embeddings, but such a representation is far from unique, is supported on arbitrary vocabulary items rather than schema concepts, and is neither computed by the projection nor seen by the model. Membership in $\mathcal{H}$ therefore certifies nothing about meaning, which is precisely why we do not adopt $\mathcal{H}$ as the semantic manifold. Nor is there any tension with the statement of Section 1 that freely projected prompts land off the manifold: that statement concerns the narrow region of embedded text, which a vector optimised with no reference to it has no reason to meet, not the full-dimensional polytope $\mathcal{H}$; Figure 1 depicts the former, not $\mathcal{H}$.

(ii) What semantic blindness asserts. The deficit named in Section 1 is, operationally, the absence of a certificate: a free projection returns a vector that is the embedding of no word and arrives with no expression in terms of tokens, concepts, or schema elements, so nothing ties it to the meanings by which the LLM's pre-trained knowledge is indexed. RelAnchor's guarantee is correspondingly stronger than any membership statement: every prompt token comes with an explicit convex expression $\boldsymbol{\alpha}_u$ over M named schema anchors, gated with a known coefficient towards the graph vector, and it is this expression, not a position in space, that the theory exploits, in the form of the constant $L_{\mathcal{S}}$ certified on the fixed compact anchor hull (Remark 1).

(iii) The empirical test is interventional. Because membership in $\mathcal{H}$ carries no information, the informative empirical test of semantic blindness is not a distance measurement to $\mathcal{H}$ but the rotated-anchor control of Section 6.3 and Appendix F.8: it preserves the dictionary's entire intrinsic geometry and the achievable assignments (Proposition 6) and severs exactly one thing, the anchors' identity as embedded schema text. The 5.75-point drop in mean AUROC it causes (Table 15) is direct, causal evidence for the grounding mechanism, of a kind that no measurement of $\text{dist}(\cdot, \mathcal{H})$ could supply.

F.5 Soft Assignments Are Effectively Sparse

Theorem 1 is stated for k-sparse assignments, whereas RelAnchor learns a dense softmax assignment (Eq. 4). The following two results show that, at the small temperatures we use, a softmax assignment is uniformly close to a k-sparse one, so the theorem describes the prompts the model actually produces.

Lemma 2 (Truncation). *Let $\boldsymbol{\alpha} \in \Delta^{M-1}$, let $K \subseteq [M]$ carry mass $1-\varepsilon = \sum_{j\in K} \alpha_j > 0$, and let $\boldsymbol{\alpha}^{(K)}$ be the renormalised restriction, $\alpha_j^{(K)} = \alpha_j/(1-\varepsilon)$ for $j \in K$ and 0 otherwise. Then*

$$\left\|\boldsymbol{\alpha}^\top \mathbf{A} - \boldsymbol{\alpha}^{(K)\top} \mathbf{A}\right\| \le \varepsilon\, D_{\mathcal{A}}. \tag{22}$$

Proof. If $\varepsilon = 0$ the two vectors coincide. Otherwise let $\boldsymbol{\mu}_{\text{in}} = \boldsymbol{\alpha}^{(K)\top}\mathbf{A}$ and $\boldsymbol{\mu}_{\text{out}} = \varepsilon^{-1}\sum_{j\notin K}\alpha_j \mathbf{a}_j$, both points of $\text{conv}(\mathbf{A})$. Then $\boldsymbol{\alpha}^\top\mathbf{A} = (1-\varepsilon)\boldsymbol{\mu}_{\text{in}} + \varepsilon\boldsymbol{\mu}_{\text{out}}$, hence $\boldsymbol{\alpha}^\top\mathbf{A} - \boldsymbol{\mu}_{\text{in}} = \varepsilon(\boldsymbol{\mu}_{\text{out}} - \boldsymbol{\mu}_{\text{in}})$, and $\|\boldsymbol{\mu}_{\text{out}} - \boldsymbol{\mu}_{\text{in}}\| \le D_{\mathcal{A}}$ because $\text{conv}(\mathbf{A})$ has diameter $D_{\mathcal{A}}$. □

Proposition 3 (Effective sparsity of soft assignment). *Let $\mathbf{z} \in \mathbb{R}^M$ with sorted entries $z_{(1)} \ge \cdots \ge z_{(M)}$, let K_k index the k largest entries, let $\Delta_k = z_{(k)} - z_{(k+1)} \ge 0$ be the k-th logit gap, and let $\boldsymbol{\alpha} = \text{softmax}(\mathbf{z}/\tau)$. Then the tail mass satisfies $\varepsilon_k = \sum_{j\notin K_k}\alpha_j \le \frac{M-k}{k}\, e^{-\Delta_k/\tau}$, and the k-sparse assignment $\boldsymbol{\alpha}^{(k)} = \boldsymbol{\alpha}^{(K_k)}$ satisfies*

$$\left\|F(\boldsymbol{\alpha}^\top\mathbf{A}) - F(\boldsymbol{\alpha}^{(k)\top}\mathbf{A})\right\| \;\le\; L_{\mathcal{S}}\, D_{\mathcal{A}}\, \frac{M-k}{k}\, e^{-\Delta_k/\tau}. \tag{23}$$

Under the eligibility mask of Eq. 12, the same holds with M replaced by $|\mathcal{E}(\phi(u))|$.

Proof. Write $Z_{\text{in}} = \sum_{j\in K_k} e^{z_j/\tau} \ge k\, e^{z_{(k)}/\tau}$ and $Z_{\text{out}} = \sum_{j\notin K_k} e^{z_j/\tau} \le (M-k)\, e^{z_{(k+1)}/\tau}$. Then $\varepsilon_k = Z_{\text{out}}/(Z_{\text{in}} + Z_{\text{out}}) \le Z_{\text{out}}/Z_{\text{in}} \le \frac{M-k}{k} e^{-\Delta_k/\tau}$. Lemma 2 bounds the distance of the two compositions by $\varepsilon_k D_{\mathcal{A}}$; both compositions lie in $\mathcal{S} = \text{conv}(\mathbf{A})$, so the Lipschitz property of F on $\mathcal{S}$ gives Eq. 23. Masking sets the ineligible logits to $-\infty$, which removes them from both sums. □

The bound decays exponentially in Δ_k/τ. For example, with the default $\tau = 0.07$, a `drivers` row of F1 ($|\mathcal{E}| = 37$), $k = 4$, and a logit gap of $\Delta_4 = 0.5$, the soft composition lies within $\frac{33}{4} e^{-0.5/0.07} D_{\mathcal{A}} < 0.007\, D_{\mathcal{A}}$ of a 4-sparse composition. Small temperatures thus make the learned assignment behave like the sparse assignments of Theorem 1 while keeping it differentiable, which is precisely the trade-off the hard-assignment control of the ablation study gives up.

F.6 The Gate Contracts Prompts towards the Manifold

The node tokens of RelAnchor are gated mixtures $\mathbf{p}_u = (1-\alpha)\widetilde{\mathbf{g}}_u + \alpha \mathbf{s}_u$ rather than pure compositions. The next result shows how the gate interpolates between a free and an on-manifold prompt.

Proposition 4 (Gate contraction). *Let $\text{dist}(\mathbf{x}, \mathcal{S}) = \min_{\mathbf{y}\in\mathcal{S}} \|\mathbf{x} - \mathbf{y}\|$ and let $\Pi_{\mathcal{S}}$ denote the Euclidean projection onto $\mathcal{S}$. For every $\alpha \in [0,1]$, every $\widetilde{\mathbf{g}}_u \in \mathbb{R}^{d_\ell}$, and every $\mathbf{s}_u \in \mathcal{S}$,*

$$\text{dist}(\mathbf{p}_u, \mathcal{S}) \le (1-\alpha)\,\text{dist}(\widetilde{\mathbf{g}}_u, \mathcal{S}). \tag{24}$$

Consequently, if F is $L_{\mathcal{B}}$-Lipschitz on a convex region $\mathcal{B}$ containing $\mathcal{S}$ and $\mathbf{p}_u$, then $\|F(\mathbf{p}_u) - F(\Pi_{\mathcal{S}}(\mathbf{p}_u))\| \le (1-\alpha)\, L_{\mathcal{B}}\, \text{dist}(\widetilde{\mathbf{g}}_u, \mathcal{S})$.

Proof. The point $\mathbf{y} = (1-\alpha)\Pi_{\mathcal{S}}(\widetilde{\mathbf{g}}_u) + \alpha\mathbf{s}_u$ lies in $\mathcal{S}$ by convexity, and $\|\mathbf{p}_u - \mathbf{y}\| = (1-\alpha)\|\widetilde{\mathbf{g}}_u - \Pi_{\mathcal{S}}(\widetilde{\mathbf{g}}_u)\| = (1-\alpha)\,\text{dist}(\widetilde{\mathbf{g}}_u, \mathcal{S})$, which bounds $\text{dist}(\mathbf{p}_u, \mathcal{S})$. The second claim follows from $\|\mathbf{p}_u - \Pi_{\mathcal{S}}(\mathbf{p}_u)\| = \text{dist}(\mathbf{p}_u, \mathcal{S})$ and the Lipschitz property on $\mathcal{B}$. □

The gate therefore trades off two quantities: at $\alpha = 1$ every node token is an exact composition and Theorem 1 applies directly, while at $\alpha = 0$ the prompt is as unconstrained as a free projection. The learned gate outperforms every fixed value, including both extremes (Table 19), which is the empirical face of the Composition Bottleneck of Section 1: node tokens benefit from staying anchored close to, but not exactly on, the semantic manifold, while the neighbourhood token $\mathbf{s}_{\text{G}}$ remains exactly on it.

Is the contraction vacuous for a small gate? A natural concern is that Proposition 4 only bites when α is appreciable: were training to leave the shared gate near its initialisation $\sigma(a) = 0.1$ (Appendix B.4), each node token would remain dominated by its freely projected component and the contraction factor $1 - \alpha$ would stay close to one. Three observations rule out this reading of RelAnchor. First, retraining with the gate frozen at the initialisation value—the $\alpha = 0.10$ column of Table 19—is exactly the model this concern describes, and it trails the learned gate on all ten tasks, by 6.96 AUROC points in the classification mean and 11.6% in the regression mean; a gate that effectively stayed at 0.1 could not separate from this control by margins of that size. Second, anchoring is not carried by the node gate alone: the neighbourhood token $\mathbf{s}_{\mathrm{G}}$ lies exactly in $\mathcal{S}$ at every α, and the composition $\mathbf{s}_u$ enters every node token, so replacing the dictionary while keeping the gate and all other components—the generic-vocabulary and rotated-anchor controls of Section 6.3—still costs 4.50 and 5.75 mean AUROC points. The LLM's reading of the prompt thus depends on the anchors' content and on-manifold orientation at whatever injection strength the model selects. Third, a strong contraction is not the design goal: $\alpha = 1$, where every node token is an exact composition, is the worst fixed setting in mean MAE, and Section 4.3 claims interpolation, not replacement. Proposition 4 should accordingly be read as stated: it guarantees that the distance of each node token to the anchor hull is controlled by a single learned scalar—shrinking wherever the task profits from more grounding, as the fixed-α sweep shows it often does—rather than promising, or needing, a fixed large α.

F.7 Row Distinctiveness under Sparse Composition

Theorem 1 guarantees that faithful sparse assignments exist. The next result shows that the composition map does not conflate them.

Proposition 5 (Row Distinctiveness)**.** *Suppose any* $\min(2k, M)$ *of the anchors* $\mathbf{a}_1, \ldots, \mathbf{a}_M$ *are affinely independent, a general-position condition that can only hold when* $\min(2k, M) \le d_\ell + 1$*. Then the map* $\boldsymbol{\alpha} \mapsto \boldsymbol{\alpha}^\top \mathbf{A}$ *is injective on* k*-sparse probability vectors: for* $\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_v \in \Delta^{M-1}$ *with* $\|\boldsymbol{\alpha}_u\|_0, \|\boldsymbol{\alpha}_v\|_0 \le k$*,*

$$\boldsymbol{\alpha}_u \neq \boldsymbol{\alpha}_v \quad \Longrightarrow \quad \boldsymbol{\alpha}_u^\top \mathbf{A} \neq \boldsymbol{\alpha}_v^\top \mathbf{A}. \tag{25}$$

Proof. Suppose $\boldsymbol{\alpha}_u^\top \mathbf{A} = \boldsymbol{\alpha}_v^\top \mathbf{A}$ with $\boldsymbol{\alpha}_u \neq \boldsymbol{\alpha}_v$, and let $T = \operatorname{supp}(\boldsymbol{\alpha}_u) \cup \operatorname{supp}(\boldsymbol{\alpha}_v)$, so $|T| \le \min(2k, M)$. The vector $\boldsymbol{\gamma} = (\boldsymbol{\alpha}_u - \boldsymbol{\alpha}_v)|_T$ is nonzero, satisfies $\sum_{j \in T} \gamma_j = 0$ because both assignments lie on the simplex, and satisfies $\sum_{j \in T} \gamma_j \, \mathbf{a}_j = \mathbf{0}$ by hypothesis. This is a nontrivial affine dependence among $\{\mathbf{a}_j\}_{j \in T}$, contradicting the assumed affine independence. □

Remark 2. *The hypothesis of Proposition 5 is a general-position condition. Any* $2k$ *points of* $\mathbb{R}^{d_\ell}$ *drawn from a continuous distribution are affinely independent with probability one as long as* $2k \le d_\ell + 1$*, and* $d_\ell = 2{,}048$ *is far larger than the sparsity levels of interest. Schema anchors are not random points, however: because their descriptions are generated from shared templates, a few exact affine relations can occur, for instance among the four descriptions obtained by crossing two table names with two column names in the same template when the names tokenise to the same number of tokens. The proposition then applies to every pair of assignments whose joint support avoids such a configuration, and the condition can be checked directly for any given support by a rank computation on at most* $2k$ *centred anchors. Read together, Theorem 1 and Proposition 5 address the two challenges of Section 1: the fidelity bound addresses Semantic Dilution by showing that restricting prompts to a compact, database-specific anchor dictionary costs only* $O(1/\sqrt{k})$ *in fidelity, and injectivity, together with the gate contraction of Proposition 4, addresses the Composition Bottleneck by ensuring that the composition neither collapses distinct rows onto the same prompt, the failure mode of a naive discrete assignment, nor confines the prompt exactly to the anchor hull.*

F.8 What the Rotated-Anchor Control Isolates

The rotated-anchor control of Section 6.3 replaces $\mathbf{A}$ by $\mathbf{A}_{\mathrm{rot}} = \mathbf{A}\mathbf{Q}$ for an orthogonal $\mathbf{Q}$. The next result makes precise that this control leaves everything except the LLM's reading of the prompt unchanged, so the performance gap it produces measures on-manifold alignment and nothing else.

Proposition 6 (Rotation invariance of the assignment). *Let $\mathbf{Q} \in \mathbb{R}^{d_\ell \times d_\ell}$ be orthogonal and suppose the last layer of the query networks q_ω and q_ω^{G} is affine. For every parameter setting of RelAnchor with anchors $\mathbf{A}$ there is a parameter setting of the rotated control, and vice versa, such that for every node u:* (i) *the scores, and hence the assignments $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_{\mathrm{G}}$, coincide;* (ii) *the compositions satisfy $\mathbf{s}_u^{\mathrm{rot}} = \mathbf{s}_u\mathbf{Q}$ and $\mathbf{s}_{\mathrm{G}}^{\mathrm{rot}} = \mathbf{s}_{\mathrm{G}}\mathbf{Q}$ (as row vectors); and* (iii) *the anchor Gram matrix and diameter are unchanged, $\mathbf{A}_{\mathrm{rot}}\mathbf{A}_{\mathrm{rot}}^\top = \mathbf{A}\mathbf{A}^\top$ and $D_{\mathcal{A}_{\mathrm{rot}}} = D_{\mathcal{A}}$.*

Proof. If the last layer of q_ω is $\mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, replace it by $\mathbf{x} \mapsto \mathbf{Q}^\top(\mathbf{W}\mathbf{x} + \mathbf{b})$, so that the rotated query is $\mathbf{Q}^\top q_\omega(\mathbf{h}_u)$. Then $\mathbf{A}_{\mathrm{rot}}\mathbf{Q}^\top q_\omega(\mathbf{h}_u) = \mathbf{A}\mathbf{Q}\mathbf{Q}^\top q_\omega(\mathbf{h}_u) = \mathbf{A}q_\omega(\mathbf{h}_u)$, so the scores coincide, and the eligibility masks are unchanged; the same argument applies to q_ω^{G}. This gives (i), and (ii) follows from $\boldsymbol{\alpha}_u^\top\mathbf{A}\mathbf{Q} = \mathbf{s}_u\mathbf{Q}$. The converse uses $\mathbf{Q}$ in place of $\mathbf{Q}^\top$. Claim (iii) is $\mathbf{Q}\mathbf{Q}^\top = \mathbf{I}$ and the invariance of distances under $\mathbf{Q}$. □

Every quantity that enters the Euclidean part of Theorem 1 ($D_{\mathcal{A}}$, the achievable assignments, and the geometry of the hull) is therefore identical under rotation. What changes is only whether the compositions lie on the LLM's semantic manifold: compositions over $\mathbf{A}$ lie in the anchor hull $\mathcal{S}$, whose vertices are embedded schema text, whereas $\mathrm{conv}(\mathbf{A}\mathbf{Q})$ is in general not contained in $\mathcal{S}$ and its vertices are the embeddings of no text, so the rotated prompts can only be controlled with L_{glob} (Remark 1). Up to optimisation effects, the 5.75-point loss in mean AUROC under rotation (Table 15) is thus attributable to the LLM's reading of off-manifold prompts, not to a less expressive dictionary or assignment.

F.9 Extension to Multi-Token Prompts

Theorem 1 concerns a single prompt position, whereas the prefix in Eq. 5 consists of N composed tokens $\mathbf{P} = [\mathbf{p}_{u_1}; \ldots; \mathbf{p}_{u_N}] \in \mathbb{R}^{N \times d_\ell}$.

Corollary 1 (Multi-token fidelity). *Let $\bar{F} : \mathbb{R}^{N \times d_\ell} \to \mathbb{R}^{d_o}$ be the full prefix-to-output map and suppose it is $\bar{L}_{\mathcal{S}}$-Lipschitz on $\mathcal{S}^N$ with respect to the Frobenius norm. For any target prefix $\mathbf{P}^*$ whose rows lie in $\mathrm{conv}(\mathbf{A})$ and any $k \geq 1$, there is a prefix $\mathbf{P}$ whose rows are k-sparse compositions over the anchors such that*

$$\left\|\bar{F}(\mathbf{P}) - \bar{F}(\mathbf{P}^*)\right\| \leq \bar{L}_{\mathcal{S}}\, D_{\mathcal{A}} \sqrt{\frac{N}{2k}}. \tag{26}$$

Proof. Applying Lemma 1 to each row separately gives k-sparse compositions $\mathbf{p}_{u_i}$ with $\|\mathbf{p}_{u_i} - \mathbf{p}^*_{u_i}\| \leq D_{\mathcal{A}}/\sqrt{2k}$, hence

$$\|\mathbf{P} - \mathbf{P}^*\|_F = \Big(\sum_{i=1}^{N} \|\mathbf{p}_{u_i} - \mathbf{p}^*_{u_i}\|^2\Big)^{1/2} \leq D_{\mathcal{A}} \sqrt{\frac{N}{2k}}. \tag{27}$$

All rows of $\mathbf{P}$ and $\mathbf{P}^*$ lie in $\mathcal{S}$, and the Lipschitz property of $\bar{F}$ on $\mathcal{S}^N$ concludes the proof. □

The factor $\sqrt{N}$ reflects sequence length and leaves the $1/\sqrt{k}$ dependence on sparsity unchanged; N is bounded by the fixed neighbour-sampling budget. The neighbourhood token $\mathbf{s}_{\mathrm{G}}$ is itself a composition over the anchors and can be included as an $(N{+}1)$-th row without changing the argument.

G Broader Impact

RelAnchor targets predictive tasks on relational databases, such as churn, demand, and outcome prediction, which support decisions in commerce, healthcare, and science. Such predictions can affect individuals, and the usual safeguards against biased or opaque automated decisions apply. A favourable property of our design is that the LLM never reads raw cell values: anchors are built from schema names and coarse statistics only, and entity information reaches the LLM exclusively through learned graph representations, which reduces the exposure of individual records compared with text serialization of database rows. Anchor assignments are also inspectable, since every weight refers to a named schema element, which can aid auditing of what a prediction is conditioned on.