RT-J: Large-Scale Pretraining of Relational Transformers for Context-Efficient Predictions

Rishabh Ranjan[⋈] **Vignesh Kothapalli**[⋈] **Harshvardhan Agarwal**[⋈]
Charilaos Kanatsoulis[⋈] **Roshan Upendra**[π] **Tom Palczewski**[π]
Carlos Guestrin[⋈] **Jure Leskovec**[⋈]

[⋈]Stanford University [π]SAP

https://star-project.stanford.edu/rt-j

Abstract

Foundation models for tables and relational databases are increasingly used for predictive tasks across various scientific and industrial domains. However, adapting existing models to unseen datasets requires tens of thousands of examples. Here we introduce **RT-J**, an 86M-parameter relational foundation model (RFM) that makes accurate predictions even with hundreds of in-context examples or fewer. To achieve this, we first assemble **THE JOIN**, a large collection of 639 web-crawled databases with 6,023 LLM-generated forecasting tasks. We pretrain with long contexts of up to 8,192 database cells, using in-context examples from a novel lightweight random-walk retriever. Finally, we leverage context tuning and ensembling to scale up test-time compute. On in-context regression tasks from RelBench, **RT-J** requires $10\times$ fewer examples on average to match strong feature engineering pipelines with LLM-agent or RDBLearn featurizers and TabICLv2 or LightGBM predictors. Overall, our work pushes the frontier for context-efficient predictions on relational data and highlights the importance of real-world semantics, context engineering and large-scale pretraining for RFMs.

1 Introduction

Foundation models have transformed how machine learning systems are adapted to new tasks [1, 2]. In language and vision, pretrained models can often solve new tasks using only a small number of examples in context [3, 4]. Yet this capability remains largely out of reach for relational data, even though relational databases and data warehouses store much of the world's structured enterprise, scientific, and operational data [5, 6]. Today, strong relational predictors typically require task-specific training [7–9], or substantial labeled data [10–16]. This limits predictive modeling precisely in the settings where it is most valuable, including cases where labeled examples are scarce, and users need rapid ad hoc predictions for decision-making.

Tabular foundation models (TFMs) like TabPFN [17, 18] and TabICL [19, 20] provide in-context learning for single-table prediction, and systems such as RDBLearn [21] extend them to relational databases by aggregating relational neighborhoods into engineered features. Such an approach can be effective with many labeled examples, but it flattens the database and removes the explicit relational structure needed to effectively reason over entities, relationships, and cross-table dependencies. Natively relational architectures like RelGT [8] and RelGNN [7] operate directly on multi-table databases by representing rows as nodes and primary–foreign-key links as edges, but require training for each schema and task. More recent schema-agnostic relational foundation models (RFMs), such

Correspondence to: <ranjanr,vigneshk,guestrin,jure>@cs.stanford.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

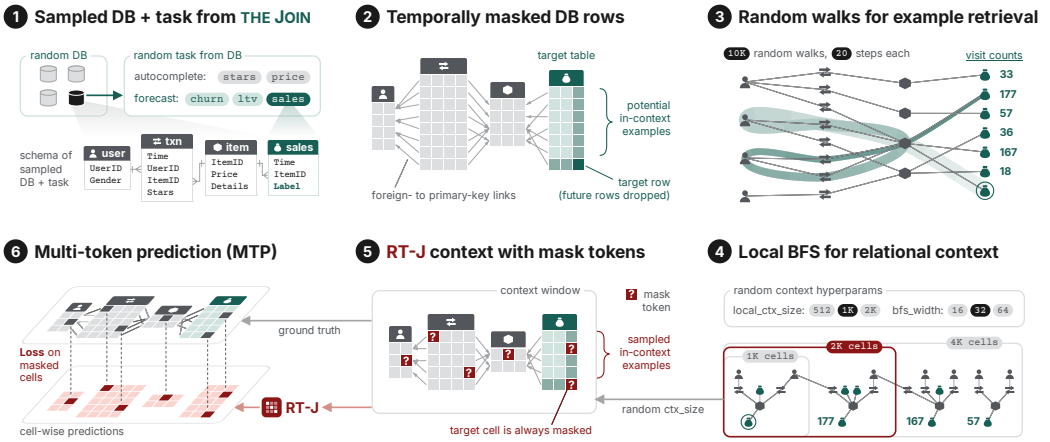


Figure 1: **RT-J** pretraining recipe. **(1)** We sample a database and prediction task from THE JOIN. The task specifies the *target table* and the *target column*. **(2)** We sample a *target row* from the target table. Rows after the target row's timestamp are dropped; kept target-table rows form the candidate *in-context examples*. **(3)** From the target row, many parallel random walks rank candidate examples by visit count, on the fly. **(4)** We process candidate examples in descending visit-count order, using each as the root of a local bounded-width breadth-first-search (BFS) over primary–foreign key links; the context size and BFS hyperparameters are sampled randomly. **(5)** The collected rows form the *context window* for the relational transformer (RT), which treats cells as *tokens*. The target table rows in the context enable *in-context learning*. The *target cell* is always masked; non-target cells may be masked optionally. **(6)** **RT-J** predicts all masked cells and is trained against their ground-truth values.

as Griffin [22], KumoRFM [10, 11], and relational transformers (RTs) [23], pretrain across many databases and adapt to unseen ones at inference time. However, these models have not yet been tailored for strong prediction with only a few labeled examples. See Appendix C for related work.

This inability is not primarily an architectural limitation, but a mismatch between existing RFM training recipes and inference with limited in-context examples. RT [23] provides a natural starting point because it operates directly over database cells, schema metadata, and relational links. However, the original RT recipe was not designed to turn the architecture into a strong in-context learner. There are three main gaps. First, publicly available real world relational pretraining data remains scarce [24, 25]. Second, existing pretraining recipes target zero-shot or fine-tuning regimes, which precludes in-context learning. Third, standard context construction either ignores relational structure or expands local neighborhoods naively, without preferentially retrieving rows relevant to the target.

In this paper, we show that a pretrained RT can become a strong context-efficient predictor when the recipe for training, data, context retrieval and test-time compute is designed for this regime. Our goal is not to introduce a new backbone architecture, but to identify the ingredients that turn an RT model into an effective in-context learner for relational databases. First, we develop a pretraining recipe (Figure 1) built on stochastic context sizes and hyperparameters, which exposes the model to long and diverse contexts. Second, to train with high-quality in-context examples, we propose a lightweight random-walk retriever (Figure 1) that fills the context window with rows selected for both relational proximity to the target and likelihood of carrying useful predictive signal. Third, we leverage context tuning and context ensembling to scale test-time compute. With this recipe, we first obtain RT-PluRel, an 86M-parameter RT model pretrained on synthetic PluRel data [26]. With task-specific fine-tuning,

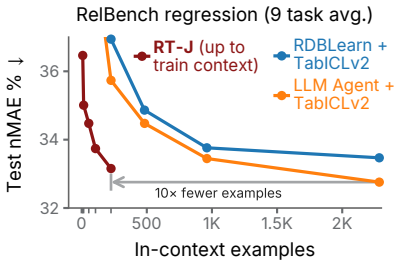


Figure 2: In-context regression. Examples are matched. **RT-J** at train context size of 8192 cells fits around 225 examples on average. At all example counts, **RT-J** is > 2 nMAE points better than the best baseline; at all performance levels, **RT-J** needs $\gtrsim 10\times$ fewer examples.

RT-PluRel ranked first among all system and model submissions on the RelArena-α [27] leaderboard (Appendix A). Finally, to improve generalization to real data, we assemble **THE JOIN**, the largest open pretraining corpus of real-world relational data to date, comprising 6K LLM-generated forecasting tasks and 11K autocomplete tasks across 639 diverse web-crawled databases. We refer to the model obtained by continued pretraining (CPT) of RT-PluRel on **THE JOIN** with our recipe as **RT-J**. Compared to the original RT pretraining [23], our pretraining used $11\times$ more total tokens with up to $8\times$ larger contexts and a $4\times$ larger model, amounting to roughly $70\times$ more pretraining FLOPs.

Together, these ingredients make **RT-J** far more context-efficient than strong feature-engineering pipelines (Figure 2). We compare **RT-J** against pipelines that compose an LLM agent or RDBLearn featurizer with a TabICLv2 or LightGBM predictor, each given the same examples as in **RT-J**'s context. Other RFMs such as KumoRFM [10, 11], Griffin [22] and RDB-PFN [28] do not support this regime, hence could not be compared against (Appendix B). On RelBench regression, **RT-J** fits 225 examples on average at its max training context size of 8192 database cells, and achieves a normalized mean absolute error (nMAE) which the strongest baseline matches only with roughly $10\times$ as many examples. Moreover, with only 15 examples, **RT-J** already beats the best error any pipeline reaches with 225 examples. At matched example counts, **RT-J** leads at every context size by 2.3–5.0 nMAE points. On classification, **RT-J** leads by 3.1 AUROC points when examples are scarcest, and the strongest pipeline catches up only at the largest context size. We attribute this efficiency to **RT-J** processing the raw relational context natively, which feature-engineering pipelines flatten into a feature table via aggregation techniques.

Our results demonstrate that context-efficient relational prediction can be attained with a pretraining recipe matched to this regime and scaling up along the axes of data, context and test-time compute. Overall, this is a step toward RFMs that work for databases the way LLMs work for text: pretrained once, reused across domains, and adapted to new tasks with only a few examples.

2 Background: Relational Transformers for Predictive Tasks on Databases

Relational databases and relational structure. Let $\mathcal{D}$ denote a relational database with schema graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$, where each node $T \in \mathcal{T}$ is a table and each directed edge $(\widetilde{T}, T) \in \mathcal{E}$ indicates that table T contains a foreign key column referencing the rows of the parent table $\widetilde{T}$. Each table T contains rows indexed by primary key(s) and columns consisting of numerical, categorical, textual, or temporal values, and optional foreign keys. Thus, the relational structure of $\mathcal{D}$ is characterized by the schema graph along with row-level bipartite connectivity via primary–foreign key links [6, 26, 29].

Prediction tasks on relational data. A prediction task specifies a target table $T^\star$, a target column $c^\star$, and a set of prediction rows $\mathcal{I}$ in $T^\star$. For each row $r \in \mathcal{I}$, the target value is $y_r = x_{r,c^\star}$. The goal of a foundation model f_θ is to predict y_r from the observed relational context around row r. For boolean or categorical targets, f_θ outputs a class distribution. For numerical targets, it outputs a real-valued prediction. This formulation covers both (1) *autocompletion* and (2) *forecasting* tasks. In an autocompletion task, $T^\star$ is an existing database table and $c^\star$ is one of its columns (such as imputing the price of a transaction). In a forecasting task, $T^\star$ may be a curated prediction table whose rows define future prediction queries (such as forecasting revenue for an entity in the next quarter). When either task is temporal, the context is restricted to rows available before the prediction time.

In-context learning for prediction tasks. In a classical in-context learning approach, f_θ predicts the target column $c^\star$ for a query row r by conditioning on a support set $\mathcal{S}_r$ of labeled rows from the target table $T^\star$. Each support row $i \in \mathcal{S}_r$ has an observed target value $y_i = x_{i,c^\star}$. A local relational context $\mathcal{C}_i$ is then constructed around each support row and a context $\mathcal{C}_r$ around the query row r (with its target value masked). The model f_θ predicts y_r using the support and query context as:

$$\hat{y}_r = f_\theta(\{(\mathcal{C}_i, y_i) : i \in \mathcal{S}_r\}, \mathcal{C}_r)\,. \tag{1}$$

Here, in-context examples are sampled explicitly as labeled rows, and each example is represented by its own relational context. However, support rows often retrieve overlapping cells through primary–foreign key links. For example, labeled rows for people who purchased the same shoe after seeing the same advertisement may include the same cells from the shoe table in their contexts. This duplication consumes the context budget and limits the number of examples in context.

Context window of relational transformers. RT [23] instantiates the context preparation idea differently. Rather than separately sampling support rows and constructing an independent context around each one, RT samples a single query-centered context $\mathcal{C}_r$ around the query row r (see stage (4) in Figure 1). This context is obtained by expanding from r along primary–foreign key links while keeping y_r masked. During this expansion, the context may naturally include other rows from the target table whose target values are observed. Thus, the support examples are not sampled separately. They are induced implicitly by the relational context around the query row. Therefore, the prediction of an RT model is given by $\hat{y}_r = f_\theta(\mathcal{C}_r)$. The benefit is that overlapping relational neighborhoods are represented only once inside a single query-centered context, rather than being duplicated across independently sampled support examples.

Masked token prediction with relational transformers. RT treats each cell in $\mathcal{C}_r$ as a token and predicts the masked target cell with a masked token prediction objective. Thus, heterogeneous prediction tasks are reduced to a common 'cell prediction' problem over any database in a schema-agnostic manner. RT computes the input token embedding by combining the cell value, datatype-specific information, and frozen text embeddings of the corresponding table and column names. During the forward pass, each transformer block of RT applies three attention layers [26] in sequence: (1) *column attention*, where tokens attend within the same column, (2) *feature attention*, where tokens attend within the same row, and (3) *neighbor attention*, where tokens attend across rows connected by primary–foreign key links. This attention structure enables information sharing across rows, columns, and tables while preserving the relational structure of the database. See Appendix F for details.

3 THE JOIN: A Large-Scale Pretraining Corpus of Relational Data

Existing collections of relational datasets, such as RelBench [24, 29], CTU [25], BIRD [30], and Spider [31, 32] comprise diverse databases but fail to capture the structural and distributional variety of predictive tasks needed for training RFMs. To address this issue, we introduce **THE JOIN**, a corpus comprising 639 relational databases across diverse domains (e-commerce, sports, media, finance, healthcare, etc.) and 6K forecasting tasks spanning various levels of difficulty, including rare events and cold-start entities. We follow a four-stage approach to curate the corpus: (1) collection of raw databases from the web, (2) standardization of data types in tables, (3) task generation using LLMs, and (4) task filtering using an XGBoost model. See Appendix D for details.

3.1 Collection

We collected the raw data from five disjoint source classes: (1) curated relational repositories (CTU, OGB, OpenAlex), (2) public APIs (SEC EDGAR, CFPB, World Bank, EIA), (3) domain specific portals (Lahman, ATP, StatsBomb, Cricsheet, OpenFlights), (4) community dumps (Stack Exchange), and (5) benchmarks (Kaggle, Spider/BIRD). Every contributing source is either a publicly downloadable archive or a public API. Licenses vary by source, from public domain and permissive licenses to research-only, non-commercial and provider-specific terms, and these per-source restrictions apply to the released data (Appendix D.6). The classes carry qualitatively different relational structures: CTU schemas are small, clean, and follow third-normal-form, whereas Stack Exchange dumps are wide and denormalized. As a preliminary quality filter, we only collect databases that have at least two tables joined by a foreign key, and at least one column that can be parsed as a timestamp. Table 4 lists the domains included in the corpus. To avoid contamination, we exclude databases that overlap with ones in RelBench [29], RelBench v2 [24] and 4DBInfer [33] using source identity rather than schema similarity.

3.2 Standardization

The databases collected in their raw form are usually messy, with inconsistencies across column data types, timestamps, and primary-foreign key links being the major issues. For example: natively boolean columns are represented as strings `"true"/"false"`, dates are represented as ISO strings, and foreign keys exist only as a column-naming convention with no explicit declaration. To address such issues, we apply five idempotent detector-and-transformer fixes that recover the canonical relational structure: (1) *boolean coercion*, rewriting string and `uint8` columns whose values fall within a canonical truth-set as booleans, (2) *numeric coercion*, casting string-encoded numerics to

integers or floats with an identifier blacklist guarding zip codes and ISBNs, (3) *foreign-key inference* by name match plus an $\geq 80\%$ value-containment test against parent primary keys, (4) *temporal normalization*, electing a single time column per table from regex, name-hint, and integer-year candidates; and (5) *entity-axis subsampling* of oversized databases, retaining all activity rows for sampled entities to preserve their full activity histories.

3.3 Task generation

While autocomplete tasks are naturally defined by simply masking a cell/column in an existing database, in forecasting tasks the target cell/column (e.g., revenue next month) does not appear in the database and has to be explicitly generated. To obtain forecasting tasks we proceed as follows.

Audit and eligibility. Each database is first parsed into a typed schema graph with two named roles: an *entity table* (whose primary key is referenced by a foreign key (FK) from another table, identifying a long-lived object such as a customer, player, or hospital) and an *activity table* (with both a timestamp column and an FK to an entity table, so each row is a timestamped event attributable to an entity). A database is marked as eligible for task generation if it has at least one entity table, at least one activity table, ≥ 5 unique timestamps in that activity table, and ≥ 100 rows. The first two conditions ensure the schema can express a per-entity forecasting task. The timestamp and row count conditions allow us to perform XGBoost-based filtering of the generated tasks for quality control.

Propose and generate. Proposing forecasting tasks for a database requires subject-matter expertise and a scalable way to verify correctness. We use an LLM (Claude Opus 4.6) for the former and a lightweight deterministic checker for the latter. For every eligible database, the LLM is given the schema description in text and is asked to write a set of task proposals in JSON. Each proposal selects one of 25 registered template families, grouped into eight reasoning regimes. These include *aggregation*, *threshold/existence*, *trend/change*, *behavior/churn*, *ranking*, *adoption/novelty*, *streak*, and *tail-event* tasks. The checker then iterates through each proposal and drops it if: (1) a referenced column does not exist, (2) the entity column is not a foreign key, or (3) the requested pattern is not one of the registered templates. The LLM proposer is run multiple times per database, where each pass sees the tasks already accepted and is asked specifically for templates with low coverage.

3.4 Task filtering

Finally, we score each task using an XGBoost model over window-aggregation features. We retain tasks that are either *learnable*, where XGBoost beats the lightweight historical baselines, or where the signal is too weak for these baselines but which still act as harder learning objectives for stronger models. Overall, this process gives us $6{,}023$ forecasting tasks, where each task is defined by its description and a table with timestamped labels (e.g., future revenue for each quarter)[1]. See Appendix D for case studies and a manual audit of the tasks for their practicality and realism.

4 Pretraining Recipe

Here we describe our key improvements to the pretraining recipe of the original RT paper [23], which enable large-scale pretraining and context-efficient predictions (Figure 1).

4.1 Long-context training

The context size L is the number of cells presented to RT and is a key scaling axis to improve prediction quality. However, training at large L is computationally expensive. Further, the same pretrained model should be able to handle both small and large contexts at test-time, so that inference compute can be traded off against quality. Hence, we train at a mix of context sizes. For each training batch, we sample $L \sim \mathrm{Unif}(\mathcal{L})$ from a fixed set $\mathcal{L}$ of context sizes. We keep batch size B constant irrespective of L for stable optimization. As GPU memory scales as $L \times B$, we use gradient accumulation adaptively to prevent out-of-memory (OOM) errors at large L.

[1] Our code is publicly available at: `https://github.com/stanford-star/agentic-task-gen`

4.2 Retrieval

The original RT recipe [23] samples the relational context $\mathcal{C}_r$ with a bounded-width breadth-first search (BFS) over primary and foreign key links starting from the target row r. As the context length L grows, BFS can exhaust the local neighborhood, retrieve less relevant rows, and miss useful support rows from the target table. Retrieval is especially important for relational data because contexts are heterogeneous. For example, when predicting whether a user will click on a shoe ad, plain BFS may spend the context budget on generic user history or similar product cells. We use a two-stage sampling approach based on random walks to first retrieve similar rows and then expand the local neighborhood of each such row to prepare the context. Our approach can hence prioritize nearby target rows repeatedly reached through shared users, shoes, and ads tables for high quality contexts.

Stage 1: Random-walks for global retrieval. The first stage uses random walks inspired by Pixie [34] and PinSAGE [35]. We run W random walks of length K from the target row r and record visit counts for rows in the target table $T^\star$. We rank these rows by visit count, optionally using timestamps when the recency bias parameter ρ is set to 1. The walk parameters W and K approximate the random-walk ordering efficiently and are tuned for latency. Unlike vector-similarity retrieval, this sampler requires no pre-computation and can run on the fly across large databases in a highly parallelizable manner. For example, we use $W = 10{,}000$ random walks, each of $K = 20$ steps, with zero data loading overhead as the context sampler running on the CPU is ready with the next batch before the forward and backward passes have completed on the GPU. While random walks add a small latency overhead at test-time, our sensitivity analysis in Appendix I shows that lower values of W and K reduce this overhead significantly while largely retaining accuracy.

Stage 2: BFS for local retrieval. For the second stage, we use the ranked rows as seeds for local BFS with context size ℓ and width bound b until the total context size reaches L. The full procedure is given in Algorithm 1 (Appendix I). Along with L, the retrieval parameters (ℓ, b, ρ) can be tuned at test time. To allow for this, we need to expose the model to diverse contexts during training. Hence, we randomly sample $(\ell, b, \rho) \sim \mathrm{Unif}(\mathcal{L}_\ell \times \mathcal{B} \times \mathcal{R})$ using the values listed in Section 5.

5 Pretraining Results

Pretraining details. We use an 86M-parameter RT model with 12 layers, 512 hidden size and 8 attention heads, incorporating the improvements of [26] (such as QK-Norm) over the original architecture of [23]. During pretraining, we sample context size L from $\mathcal{L} = \{512, \ldots, 8192\}$, the local context size ℓ from $\mathcal{L}_\ell = \{128, \ldots, 8192\}$, the BFS width-bound b from $\mathcal{B} = \{8, \ldots, 256\}$ (all powers of two) and the recency parameter ρ from $\mathcal{R} = \{0, 1\}$, and mask only the target cell (see Appendix H for experiments with multi-cell masking).

We pretrain in two phases, first on the synthetic PluRel corpus [26] (86,211 tasks over 1,900 databases)[2] and then on 13,243 tasks from a 523-database subset of **THE JOIN** (excluding the largest DBs due to memory bottlenecks). We optimize with Muon [36] at a constant learning rate after warmup and select the best checkpoint after stochastic weight averaging (SWA) [37] as recommended by Ranjan et al. [38]. The same checkpoint is used for all classification and regression tasks. Each phase takes about a day on a single node with 8×A100 GPUs. See Appendix G for details.

Evaluation. We evaluate on RelBench [29], which comprises 21 forecasting tasks across 7 databases (9 regression and 12 binary classification tasks). These tasks span rare-event, cold-start, heavy-tailed and long-horizon regimes (see Appendix E for task descriptions). All databases used at evaluation are held out from pretraining. Following the official RelBench leaderboard, we use Z-score normalized mean absolute error (nMAE, lower is better) for regression and AUROC (higher is better) for classification tasks and report the average metrics.

5.1 Baseline comparisons

Baselines. We compare **RT-J** against the strongest publicly available in-context learning (ICL) pipelines for relational prediction. A pipeline consists of a featurizer followed by a predictor. We

[2]The dataset is publicly available at: `https://huggingface.co/datasets/stanford-star/plurel`

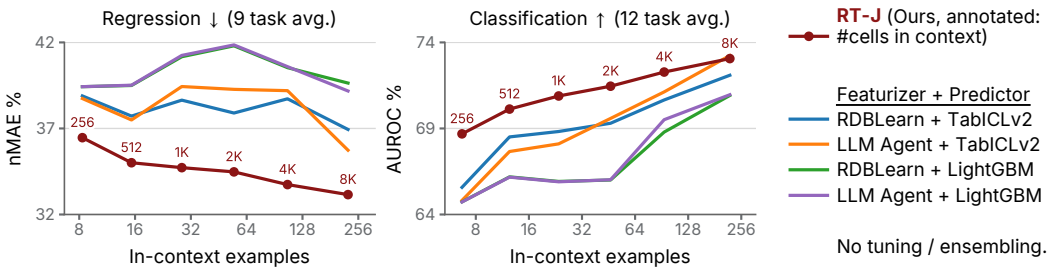


Figure 3: **RT-J** vs baselines. On regression (left), **RT-J** dominates baselines at every example count, by up to 5 nMAE points over the strongest one. On classification (right), **RT-J** leads by up to 3 AUROC points, especially when examples are scarce. Per-task plots are in Appendix J.

consider two featurizers: (1) **RDBLearn** [21], which aggregates relational neighborhoods into a flat feature table and (2) an **LLM Agent** (Claude Code with Opus 4.6), which explores the database and writes SQL queries to perform feature engineering before applying a tabular predictor. We use two predictors: (1) **TabICLv2**, an in-context tabular foundation model [19, 20], and (2) **LightGBM** [39], gradient-boosted decision trees at the library's default hyperparameters. Other RFMs such as KumoRFM [10, 11], Griffin [22], RDB-PFN [28] and TabPFN-Rel [15, 16] could not be used as baselines for reasons detailed in Appendix B. We compare against RT-PluRel in Section 5.2.

Setup. For every task, we sweep the test-time context size $L \in \{256, 512, 1024, 2048, 4096, 8192\}$ cells for **RT-J**, holding all other context hyperparameters fixed, ensuring that the 256-cell context is a subset of the 512-cell context and so on. For a fair label-matched evaluation of the baselines, we extract the task rows from the context prepared for **RT-J**, but do not further constrain the feature extraction or processing from the source database. Any checkpoint selection or hyperparameter tuning is done on the validation split, and all numbers are reported on the full test splits.

On regression, **RT-J** beats every baseline at every context size, by 2.3–5.0 nMAE points (see Figure 3). It is also far more example-efficient: with 15 in-context examples it already beats the best error any baseline reaches with 225, and the strongest pipeline needs around 2300 examples, 10× more, to surpass it. Since all methods see the same examples, we hypothesize that the gap comes from **RT-J** better exploiting the relational context around them, which baselines flatten into a set of features. On classification, **RT-J** leads by 3.1 AUROC points when examples are scarcest, but the baselines improve steeply with more examples and the strongest catches up at the rightmost end. Among the baselines, the predictor matters more than the featurizer: TabICLv2 beats LightGBM at every context size on both metrics, while LLM Agent and RDBLearn features perform similarly.

5.2 Impact of synthetic and real pretraining phases

Setup. We isolate the contribution of each pretraining phase by comparing three RT models with the same architecture: randomly initialized RT, RT-PluRel (PluRel phase only) and **RT-J** (PluRel followed by **THE JOIN**), on both fine-tuning and in-context learning regimes. These pipelines include test-time scaling (Section 6) and reflect our submissions to the official RelBench leaderboard. Appendix A details both pipelines and reports per-task results.

Observations. Table 1 shows the results. Synthetic pretraining on PluRel improves fine-tuning over a randomly initialized RT by 1.7 nMAE and 0.8 AUROC points, and continued pretraining on **THE JOIN** adds 0.5 nMAE and 0.5 AUROC points on top. The real-data phase matters most in-context, where **RT-J** improves over RT-PluRel by 0.6 nMAE and 1.5 AUROC points.

6 Test-Time Compute Scaling

Considering the fixed context window of **RT-J**, choosing informative input cells for a predictive task becomes crucial. As the optimal context hyperparameters can be task dependent, and sampling cells with a particular configuration is inherently random, we employ (1) context tuning and (2) context ensembling as two complementary ways to improve predictions of **RT-J**. The compute, time and

Table 1: Regression nMAE and classification AUROC averaged over all RelBench tasks for RT models with no pretraining (RT), synthetic-only pretraining (RT-PluRel) and synthetic + real pretraining (**RT-J**), for both fine-tuning and in-context learning settings. Per-task results are in Appendix A.

		Fine-tuned		In-context	
Model	Pretraining data	nMAE % ↓	AUROC % ↑	nMAE % ↓	AUROC % ↑
RT	none (random init)	29.26	77.76	–	–
RT-PluRel	PluRel	27.57	78.53	32.95	72.98
RT-J	PluRel → THE JOIN	**27.11**	**79.02**	**32.35**	**74.45**

label overheads of tuning and ensembling approaches are compared in Appendix J.3. Here we use a subsampled test set for computational feasibility.

6.1 Context hyperparameter tuning

Intuitively, during inference with **RT-J**, different tasks might prioritize width vs. depth of BFS, local vs. global neighborhoods, and recent vs. long-term patterns differently. We therefore tune (L, ℓ, b, ρ) separately for each task on its validation split and apply the best configuration on the test split. The full protocol and the selected per-task configurations are given in Table 13 (Appendix J).

Observations. From Figure 4, we notice that context tuning is primarily useful for regression. With a single context seed, it reduces mean nMAE from 33.4 to 32.5, and improves eight of the nine regression tasks. The largest gains occur on `rel-trial/site-success` ($91.0 \rightarrow 88.6$) and on `rel-hm/item-sales` ($14.0 \rightarrow 11.8$), the latter by selecting a smaller 512-cell context. For classification, tuning *hurts* a single seed on average ($72.9 \rightarrow 72.4$ AUROC), as the validation choice transfers to test on only five of twelve tasks (Table 13 in Appendix J). Moreover, among the selected configurations, three of nine regression tasks (`rel-amazon/item-ltv`, `rel-amazon/user-ltv`, `rel-hm/item-sales`) and four of twelve classification tasks (`rel-avito/user-visits`, `rel-event/user-ignore`, `rel-event/user-repeat`, `rel-stack/user-engagement`) prefer a context with $L < 8192$ cells.

6.2 Context ensembling

Both the random-walk retrieval and the bounded-width local BFS stages of our context sampler are stochastic, with different random seeds yielding different yet equally viable contexts. Context ensembling simply averages the predictions made from n_{ens} independently sampled contexts. Ensembling reduces prediction noise due to retrieval by averaging over contexts and can be computed in parallel across n_{ens}. We sweep over $n_{\text{ens}} \in \{1, 2, 4, 8, 16\}$ using different sampler seeds with a default configuration of $L = 8192$, $\ell = 256$, $b = 32$, $\rho = 1$ and showcase the benefits.

Observations. Figure 4 shows the results. With the per-task-tuned configuration, increasing n_{ens} from 1 to 16 yields a $+1.3$ AUROC point aggregate gain for classification tasks, while at the default context it is slightly lower at $+1.1$ AUROC points. Regression tasks show smaller but consistent improvements (0.2 nMAE points on average at the default context). Based on the per-task curves in Appendix J, the largest default-context gains accrue to `rel-trial/study-outcome` with $+5.3$, `rel-event/user-repeat` with $+2.6$ and `rel-hm/user-churn` with an improvement of $+2.4$ AUROC points from n_{ens}=1 to 16, while `rel-avito/user-clicks` degrades by 3.0 AUROC points. On regression tasks the gains top out at 0.7 nMAE points on `rel-event/user-attendance`.

7 Ablation Studies

Here we isolate the impact of the key components of our recipe. The schema-semantics (Section 7.1) and context-retrieval (Section 7.2) ablations swap a single component of the full **RT-J** pipeline at inference time, evaluated on a fixed 8192-row subsampled test set per task. Table 2 summarizes the gain from various components. More pretraining ablations are in Appendix H.

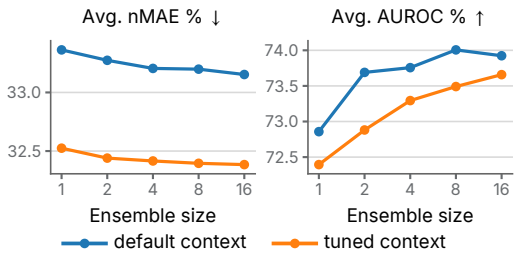


Figure 4: Context ensembling at test time, with default and per-task tuned context hyperparameters. Ensembling improves both metrics for both configurations. Per-task plots are in Appendix J.

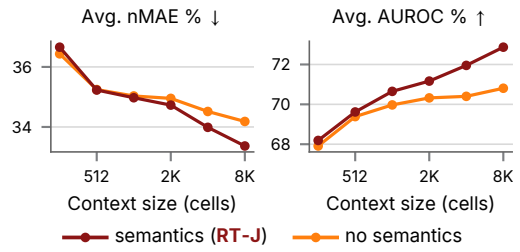


Figure 5: Schema-semantics ablation. Shuffling table and column names at evaluation degrades results increasingly with context. Per-task plots are in Appendix J.

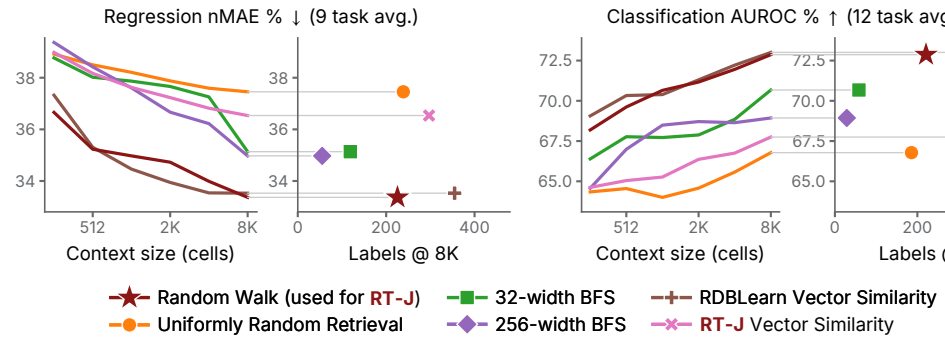


Figure 6: Context-retrieval ablation with **RT-J** at test-time. Left sub-panels plot the metric against context size L and right sub-panels plot it against the average labels delivered at $L = 8192$ (non-label cells provide relational context). Random-walk (ours) is competitive with the best, despite being cheaper and more scalable. **RT-J** generalizes to other retrievers too. Retriever quality is not strongly correlated with the label count; relational context is important too. Per-task plots are in Appendix J.

7.1 Schema semantics

RT-J consumes language model (LM) embeddings of table/column names in addition to cell values (see also the cell embedding design in Appendix F). Figure 5 shows a comparison with and without such semantics, where semantics are removed by shuffling table/column names at evaluation time.

Observations. Shuffling table/column names degrades classification at every point of the sweep, by up to 2.1 AUROC points, and regression by up to 0.8 nMAE points. Notably, the gap widens with more in-context examples, which suggests that schema text provides prior information that the examples do not replace. These observations highlight that **RT-J** has evidently absorbed world knowledge attached to real table and column names during pretraining on THE JOIN.

7.2 Alternatives to random-walk retrieval

We compare our random-walk retrieval approach (Section 4) against five alternatives at inference: uniformly random task-row selection, BFS of width $\{32, 256\}$ rooted at the target (the original scheme of [23]), and two vector-similarity retrievers that score task rows by RDBLearn features or **RT-J** embeddings. In particular, the **RT-J** embedding of a task row is the final hidden state at its masked target cell, computed over a 256-cell BFS context around the row. Both similarity retrievers rank labeled task rows dated before the target by cosine similarity and use them, in that order, as seeds for the same local BFS as random-walk retrieval.

Observations. Figure 6 shows that our random walk approach and similarity search over RDBLearn's precomputed features are well ahead of the other retrievers and effectively tie at $L = 8192$

Table 2: Summary of gains, in nMAE (regression) and AUROC (classification) points.

Comparison	Reg.	Clf.	Comparison	Reg.	Clf.
Pretraining data			*Context retrieval*		
RT-J vs. RT-PluRel	+0.60	+1.47	random walk vs. width-256 BFS	+1.60	+3.93
THE JOIN-only vs. PluRel-only	+1.79	+2.83	random walk vs. uniform random	+4.09	+6.09
Test-time compute					
tuned vs. default context	+0.84	−0.47	16 context ensemble vs. 1 context	+0.21	+1.06

(33.4 versus 33.5 nMAE and 72.9 versus 73.0 AUROC). Random walk leads regression at the two smallest contexts ($L = 256$, by 0.7 nMAE, and $L = 512$) and again at $L = 8192$, whereas the similarity retriever is ahead by 0.4–0.8 nMAE at $L = 1024$–4096 and by at most 0.9 AUROC.

The scatter panels in Figure 6 separate label count from label relevance. RDBLearn-feature similarity packs the most labels into the context (394) and reaches parity with random walk performance (that uses 223 labels at $L = 8192$). However, uniform random selection retrieves a similar count (208) and trails by 4.1 nMAE and 6.1 AUROC. **RT-J**-embedding similarity stays 2.3–3.2 nMAE and 3.6–5.4 AUROC behind random walk at every L, yet similarity over RDBLearn's engineered features keeps improving with L, highlighting the importance of the embedding quality for ranking relevant task rows. The two BFS variants are restricted to their fixed local neighborhoods and trail random walk by 1.8 and 1.6 nMAE with 2.7× and 5.6× fewer labels, respectively. Overall, our random walk retriever sits at the label-efficient end of this trade-off as it does not need per-task feature pipelines that RDBLearn requires, while providing higher-quality contexts for predictions.

8 Discussion and Conclusion

Our results show that **RT-J** can make accurate predictions on new datasets with as few as a hundred examples, in a single forward pass. This makes it practical to solve a wide range of predictive tasks without building dedicated feature engineering pipelines, or relying on extensive labeled data. Examples include flagging likely churn or fraud in a small-scale enterprise without a large customer base; predicting outcomes of novel clinical trials with small cohorts; and answering one-off predictive questions from an analyst. **RT-J** could be exposed directly in the database system, allowing users to request quick ad-hoc forecasts without an expensive labeling step. It could also be offered as a tool for LLM agents to call when a question requires predicting an unknown value, unlike transforming existing ones with SQL. **THE JOIN** is a first step toward capturing the immense diversity of real-world applications of relational foundation models (RFMs) like **RT-J**. Beyond synthetic pretraining corpora prevalent for structured data foundation models, **THE JOIN** offers hundreds of real databases across science, healthcare, finance, sports, etc., with semantically meaningful predictive tasks spanning rare-event, cold-start, heavy-tailed and long-horizon regimes.

Conclusion. We developed a pretraining recipe for relational transformers (RTs) [23] with long-context training, random-walk retrieval of high-quality in-context examples, and test-time compute scaling via context tuning and ensembling. With this recipe, we obtain **RT-J** by first pretraining on PluRel [26] synthetic data and then on **THE JOIN**, our large-scale real data corpus. **RT-J** matches strong feature engineering pipelines with as much as 10× fewer examples. Experiments highlight the role of real data pretraining, schema semantics and context engineering. Our work moves RFMs closer to practical systems that can solve novel predictive tasks over databases on the fly.

Limitations and future work. **RT-J** has been trained up to 8,192 context cells; future work can explore efficient longer context training, which would allow both for more in-context examples, and larger relational contexts for them. **RT-J** is unable to solve link prediction and recommendation tasks, sharing this limitation with RT [23]. Our evaluation is limited to RelBench, with only 21 tasks on 7 databases—inspired by **THE JOIN**, future work can expand the scope of real-world relational benchmarks. We release (1) the RT-PluRel and **RT-J** models, (2) **THE JOIN** data, (3) the LLM agent task curation pipeline, and (4) our pretraining, fine-tuning and inference code at `star-project.stanford.edu/rt-j`, hoping to accelerate future research and adoption of RFMs.

Acknowledgments and Disclosure of Funding

We thank Anvita Gupta, Joe Meyer, Johannes Hoffart, Jonas De Schouwer, Lata Nair, Mahmoud Mohammadi, Marcel Roed, Mark Znidar, Matthias Fey, Michael Bereket, Rok Sosic, Ryan Welch, Valter Hudovernik, Vijay Prakash Dwivedi, Viswa Ganapathy, Yanay Rosen, and Yangyi Shen for discussions, feedback and support. We thank the anonymous reviewers for helping improve the paper. We also gratefully acknowledge the support of NSF under Nos. CCF-1918940 (Expeditions), DMS-2327709 (IHBEM), IIS-2403318 (III); NIH under No. 1U24NS146314-01, Gates Foundation, Stanford Data Applications Initiative, Wu Tsai Neurosciences Institute, Stanford Institute for Human-Centered AI, Chan Zuckerberg Initiative, Amazon, Genentech, SAP, and SCBX. RR is supported by the Amazon Core AI Fellowship. The content is solely the responsibility of the authors and does not necessarily represent the views of the funding entities.

References

[1] Rishi Bommasani et al. On the opportunities and risks of foundation models, 2022. URL https://arxiv.org/abs/2108.07258.

[2] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. *International Journal of Machine Learning and Cybernetics*, pages 1–65, 2024.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[4] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736, 2022.

[5] Liane Vogel, Benjamin Hilprecht, and Carsten Binnig. Towards foundation models for relational databases [vision paper]. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

[6] 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 *Forty-first International Conference on Machine Learning*, 2024.

[7] Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. RelGNN: Composite message passing for relational deep learning. In *Forty-second International Conference on Machine Learning*, 2025.

[8] Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer, 2025. URL https://arxiv.org/abs/2505.10960.

[9] Jakub Peleška and Gustav Šír. Transformers meet relational databases, 2024. URL https://arxiv.org/abs/2412.05218.

[10] Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen, and Jure Leskovec. KumoRFM: A foundation model for in-context learning on relational data, 2025.

[11] Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. KumoRFM-2: Scaling foundation models for relational learning, 2026. URL https://arxiv.org/abs/2604.12596.

[12] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[13] Hoang Thanh Lam, Beat Buesser, Hong Min, Tran Ngoc Minh, Martin Wistuba, Udayan Khurana, Gregory Bramble, Theodoros Salonidis, Dakuo Wang, and Horst Samulowitz. Automated data science for relational data. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2689–2692. IEEE, 2021.

[14] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[15] Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report, 2026. URL https://arxiv.org/abs/2605.13986.

[16] Benjamin Jäger, Nick Erickson, Léo Grinsztajn, Felix Birkel, Klemens Flöge, Oscar Key, Kürşat Kaya, Jonas Kübler, Adèle Frankel, Tobias Schröder, Anurag Garg, Jan Hendrik Metzen, David Salinas, Simon Bing, Kristina Collins, Tuana Çelik, Vahid Balazadeh, Lydia Sidhoum, Tomás Pereda, Brendan Roof, Andrej Tschalzev, Siyuan Guo, Philipp Singer, Lennart Purucker, Jake Robertson, Marie Salmon, Philipp Jund, Jerry Chen, Diana Kriuchkova, Arthur Cahu, Eliott Kalfon, Adrian Hayler, Georg Grab, Vitor Monteiro, Lilly Wehrhahn, Dominik Safaric, Clara Cornu, Alan Arazi, Rylee Grace, Simone Alessi, Mihir Manium, Bernhard Schölkopf, Yann LeCun, Madelon Hulsebos, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3.5: Technical report, 2026. URL https://arxiv.org/abs/2609.17895.

[17] 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*, 2023.

[18] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[19] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025.

[20] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model, 2026.

[21] Yanlin Zhang, Linjie Xu, Quan Gan, David Wipf, and Minjie Wang. RDBLearn: Simple in-context prediction over relational databases, 2026. URL https://arxiv.org/abs/2602.18495.

[22] 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 *Forty-second International Conference on Machine Learning*, 2025.

[23] Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. In *The Fourteenth International Conference on Learning Representations*, 2026.

[24] Justin Gu, Rishabh Ranjan, Charilaos Kanatsoulis, Haiming Tang, Martin Jurkovic, Valter Hudovernik, Mark Znidar, Pranshu Chaturvedi, Parth Shroff, Fengyu Li, et al. RelBench v2: A large-scale benchmark and repository for relational data. *arXiv preprint arXiv:2602.12606*, 2026.

[25] Jan Motl and Oliver Schulte. The CTU Prague relational learning repository. *arXiv preprint arXiv:1511.03086*, 2015.

[26] Vignesh Kothapalli, Rishabh Ranjan, Valter Hudovernik, Vijay Prakash Dwivedi, Johannes Hoffart, Carlos Guestrin, and Jure Leskovec. PluRel: Synthetic data unlocks scaling laws for relational foundation models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=rM9r9qyRSA.

[27] Adrian Hayler, Klemens Flöge, Alan Arazi, Rishabh Ranjan, Jure Leskovec, Felix Birkel, Brendan Roof, Anurag Garg, Kristina Collins, Lydia Sidhoum, et al. Advancing open and reproducible relational learning: RelArena-α, TabPFN-Rel and RPI. *arXiv preprint arXiv:2608.16319*, 2026.

[28] Yanbo Wang, Jiaxuan You, Chuan Shi, and Muhan Zhang. Relational in-context learning via synthetic pre-training with structural prior. In *Forty-third International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.03805.

[29] Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, et al. RelBench: A benchmark for deep learning on relational databases. *Advances in Neural Information Processing Systems*, 37:21330–21341, 2024.

[30] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. *Advances in Neural Information Processing Systems*, 36:42330–42357, 2023.

[31] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3911–3921, 2018.

[32] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *The Thirteenth International Conference on Learning Representations*, 2025.

[33] Minjie Wang, Quan Gan, David Wipf, Zhenkun Cai, Ning Li, Jianheng Tang, Yanlin Zhang, Zizhao Zhang, Zunyao Mao, Yakun Song, et al. 4DBInfer: A 4D benchmarking toolbox for graph-centric predictive modeling on RDBs. *Advances in Neural Information Processing Systems*, 37:27236–27273, 2024.

[34] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 world wide web conference*, pages 1775–1784, 2018.

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

[36] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

[37] P Izmailov, AG Wilson, D Podoprikhin, D Vetrov, and T Garipov. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885, 2018.

[38] Rishabh Ranjan, Saurabh Garg, Mrigank Raman, Carlos Guestrin, and Zachary Lipton. Post-hoc reversal: Are we selecting models prematurely? *Advances in Neural Information Processing Systems*, 37:91460–91491, 2024.

[39] 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 *Advances in Neural Information Processing Systems*, 2017.

[40] Charilaos I Kanatsoulis, Evelyn Choi, Stephanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

[41] Vijay Prakash Dwivedi, Charilaos Kanatsoulis, Shenyang Huang, and Jure Leskovec. Relational deep learning: Challenges, foundations and next-generation architectures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5999–6009, 2025.

[42] Marco Spinaci, Marek Polewczyk, Johannes Hoffart, Markus C. Kohler, Sam Thelin, and Tassilo Klein. PORTAL: Scalable tabular foundation models via content-specific tokenization, 2024. URL https://arxiv.org/abs/2410.13516.

[43] Myung Jun Kim, Leo Grinsztajn, and Gael Varoquaux. CARTE: Pretraining and transfer for tabular learning. In *International Conference on Machine Learning*, pages 23843–23866. PMLR, 2024.

[44] Marek Wydmuch, Łukasz Borchmann, and Filip Graliński. Tackling prediction tasks in relational databases with LLMs, 2024. URL https://arxiv.org/abs/2411.11829.

[45] Fang Wu, Vijay Prakash Dwivedi, and Jure Leskovec. Large language models are good relational learners, 2025. URL https://arxiv.org/abs/2506.05725.

[46] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023.

[47] Dominique Beaini, Shenyang Huang, Joao Alex Cunha, Zhiyi Li, Gabriela Moisescu-Pareja, Oleksandr Dymov, Samuel Maddrell-Mander, Callum McLean, Frederik Wenkel, Luis Müller, et al. Towards foundational models for molecular learning on large-scale multi-task datasets. In *The Twelfth International Conference on Learning Representations*, 2024.

[48] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, and Jure Leskovec. Prodigy: Enabling in-context learning over graphs. *Advances in Neural Information Processing Systems*, 36:16302–16317, 2023.

[49] Jianan Zhao, Zhaocheng Zhu, Mikhail Galkin, Hesham Mostafa, Michael M Bronstein, and Jian Tang. Fully-inductive node classification on arbitrary graphs. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

[51] Qifang Zhao, Weidong Ren, Tianyu Li, Hong Liu, Xingsheng He, and Xiaoxiao Xu. GraphGPT: Generative pre-trained graph eulerian transformer. In *Forty-second International Conference on Machine Learning*, 2025.

[52] Maya Bechler-Speicher, Yoel Gottlieb, Andrey Isakov, David Abensur, Ami Tavory, Daniel Haimovich, Ido Guy, and Udi Weinsberg. Billion-scale graph foundation models, 2026.

[53] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training, 2025. URL https://arxiv.org/abs/2502.16982.

[54] Changxin Tian, Jiapeng Wang, Qian Zhao, Kunlong Chen, Jia Liu, Ziqi Liu, Jiaxin Mao, Wayne Xin Zhao, Zhiqiang Zhang, and Jun Zhou. WSM: Decay-free learning rate schedule via checkpoint merging for LLM pre-training, 2025. URL https://arxiv.org/abs/2507.17634.

[55] Craig Kapfer, Kurt Stine, Balasubramanian Narasimhan, Christopher Mentzel, and Emmanuel Candes. Marlowe: Stanford's GPU-based computational instrument, January 2025. URL https://doi.org/10.5281/zenodo.14751899.

[56] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[57] Mingze Dong, Leda Wang, and Yuval Kluger. Understanding and enhancing mask-based pretraining towards universal representations. In *Advances in Neural Information Processing Systems*, 2025.

[58] Vignesh Kothapalli, Hamed Firooz, and Maziar Sanjabi. CoT-ICL lab: A synthetic framework for studying chain-of-thought learning from in-context demonstrations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14620–14642, 2025.

[59] Vignesh Kothapalli, Ata Fatahibaarzi, Hamed Firooz, and Maziar Sanjabi. To think or not to think: The hidden cost of meta-training with excessive CoT examples. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15619–15644, 2026.

A Leaderboard Results

RelArena-α. RelArena-α [27] re-runs every submission on the 21 entity-level forecasting tasks of RelBench v1 under a single API, data state and evaluation protocol, and aggregates per-task results into Elo ratings. It separates *models*, tuned under a standardized, compute-matched tuning regime, from *systems*, which follow the same data states and evaluation but bring their own tuning protocol. Figure 7 reproduces the combined model + system leaderboard from Hayler et al. [27]. RT-PluRel, submitted as a system with its own sequential tuning regime, ranks first with an Elo of 1861, ahead of TabPFN-Rel (API) at 1826 and of all relational deep learning baselines (GraphSAGE, RelGT, RelGNN) and RDBLearn. **RT-J** was not part of this release.

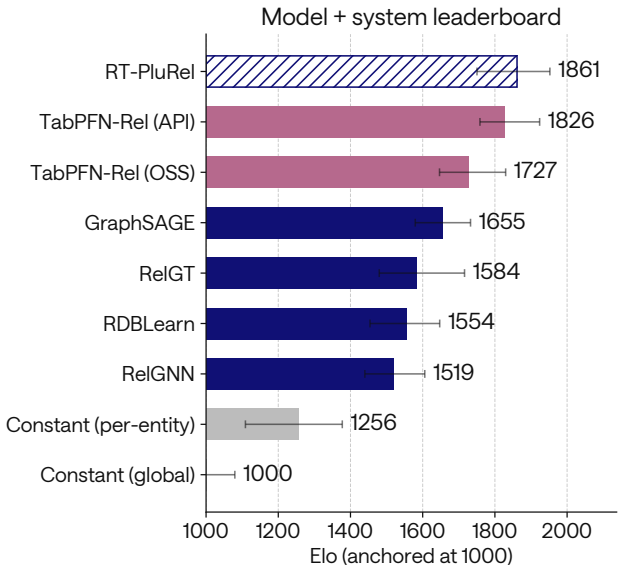


Figure 7: Combined model + system leaderboard of RelArena-α, reproduced from Hayler et al. [27]: Elo over the 21 RelBench v1 forecasting tasks (anchored at 1000 for the global constant predictor). RT-PluRel (hatched) is a system submission with its own tuning protocol; TabPFN-Rel variants in pink, other baselines in blue, constant predictors in grey.

RelBench leaderboard. Table 3 lists the per-task and mean test metrics of the RT family on the RelBench leaderboard, both fine-tuned on each task and in-context. In-context, **RT-J** improves over RT-PluRel on both regression (32.35 vs. 32.95 nMAE) and classification (74.45 vs. 72.98 AUROC). The gains from THE JOIN carry over to fine-tuning, where **RT-J** again has the best means of the three RT models (27.11 nMAE and 79.02 AUROC, against 27.57 and 78.53 for RT-PluRel and 29.26 and 77.76 for RT). We detail the pipelines used for our submissions below.

Fine-tuning pipeline. Each model is fine-tuned separately on each of the 21 tasks in four stages. (1) A selection run trains on the train split for up to 50K steps, evaluating every 100 steps on 1024 validation rows with a 4-seed context ensemble and stopping after 10K steps without improvement. (2) A context search scores 96 context configurations (L, ℓ, b, ρ) with $L \leq 1024$ on 4096 validation rows. (3) A refit trains on train and validation together for the selected number of steps, scaled by the ratio of train-plus-validation to train rows. (4) The SWA weights of the refit predict the full test split under the selected context hyperparameters with an 8-seed context ensemble. RT-PluRel and **RT-J** train an additive delta on their frozen pretrained weights, whereas RT trains all weights from random initialization. All runs use Muon with learning rate 5×10^{-4} and weight decay 0.1, batches of 256 contexts of up to 1024 cells, binary cross-entropy for classification and L1 loss for regression.

In-context pipeline. The pretrained checkpoint is frozen, and no RelBench database is seen in pretraining. Per task, we score 120 context configurations (L, ℓ, b, ρ) with $L \leq 8192$ on 4096 validation rows with a 4-seed context ensemble, then run each of the four best configurations with four context seeds on the full test split and average the 16 predictions per row.

Table 3: **RT models on the RelBench leaderboard.** Per-task test metrics of RT [23], RT-PluRel and **RT-J**, fine-tuned on each task or used in-context, as listed on the RelBench leaderboard. All metrics in percent. Best per row within each group in bold; lower is better for nMAE, higher is better for AUROC.

		Fine-tuned			In-context	
DB	Task	RT (rand init)	RT-PluRel	**RT-J**	RT-PluRel	**RT-J**
Regression (nMAE %, lower is better)						
rel-amazon	item-ltv	7.50	7.26	**7.19**	8.65	**7.81**
rel-amazon	user-ltv	**24.15**	24.58	24.29	29.60	**29.14**
rel-avito	ad-ctr	38.53	37.86	**36.03**	**41.71**	42.01
rel-event	user-attendance	32.52	31.64	**31.63**	43.78	**42.69**
rel-f1	driver-position	48.75	39.72	**37.67**	42.41	**38.75**
rel-hm	item-sales	7.75	**7.53**	7.57	11.44	**10.11**
rel-stack	post-votes	13.77	13.62	**12.79**	**14.70**	14.74
rel-trial	site-success	78.36	**75.17**	76.54	**87.65**	89.62
rel-trial	study-adverse	12.06	10.74	**10.28**	16.57	**16.33**
Mean		29.26	27.57	**27.11**	32.95	**32.35**
Classification (AUROC %, higher is better)						
rel-amazon	item-churn	82.96	**83.19**	83.16	74.40	**78.61**
rel-amazon	user-churn	71.30	71.34	**71.46**	66.24	**67.20**
rel-avito	user-clicks	**68.59**	65.89	66.81	46.20	**51.39**
rel-avito	user-visits	66.21	66.75	**66.89**	61.44	**64.09**
rel-event	user-ignore	81.41	80.85	**85.61**	76.78	**80.97**
rel-event	user-repeat	78.97	**80.29**	79.84	77.65	**79.12**
rel-f1	driver-dnf	79.93	82.09	**82.77**	81.89	**83.08**
rel-f1	driver-top3	83.84	90.01	**91.71**	89.69	**91.00**
rel-hm	user-churn	70.24	**70.61**	70.49	**65.77**	65.66
rel-stack	user-badge	**89.33**	89.25	89.21	**83.78**	81.11
rel-stack	user-engagement	90.74	**90.91**	90.65	**87.86**	86.76
rel-trial	study-outcome	69.55	**71.21**	69.57	64.08	**64.43**
Mean		77.76	78.53	**79.02**	72.98	**74.45**

B Other Relational Foundation Models

Beyond the pipelines of Section 5, we looked into comparing **RT-J** with recent relational foundation models (RFMs), but none could be evaluated under our protocol, in which every method sees the same in-context examples.

KumoRFM [10]: closed weights, and API access does not allow controlling the in-context examples, precluding a fair comparison.

KumoRFM-2 [11]: closed weights. API access was released only a few weeks before our submission deadline, and did not work during rebuttal due to the ongoing migration to NVIDIA.

Griffin [22]: the released checkpoints are incapable of in-context learning, and pretraining Griffin with our recipe is not feasible, as the recipe is engineered around transformer-based architectures, which Griffin is not.

RDB-PFN [28]: supports only classification tasks, whereas our evaluation covers both regression and classification.

TabPFN-Rel with TabPFN-3 and TabPFN-3.5 [15, 16]: released after our submission deadline. It is compared against RT-PluRel on RelArena-α in Hayler et al. [27] (Appendix A), where RT-PluRel ranks first (Elo 1861 vs. 1826 for TabPFN-Rel (API-only)); evaluating it under our matched-example protocol is left to future work. See the latest RelArena-α leaderboard, with **RT-J**, TabPFN-3.5 and other newer submissions, at `https://github.com/PriorLabs/relarena#-leaderboards`.

C Related Work

Relational deep learning and benchmarks. Relational deep learning (RDL) is a machine learning paradigm that operates directly on relational databases, avoiding the need to manually flatten a database into a single feature table [6]. A common formulation represents a database as a heterogeneous relational entity graph: rows are nodes, tables define node types, and primary–foreign-key links define typed edges [29]. Graph neural networks can then propagate information across related rows and tables, enabling end-to-end prediction over multi-table data [7, 29]. Recent work has improved the efficiency and expressiveness of relational message passing [7], and transformer-based relational architectures have also been proposed [8, 9]. Kanatsoulis et al. [40] introduce an efficient positional encoding framework based on lightweight message-passing operations, providing structural features that have been shown to improve relational deep learning models. Broader surveys and position papers frame these models as part of a larger agenda for relational deep learning [41].

Benchmarking has been central to this progress. RelBench [29] introduced a standardized suite of relational prediction tasks, while 4DBInfer [33] and RelBench v2 [24] expand the scale and diversity of relational benchmarks, including broader task families and database domains. These benchmarks make it possible to compare relational models under realistic multi-table and temporal constraints.

Relational foundation models. Recent work has begun to develop foundation models specifically for relational data. These models seek to transfer across schemas and tasks rather than training a separate predictor for every database. KumoRFM and KumoRFM-2 study in-context learning for relational data at scale [10, 11]. Griffin develops a graph-centric relational foundation model for transfer across relational databases [22]. The Relational Transformer (RT) takes a complementary cell-level approach: database cells are treated as tokens, schema semantics are included in the input, and relational attention masks encode primary–foreign-key structure [23]. PluRel [26] enables RT pretraining with synthetic data. It synthesizes multi-table databases from scratch, including schemas, primary–foreign-key links, and feature distributions, enabling controlled studies of synthetic data scaling for relational foundation models.

Our work builds directly on RT because its cell-level formulation is well-suited to context-efficient relational prediction. Labeled task rows, unlabeled rows, schema metadata, and linked evidence can all appear in the same context window, making masked cell prediction a natural interface for both pretraining and inference. However, the original RT work primarily establishes a schema-agnostic architecture; it does not develop a scalable pretraining and inference recipe. We therefore use RT as the backbone and study how to make it effective when only a small number of in-context examples are available.

Tabular foundation models and relational tabularization. Tabular foundation models aim to bring pretraining and in-context learning to single-table prediction. TabPFN and its successors show that transformers can perform strong in-context prediction when labeled rows from a tabular task are provided in the context [17, 18]. More recent models, including TabICLv2 and other large-scale tabular foundation models, extend this paradigm to larger datasets, richer feature spaces, and broader tabular distributions [19, 20, 42, 43]. These methods provide strong evidence that structured prediction can benefit from foundation-model-style training.

Relational databases, however, contain structure that is not present in ordinary single tables. Predictive evidence may lie several joins away from the target row, and the relevant signal may involve temporal histories, linked entities, or multi-hop dependencies. One way to use tabular foundation models in this setting is to first transform the database into a single feature table. RDBLearn follows this strategy by aggregating relational neighborhoods into engineered features and then applying a tabular in-context learner [21]. This approach is effective and provides a strong baseline, but the flattening step changes the input representation: the model no longer reasons directly over cells, schemas, timestamps, primary–foreign-key links, or multi-table structure.

Large language models for relational data. Another line of work applies large language models to structured and relational data by serializing tables or database neighborhoods into text [44]. This has the advantage of using powerful pretrained language models without designing a specialized architecture. However, serialization can be inefficient and brittle. Relational structure must be expressed as text, long database contexts can exceed the model context window, and the resulting input

format may differ substantially from the data distribution seen during language-model pretraining. Hybrid approaches that combine graph or relational encoders with LLMs can partially address these issues, but they often still depend on task-specific components or limited context construction [45].

Graph foundation models. Several papers have paved the way toward graph foundation models. Mole-BERT [46] pretrains GNNs with masked atom prediction and triplet-based contrastive learning for node- and graph-level drug-discovery tasks. Beaini et al. [47] further show that molecular pretraining benefits from scale, using large multi-task datasets with billions of labels and demonstrating that combining quantum and biological supervision improves low-resource performance.

Recent work has extended pretraining graph models beyond molecules. PRODIGY [48] introduces prompt-based graph representations for in-context learning. GraphAny [49] studies zero-shot node classification across graphs with disjoint feature and label spaces, using LinearGNN ensembles and inductive attention inspired by linear least-squares estimation. ULTRA [50] learns transferable relational representations for zero-shot reasoning over unseen knowledge graphs. GraphGPT [51] converts graphs into reversible token sequences using Eulerian paths, enabling scalable generative Transformer pretraining for graph-structured data. Most recently, Bechler-Speicher et al. [52] scale the graph foundation model paradigm to heterogeneous billion-scale graphs, proposing a billion-parameter Transformer architecture and a recipe for link prediction pretraining.

D Additional Details on THE JOIN

D.1 Agentic execution

Constructing THE JOIN corpus manually would require months of human engineering, particularly for web collection as each source presents different download mechanisms, file formats, schema conventions, parsing failures, and data-quality issues. We therefore used Claude Code agents as the execution layer for corpus construction. The agents identify and inspect candidate sources, implement and debug domain-specific collectors, adapt standardization routines to newly observed edge cases, run the pipeline, inspect logs and intermediate artifacts, and iterate until each stage reaches its completion criteria. The agents were used for engineering and orchestration, while task validation, scoring, filtering, baseline computation, loadability checks, and provenance tracking were performed by deterministic scripts based on human-defined criteria.

D.2 Phase 1: Collection

Collecting from heterogeneous sources. The first phase builds a raw relational snapshot from a broad set of public data sources. This is difficult to automate with a single scraper because different sources expose data in very different ways: as SQL dumps, SQLite databases, CSV bundles, parquet or JSON exports, archive mirrors, or paginated APIs. We therefore use source-specific collectors. Each collector downloads the data, parses it in its native format, recovers table and key metadata when available, and converts the result into a common parquet-based representation. We also exclude databases that exactly overlap with established benchmark sources in RelBench, RelBench v2 and 4DBInfer, using source identity rather than schema similarity as the criterion.

Respecting upstream constraints. The collectors are designed to be reproducible while remaining respectful of the sources. API-based collectors follow the documented rate limits and include the required identifying information. Bulk-download sources are handled at the archive level, and for sources that publish periodic snapshots, the refresh cadence of our corpus is limited by the cadence of the upstream release.

Normalizing the collected data. Before any database is passed to later stages, the collector performs basic normalization. For example, (1) primary and foreign keys are converted to integer indices, (2) timestamps and booleans are stored in consistent types, and (3) mixed-type columns are either normalized or rejected. We also store primary-key, foreign-key, and time-column metadata alongside each Parquet table, so that downstream stages do not rely on source-specific assumptions. Table 4 lists the domains included in the corpus.

Table 4: Per-domain corpus breakdown across all collected domains, grouped by italic subject-area headers and sorted by descending DB count. *DBs*: raw count. The *License(s)* and *Ref.* cells list one (license, reference) pair per line with rows aligning across the two columns.

Domain	Source	DBs	License(s)	Ref.
Commerce & Online Platforms				
tech community	Stack Exchange / GH Archive / HN	75	No explicit license Open public API CC BY-SA 4.0	gharchive.org hn.algolia.com archive.org
recommendations	Amazon / MovieLens / IMDb / UCI	34	MIT (Amazon ToS) Non-commercial Research-use Research-only, no redistribution Research-only, no redistribution Public Domain CC BY 4.0	github.com data.imdb.com cseweb.ucsd.edu grouplens.org grouplens.org research.gov archive.ics.uci.edu
prediction market	Manifold / Polymarket	2	ToS-restricted ToS-restricted	manifold.markets polymarket.com
e-commerce	RecSys Challenge 2015	1	CC BY-NC-ND 4.0	kaggle.com
real estate	Zillow Research	1	Research-use	zillow.com
Sports, Culture & Media				
motorsport	Wikipedia / Racing-Reference / MotoGP	9	CC BY-SA 4.0 CC BY-SA 4.0 CC BY-SA 4.0; non-commercial CC BY-SA 4.0 Public unofficial API Non-commercial, no redistribution CC BY-SA 4.0	en.wikipedia.org en.wikipedia.org racing-reference.info en.wikipedia.org motogp.com racing-reference.info en.wikipedia.org
food	SNAP / Open Food Facts / USDA / Food.com	6	Research-use Research-use ODbL Public Domain Kaggle ToS	snap.stanford.edu snap.stanford.edu world.openfoodfacts.org fdc.nal.usda.gov kaggle.com
sports	ATP / Cricsheet / Lahman / StatsBomb	4	CC BY-NC-SA 4.0 CC0 CC BY-SA 3.0 StatsBomb user agreement	github.com cricsheet.org sabr.org github.com
music	FMA / Last.fm / ListenBrainz	3	CC BY 4.0 Research-use CC0	github.com grouplens.org listenbrainz.org
museum	Met / MoMA / Tate	3	CC0 CC0 CC0	github.com github.com github.com
gaming	EVE Online / Lichess / Steam	3	Open public API CC0 Research-use	developers.eveonline.com database.lichess.org cseweb.ucsd.edu
cultural	Europeana / Wikidata	2	CC0 CC0	europeana.eu wikidata.org
cultural surveys	D-PLACE / Pew	2	CC BY 4.0 Public-use data (Pew)	d-place.org pewresearch.org
podcast	Apple Podcasts (Kaggle)	1	CC BY-NC-SA 4.0	kaggle.com
history	Chronicling America	1	Public Domain	chroniclingamerica.loc.gov
Government, Finance & Energy				
US government	BTS / FDA / FDIC / FEC / HMDA / NHTSA	11	Public Domain	usa.gov
energy	BOEM / Ember / OPSD / PHMSA / Sodir / EIA / GeoNuclearData	10	Public Domain CC BY 4.0 CC BY 4.0 CC BY 4.0 Public Domain NLOD Public Domain ODbL	data.boem.gov github.com ember-energy.org open-power-system-data.org phmsa.dot.gov norge.no eia.gov github.com
international	Eurostat / GDELT / UNHCR / World Bank	5	Eurostat re-use Open (attribution) CC BY 4.0 CC BY 4.0	ec.europa.eu gdeltproject.org unhcr.org datacatalog.worldbank.org
finance	CFPB / SEC EDGAR / PKDD (Berka)	3	Public Domain Public Domain Research-use	consumerfinance.gov sec.gov relational.fel.cvut.cz
politics	Our World in Data / V-Dem	2	CC BY 4.0 CC BY 4.0	ourworldindata.org v-dem.net
civic	NYC OpenData (311)	1	Public Domain	opendata.cityofnewyork.us
crypto	CoinGecko	1	API ToS (free tier)	coingecko.com
labor	ILOSTAT	1	CC BY 4.0	ilostat.ilo.org
legal	OpenSanctions	1	CC BY-NC 4.0	opensanctions.org

continued on next page

(continued from previous page)

Domain	Source	DBs	License(s)	Ref.
patent	USPTO / PatentsView	1	CC BY 4.0	data.uspto.gov
UK public data	NHS prescriptions (OGL)	1	OGL v3.0	nationalarchives.gov.uk
Health & Public Services				
clinical	ChEMBL / openFDA	4	CC BY-SA 3.0 Public Domain	chembl.gitbook.io fda.gov
biology	ClinVar / GBIF / PubChem / Reactome	4	Public Domain CC BY 4.0 Public Domain CC0	ncbi.nlm.nih.gov gbif.org ncbi.nlm.nih.gov reactome.org
transport	GTFS feeds / OpenFlights / Citi Bike	3	Per-agency ODbL Free use	mobilitydata.org openflights.org citibikenyc.com
healthcare quality	CMS Provider Data	2	Public Domain	cms.gov
education	OULAD (Open University)	1	CC BY 4.0	research.stem.open.ac.uk
public health	WHO GHO	1	Open access	who.int
agriculture	USDA NASS	1	Public Domain	quickstats.nass.usda.gov
disaster	OpenFEMA	1	Public Domain	fema.gov
skills	O*NET	1	CC BY 4.0	onetcenter.org
Scientific & Reference				
text-to-SQL	Spider / BIRD / Spider 2.0 / LiveSQL	259	CC BY-SA 4.0 Public access (Oracle) MIT CC BY-SA 4.0	bird-bench.github.io livesql.oracle.com github.com yale-lily.github.io
CTU repository	CTU Prague Repository	114	Research-use MS sample DB MIT MySQL sample DB Research-use Research-use CC BY-SA 4.0 MS sample DB MS sample DB BSD TPC license TPC license TPC license TPC license	relational.fel.cvut.cz microsoft.com github.com mysqltutorial.org data.imdb.com grouplens.org sabr.org microsoft.com microsoft.com dev.mysql.com tpc.org tpc.org tpc.org tpc.org
temporal network	SNAP / KONECT	12	BSD CC BY 4.0	snap.stanford.edu zenodo.org
network	SNAP / KONECT (raw)	10	BSD	snap.stanford.edu
academic	OpenAlex / DBLP / CORDIS / NIH / Unpaywall	7	CC BY 4.0 CC BY 4.0 ODC-BY Public Domain CC0 CC0	europa.eu europa.eu dblp.org reporter.nih.gov openalex.org unpaywall.org
geospatial	SNAP / Foursquare / GeoNames / OSM / Overture	7	BSD Academic-use only CC BY 4.0 BSD ODbL CDLA-Permissive	snap.stanford.edu sites.google.com geonames.org snap.stanford.edu openstreetmap.org overturemaps.org
graph learning	OGB / FB15k / WN18RR / YAGO	5	MS research license ODC-BY WordNet license CC BY 3.0	microsoft.com github.com wordnet.princeton.edu yago-knowledge.org
FiveThirtyEight	FiveThirtyEight datasets	4	CC BY 4.0	github.com
ecology	iNaturalist / PBDB / RAM Legacy / GBIF	4	Various CC BY 4.0 CC BY 4.0 Mixed CC0 / CC BY / CC BY-NC	inaturalist.org paleobiodb.org ramlegacy.org gbif.org
linguistics	Glottolog / OpenSubtitles / WALS	3	CC BY 4.0 Restricted CC BY 4.0	glottolog.org opus.nlpl.eu wals.info
security	NVD / Exploit-DB / MITRE ATT&CK	3	Public Domain GPL 2.0 MITRE ATT&CK terms	nvd.nist.gov exploit-db.com github.com
astronomy	NASA Exoplanet Archive / SDSS	2	Public Domain Free use	ipac.caltech.edu sdss.org
forecasting	Good Judgment Project	1	CC0	dataverse.harvard.edu
hydrology	USGS Water Data	1	Public Domain	waterdata.usgs.gov
environment	EPA AQS	1	Public Domain	epa.gov
network infra	RIPE / RouteViews	1	Open data	ripe.net
ocean	NOAA NDBC	1	Public Domain	ndbc.noaa.gov
storm	NOAA Storm Events	1	Public Domain	ncei.noaa.gov
threat intel	URLhaus (abuse.ch)	1	abuse.ch ToS	urlhaus.abuse.ch

D.3 Phase 2: Standardization

Creating a common representation. After collection, each database is standardized into a uniform relational format. The purpose of this stage is to make heterogeneous sources loadable and comparable, without altering their semantic content. Each fix is implemented as an idempotent detector–transformer pass over the collected parquet files: a pass first checks whether the target invariant already holds, and only rewrites the data when needed.

Recovering relational structure. Many public datasets contain useful structure that is implicit rather than declared. We therefore recover primary keys, foreign keys, time columns, boolean fields, and numeric columns when they can be inferred reliably. These repairs are conservative: candidate keys must satisfy uniqueness checks, foreign keys must pass containment checks, and identifier-like fields such as zip codes, ISBNs, phone numbers, and barcodes are not coerced into numeric values.

Normalizing common type issues. The standardization pass resolves common inconsistencies across sources, such as booleans stored as strings, dates stored in mixed textual formats, year columns stored as integers, and numeric values stored as strings. Ambiguous cases are left unchanged or flagged rather than silently modified. This keeps the downstream interface simple while avoiding brittle source-specific assumptions.

Controlling scale. For databases that are too large for repeated downstream processing, we apply deterministic entity-axis subsampling: entities are sampled with a fixed seed, all associated activity rows are retained, and keys are reindexed to preserve relational consistency.

Preserving auditability. All standardization fixes are performed by mechanical scripts, and each transformation is governed by explicit rules. As a result, downstream task generation and evaluation operate on a consistent representation, while every repair remains reproducible and inspectable.

D.4 Phase 3: Task proposal and validation

Proposing tasks from schema context. For each eligible database, the proposer receives a structured summary of the database, including its domain, tables, row counts, key relationships, time columns, column types, basic column statistics, and a small number of example rows from activity tables. This context is wrapped in a fixed prompt that specifies the allowed task templates, the required JSON format, and the validation rules. The proposer also receives the names of previously generated tasks for that database, which helps reduce duplicates across passes.

Constraining the proposal space. Task proposals are restricted to a registered set of templates, listed in Table 5. This keeps the generated tasks interpretable and mechanically checkable, while still allowing broad coverage across domains. Proposals that refer to unknown templates are rejected before further processing.

Validating mechanically. The LLM proposes candidate tasks, but acceptance is decided by a deterministic validator. The validator checks that referenced tables and columns exist, entity columns are valid foreign keys, numeric fields are used where required, generated target names do not collide with existing columns, task names follow the required format, and binary thresholds avoid trivially imbalanced labels. Rejected proposals are logged with their failure reason and are not written to the released task set.

Improving diversity across passes. Each database is processed through multiple proposal passes. Early passes tend to recover natural or obvious tasks for the schema, while later passes explicitly target under-represented templates and harder prediction settings, such as rare positives, cold-start behavior, heavy-tailed regression, ranking, and derived-ratio tasks. This multi-pass strategy increases task diversity while preserving the same mechanical validation standard for every accepted proposal.

Keeping generation auditable. The proposal stage uses an LLM to suggest candidate tasks, but the final task set is determined by explicit templates and deterministic checks. This separation allows the pipeline to benefit from the LLM's ability to identify plausible prediction problems from heterogeneous schemas, while keeping task inclusion reproducible, inspectable, and independent of unverified free-form judgments.

```
Database: civic/nyc_311
Total: 4 tables, 501223 rows
Entity tables: ['agencies', 'locations', 'complaint_types']

Schema:
  agencies (15 rows pkey=agency_id):
    columns: agency(string), agency_name(string), agency_id(Int64, range=[0..14])
  locations (270 rows pkey=location_id):
    columns: borough(string), incident_zip(Int64, range=[8854..92108]), location_id(Int64,
     range=[0..269])
  complaints (500000 rows pkey=unique_key time_col=created_date
              fks={'complaint_type_id': 'complaint_types', 'agency_id': 'agencies', 'location_id':
     'locations'}):
    columns: unique_key(Int64), complaint_type_id(Int64), agency_id(Int64), location_id(Int64),
             created_date(datetime64[us, UTC]), closed_date(datetime64[us, UTC]),
             resolution_hours(Float64, mean=49.6, range=[0..1105]), status(string),
             community_board(string), open_data_channel_type(string)
  complaint_types (938 rows pkey=complaint_type_id):
    columns: complaint_type(string), descriptor(string), complaint_type_id(Int64, range=[0..937])

Activity tables (have time_col + FK to entity):
  complaints: time_col=created_date, entity_fks={'complaint_type_id': 'complaint_types',
              'agency_id': 'agencies', 'location_id': 'locations'}, 500000 rows
    sample: [{'unique_key': 0, 'agency_id': 0, 'created_date': '2026-04-14 02:33:29+00:00',
              'resolution_hours': None, 'status': 'Open', 'community_board': '15 BROOKLYN', ...}, ...]

Existing tasks for this DB (do NOT duplicate these):
  predict-agency-avg-resolution-hours, predict-location-max-resolution-hours,
  predict-agency-slow-resolution, predict-complaint-type-active,
  predict-agency-resolution-trend, ... and 24 more

TASK: Propose up to 15 high-quality forecasting tasks that a domain expert would find meaningful.
Each task predicts a future value for an entity over a time window.

PRIORITY: Our existing task suite is heavy on aggregate_numeric, threshold_binary, existence_binary,
count_events, churn_activity. Use at least 40% of your proposals from these under-represented
patterns (where applicable): time_until_next (REG), streak_length (REG), rank_change (CLF),
rank_regression (REG), peer_outperform (CLF), new_category_adoption (CLF), will_repeat_last (CLF),
trend_reversal (CLF), volatility_numeric (REG), concentration (REG), derived_ratio (REG).

If the schema supports a `group_col` (race/week/country) -> propose rank_change + rank_regression.
If there's a natural `peer_col` (category/country/segment) -> propose peer_outperform. If there's
a `category_col` (product/type/status) -> propose new_category_adoption + will_repeat_last.

ALLOWED PATTERNS:
  - aggregate_numeric (REG): mean/sum/max of numeric col next window. Needs `source_col`,
    `aggregation` in [mean, sum, max, min].
  - threshold_binary (CLF): aggregate exceeds threshold. Needs `source_col`, `aggregation`,
    `threshold`, `compare` in [gt, gte, lt, lte].
  - existence_binary (CLF): entity appears in event_table next window. Needs `event_table`,
    `event_entity_col`, `event_time_col`.
  ... [more templates omitted]
  - heavy_tail_regression (REG, HARD TIER): log1p(agg(source_col)) per entity next window.
    Entity pool = ALL historical entities. Matches rel-amazon/item-ltv, rel-hm/item-sales.

HARD REQUIREMENTS (validator rejects violations):
  1. entity_col MUST be a foreign key in the activity_table.
  2. source_col / numerator_col / denominator_col MUST be numeric columns in the activity_table.
  3. target_col must be short and not collide with an existing column.
  4. task_name must be kebab-case starting with "predict-".
  5. For derived_ratio: pick MEANINGFUL pairs (batting average, completion rate, margins).
  6. For threshold_binary: thresholds near the median of the column.
  7. Prefer real-world interpretability over mechanical column selection.

Return ONLY a JSON array of task objects:
[ { "task_name": "predict-...", "task_type": "REGRESSION" | "BINARY_CLASSIFICATION",
    "pattern": "aggregate_numeric" | ..., "entity_table": "...", "entity_col": "...",
    "activity_table": "...", "time_col": "...", "target_col": "...", "target_description": "...",
    ... template-specific fields ..., "skip_cols": [], "tables_used": [...],
    "timedelta_multiplier": 1 } ]
```

Figure 8: Full proposer prompt for `civic/nyc_311` (one representative database). The schema, FK map, sample rows, and existing-task list are populated per database; the `ALLOWED PATTERNS`, `HARD REQUIREMENTS`, and JSON contract blocks are constant across databases (the `ALLOWED PATTERNS` block in particular is abbreviated here—the full version enumerates all 25 templates with their required fields).

```
{
  "task_name": "predict-agency-avg-resolution-hours",
  "task_type": "REGRESSION",
  "pattern": "aggregate_numeric",
  "entity_table": "agencies",
  "entity_col": "agency_id",
  "activity_table": "complaints",
  "time_col": "created_date",
  "target_col": "avg_resolution_hours",
  "target_description": "Predict the mean resolution time (hours) for complaints handled by an agency
     in the next window.",
  "source_col": "resolution_hours",
  "aggregation": "mean",
  "tables_used": ["agencies", "complaints"],
  "timedelta_multiplier": 1
}
```


Figure 9: First accepted proposal for `civic/nyc_311`, returned by the Claude Opus proposer in response to the prompt of Figure 8. The proposal passes the mechanical validator (the `entity_col` is a declared FK in the `activity_table`, `source_col` is a numeric column of the same table, `target_col` does not collide with any existing column, and the `pattern` is registered) and is forwarded to the generator for instantiation.

Table 5: The 25 task-template families instantiated by the corpus, organized under italic signal-type headers. Each row gives the prediction target as the model sees it, paired with one concrete domain example. R: regression; C: binary classification.

Template	Type	Prediction target (example)
Aggregation—volume / intensity of activity		
`aggregate_numeric`	R	mean / sum of a numeric column over the next window *(e.g. a player's home runs next season)*
`count_events`	R	number of events for the entity in the next window *(e.g. orders per customer next month)*
`distinct_count`	R	distinct values of a column the entity touches *(e.g. distinct co-authors next year)*
`concentration`	R	entity's share of total activity *(e.g. seller's share of category sales)*
`derived_ratio`	R	ratio of two numeric columns over the next window *(e.g. new-to-returning customer ratio)*
Threshold / existence—continuous signal at a decision boundary		
`threshold_binary`	C	whether an aggregate exceeds a threshold *(e.g. revenue >$1M next quarter)*
`existence_binary`	C	whether the entity appears in the event table next window *(e.g. user posts at least once)*
`top_k_binary`	C	whether the entity is in the top $K\%$ on a metric *(e.g. artist enters top 1% of plays)*
Trend / change—direction and magnitude of change		
`trend_binary`	C	whether a metric rises relative to the current window *(e.g. weekly volume rises)*
`trend_reversal`	C	whether the direction of change flips between windows *(e.g. sales reverse from rising to falling)*
`change_numeric`	R	numeric delta from current to next window *(e.g. change in monthly energy consumption)*
`volatility_numeric`	R	standard deviation of a metric in the next window *(e.g. next-week price volatility)*
Behavior / churn—entity lifecycle		
`churn_activity`	C	entity has *no* activity in the next window *(e.g. subscriber cancels next month)*
`return_binary`	C	dormant entity reactivates next window *(e.g. dormant Stack Exchange user posts again)*
`time_until_next`	R	days until the entity's next event *(e.g. days until a player's next ATP match)*
`will_repeat_last`	C	next event's category equals the most-recent one *(e.g. customer re-buys the same product)*
`first_event_binary`	C	entity experiences a first occurrence next window *(e.g. first purchase by a new account)*
Ranking—relative position within a peer group		
`rank_change`	C	entity's rank within its group improves by $\geq K$ *(e.g. driver climbs ≥5 places)*
`rank_regression`	R	numeric rank within the group ($1 =$ best) *(e.g. predict the driver's next-race finish position)*
`peer_outperform`	C	entity beats the mean of its peer group *(e.g. artist beats peer-group mean plays)*
Adoption / novelty—first-time events		
`new_category_adoption`	C	entity touches a category value it never has before *(e.g. first purchase from a new product category)*
Streak—resilience as a single metric		
`streak_length`	R	longest run of consecutive active sub-periods *(e.g. consecutive seasons a player remains active)*
Tail-event (hard-tier)—rare and extreme regimes		
`cold_start_binary`	C	newly-appearing entity has repeat activity in its first window *(e.g. new user posts within 24h of joining)*
`rare_positive_binary`	C	rare event ($\leq 10\%$ positive rate) occurs for the entity *(e.g. rare adverse-quality hospital event)*
`heavy_tail_regression`	R	$\log(1+x)$ of an aggregate x; preserves heavy upper tail *(e.g. $\log(1+$ next-month spend$)$, large spenders preserved)*

D.5 Phase 4: Baselines and quality filtering

Measuring basic task signal. After task generation, each task is evaluated with a small set of baselines. We first compute two cheap historical baselines: an entity-mean predictor, which uses each entity's average training label, and a last-value predictor, which uses the entity's most recent observed label when available. These baselines are inexpensive and provide a simple check for whether the task contains recoverable temporal or entity-level signal.

Training a lightweight ML baseline. We also train a fixed XGBoost baseline for each task. For classification tasks we use `XGBClassifier`, and for regression tasks we use `XGBRegressor`, with `n_estimators=100`, `max_depth=6`, `learning_rate=0.1`, `tree_method="hist"`, and `n_jobs=2`. Features are simple window aggregations over the activity table: for the 20 most populous numeric columns, we compute `count`, `mean`, `sum`, `max`, and `min` within a lookback window of length $5 \times$ `timedelta` ending at the prediction time. To keep the pass scalable, training is capped at $5{,}000$ rows per task and evaluation at $2{,}000$ rows per task. This baseline is not intended to be a strong relational model and is only intended to provide a consistent and reproducible reference point for filtering tasks across domains.

Filtering and difficulty assignment. The quality filter removes tasks that are unlikely to be useful for downstream evaluation, such as tasks that are nearly trivial, exhibit severe distribution drift, or fail during baseline fitting. Tasks with non-trivial baseline signal are retained, and we also retain non-trivial tasks for which XGBoost does not recover signal, since stronger relational models may still exploit structure missed by the lightweight baseline. Retained tasks are assigned a coarse difficulty level from their XGBoost score: easy, medium, or hard.

Preparing the release format. After loadability validation, each task is converted into the release format. The train, validation, and test splits are concatenated in chronological order into a single time-ordered training file, while preserving the schema metadata required by downstream loaders.

D.6 Asset licenses, Citations, and Compliance

Compliance practice. Every contributing source is either (a) a bulk archive that the upstream provider publishes for download, or (b) a public API that the provider documents for programmatic access. The terms of these sources differ: most are public domain or under permissive or share-alike licenses, but some restrict use to research, to non-commercial purposes, or to the provider's own terms of service, and these restrictions carry over to the corresponding databases in **THE JOIN**. We do not crawl arbitrary websites, do not bypass paywalls, do not ignore `robots.txt`, and do not scrape sources whose terms of service forbid automation.

Released-corpus license. The corpus assembly (schemas, standardization and generated tasks) is released under **CC-BY-SA 4.0**, but each database remains subject to the terms of its upstream source, which take precedence where they are more restrictive. Per-database metadata records (i) the upstream source identifier and URL and (ii) the upstream license string verbatim. Restrictions inherited from the sources include non-commercial use only (e.g. ATP Tennis, Apple Podcasts, OpenSanctions, IMDb, Racing-Reference), no derivatives (RecSys Challenge 2015), research or academic use only (e.g. the CTU repository, MovieLens, Foursquare, and the McAuley Lab datasets), and provider terms of service (e.g. Manifold, which prohibits commercial AI/ML training, Polymarket, StatsBomb, CoinGecko, URLhaus and OpenSubtitles). Users of the corpus, and of models trained on it, must check and honor these per-source restrictions, particularly for commercial use.

PII. The corpus contains no personally-identifying information beyond what each upstream source has already published under its own license. Where source data uses pseudonymized identifiers (e.g. Stack Exchange user IDs, Lahman player IDs), we preserve those identifiers verbatim and do not attempt to de-anonymize or join across sources.

D.7 Case Studies

To illustrate what a forecasting task in **THE JOIN** represents and the diversity of the corpus, we present three case studies from distinct real-world domains. In every case, the prediction target is a quantity monitored in the corresponding domain and instantiates a different label family among the twenty-five task patterns defined in the corpus.

Forecasting an account's peak balance (`join-berka`) Our first example predicts a continuous quantity from a transaction stream. The Berka dataset from the PKDD'99 Challenge is a complete relational snapshot of a Czech retail bank, including its accounts, the clients who hold them, the permanent orders and loans they carry, and a central ledger of just over a million individual transactions, each stamped with the resulting running balance. Its eight tables form a multi-hop schema (Figure 10), where `accounts` are linked to their `clients` through `disp`, and both `accounts` and `clients` are linked to the `district` in which they are registered. As peak balance is a quantity that retail banks monitor directly, we create the `account-max-balance` regression task that, for each account and seed time, predicts the highest balance reached during the following window of roughly 17.5 days.

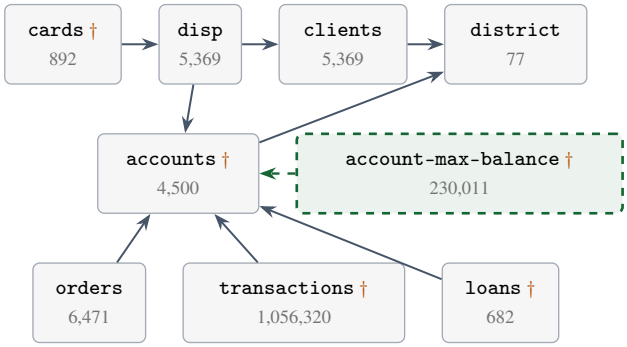


Figure 10: Schema of `join-berka`: every table, its row count, and the foreign key links. Arrows point from a foreign key to the table it references; † marks a table carrying a time column. The dashed box is the task table derived from the database.

Forecasting new-user retention (`join-brightkite`) A cold-start task evaluates an entity at its first appearance. This setting is of independent interest because it isolates a model's ability to generalize from relational context rather than from the entity's own past. To showcase such a task in THE JOIN, we consider Brightkite, a location-based social network in which users check in at venues and form friendships. The `newuser-will-return` task considers each user in the window of their first-ever check-in and predicts whether they will check in again within the next two windows, each roughly 6.7 days. Because the user's own label history is unavailable, a model cannot exploit autocorrelation and must instead generalize from side information, including the venues visited and the user's position in the friendship graph.

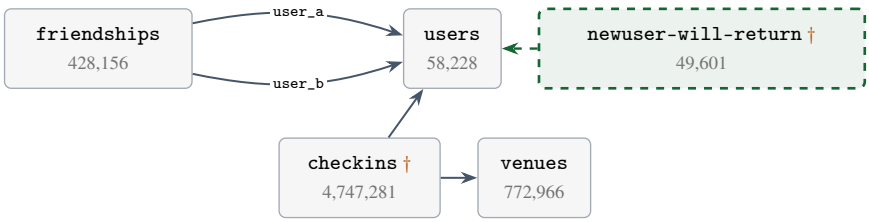


Figure 11: Schema of `join-brightkite`. Here `friendships` and `users` tables are joined by more than one foreign key, so the edges are drawn separately and labeled with the referencing column.

Forecasting who beats their cohort (`join-oulad`) We consider the Open University Learning Analytics Dataset (OULAD), a relational record of a large online university that links each student to their virtual-learning-environment clickstream and assessment scores. The `student-score-beats-cohort` task predicts whether a student's mean assessment score in the next window (of roughly 7.2 days) will exceed the mean score of their presentation cohort over the same window. The comparison threshold is therefore the cohort average rather than a fixed constant, and must be computed before the label can be assigned. Because the label is defined relative to a peer aggregate that changes over time, a model cannot score each student in isolation. Instead, it must compute the cohort context at each step and track the student's trajectory across windows.

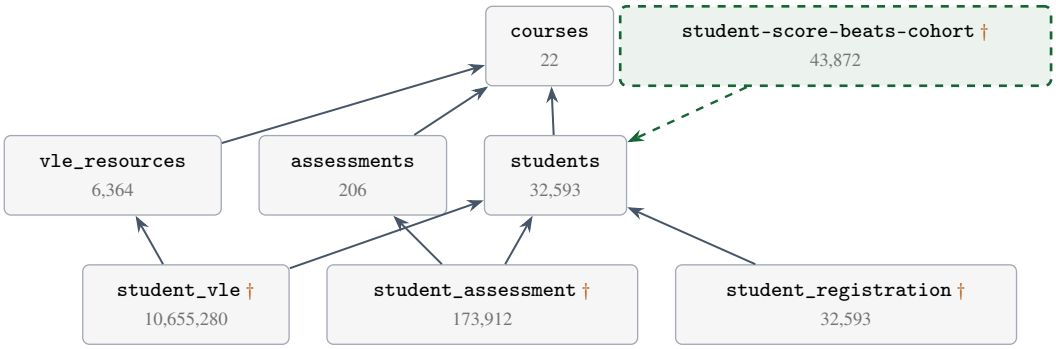


Figure 12: Schema of `join-oulad`.

Table 6: Binary classification forecasting tasks in RelBench.

Dataset	Task	Entity	Prediction target
rel-amazon	user-churn	User	Whether the user does not review any product in the next 3 months.
rel-amazon	item-churn	Item	Whether the item receives no reviews in the next 3 months.
rel-avito	user-visits	User	Whether the user visits more than one ad in the next 4 days.
rel-avito	user-clicks	User	Whether the user clicks more than one ad in the next 4 days.
rel-f1	driver-dnf	Driver	Whether the driver does not finish a race in the next month.
rel-f1	driver-top3	Driver	Whether the driver qualifies in the top 3 for a race in the next month.
rel-hm	user-churn	User	Whether the customer makes no transactions in the next week.
rel-stack	user-engagement	User	Whether the user makes any votes, posts, or comments in the next 3 months.
rel-stack	user-badge	User	Whether the user receives a new badge in the next 3 months.
rel-trial	study-outcome	Study	Whether the trial achieves its primary outcome in the next year.
rel-event	user-repeat	User	Whether a previously active user attends another event in the next 7 days.
rel-event	user-ignore	User	Whether the user ignores more than two event invitations in the next 7 days.

D.8 Manual Audit of Generated Tasks

The deterministic checks of Section 3.3 ensure that tasks are valid against the schema, but not that they are meaningful. We therefore manually reviewed the LLM task descriptions and label-generation SQL, and found the following.

Targets are semantically named quantities. In `join-berka`, tasks ignore the meaningless `a2`–`a16` columns of `district` and instead predict `max_balance`, `goes_negative`, `days_to_next_txn`, etc.

Label SQL uses domain concepts, not arbitrary aggregations. In `join-oulad`, `student-score-beats-cohort` compares a student to their cohort's contemporaneous mean. In `join-citi-bike`, `station-time-until-next-trip` predicts station idle time, which operators need for inventory planning.

Tasks match real practitioner decisions. `account-max-balance` (`join-berka`) mirrors overdraft-exposure and credit-limit review; `newuser-will-return` (`join-brightkite`) is a standard onboarding-retention metric; `movie-new-tag-adoption` (`join-movielens`) tracks metadata coverage.

A small number of tasks (about 100) did aggregate non-semantic columns, e.g., `student-feature1-to-feature0-ratio` in `join-act_mooc`.

E RelBench Forecasting Tasks

Binary classification tasks. The binary classification tasks in RelBench ask whether a future event or condition will occur for a given entity. They cover churn, future engagement, ad-click behavior, race outcomes, clinical outcomes, and event attendance behavior. See Table 6 for task descriptions.

Regression tasks. The regression tasks ask for a continuous or count-valued future outcome. They cover lifetime value, item sales, click-through rate, future votes, race position, adverse-event counts, site success, and event attendance. See Table 7 for task descriptions.

Table 7: Regression forecasting tasks in RelBench.

Dataset	Task	Entity	Prediction target
rel-amazon	user-ltv	User	Value of the products the user buys and reviews in the next 3 months.
rel-amazon	item-ltv	Item	Value of the purchases and reviews the item receives in the next 3 months.
rel-avito	ad-ctr	Ad	Click-through rate for the ad over the next 4 days.
rel-f1	driver-position	Driver	Average finishing position of the driver across races in the next 2 months.
rel-hm	item-sales	Item	Total sales for the article in the next week.
rel-stack	post-votes	Post	Number of votes the post receives in the next 3 months.
rel-trial	study-adverse	Study	Number of affected patients with severe adverse events in the next year.
rel-trial	site-success	Site	Success rate of a trial site in the next year.
rel-event	user-attendance	User	Number of events the user responds yes or maybe to in the next 7 days.

E.1 Hard Regimes in RelBench

THE JOIN places special emphasis on rare-event, cold-start, heavy-tailed and long-horizon tasks. RelBench is too small to serve as a pretraining corpus, but its tasks already cover these regimes (Table 8).

Table 8: RelBench tasks in each hard regime. A task can belong to several regimes.

Regime	RelBench tasks	Count
rare event	rel-avito/user-clicks, rel-stack/user-badge, rel-stack/user-engagement	3
cold start	rel-trial/study-outcome, rel-trial/study-adverse, rel-f1/driver-position, rel-f1/driver-dnf, rel-f1/driver-top3, rel-trial/site-success, rel-avito/user-clicks	7
heavy-tailed target	rel-amazon/item-ltv, rel-trial/study-adverse, rel-hm/item-sales, rel-stack/post-votes, rel-amazon/user-ltv	5
long horizon	rel-trial/study-outcome, rel-trial/study-adverse, rel-trial/site-success	3

Rare events. Three RelBench tasks sit at or below 5% positive: rel-avito/user-clicks (3.9%), rel-stack/user-badge (4.8%), and rel-stack/user-engagement (5.0%), comparable to the rare end of THE JOIN.

Cold start. rel-trial/study-outcome and rel-trial/study-adverse are 100% cold start at test time: they key on nct_id, each trial contributes one row, so no test trial appears in training. This is the most extreme cold-start setting in the suite, beyond the median of THE JOIN's cold-start tasks. rel-f1/driver-position (79%), rel-f1/driver-dnf and rel-f1/driver-top3 (71% each), rel-trial/site-success (63%), and rel-avito/user-clicks (61%) follow.

Heavy tails. rel-amazon/item-ltv has target skew 133 and kurtosis 29,079; rel-trial/study-adverse has skew 39; rel-hm/item-sales, rel-stack/post-votes, and rel-amazon/user-ltv sit near skew 21. About 140 tasks in THE JOIN are tail-dominated (the top 1% of values hold more than 30% of the mass); the heavy-tailed RelBench tasks are extreme versions of these.

Long horizon. The three rel-trial tasks forecast a full year ahead. The rel-amazon and rel-stack tasks follow with 91-day windows, while the remaining tasks look 4–60 days ahead.

F Relational Transformer Architecture

The **RT-J** model comprises 12 transformer blocks with hidden dimension $d = 512$, $H = 8$ attention heads of dimension $d_h = 64$, and feed-forward dimension $d_{\text{ff}} = 2048$. Each block applies column, feature, and neighbor attention sequentially, followed by a gated feed-forward network. The resulting model contains 86M trainable parameters, excluding the frozen text encoder.

Cell embeddings. Each cell i is associated with a value v_i, semantic type τ_i, column c_i, table t_i, and row r_i. Boolean values are cast to $\{0, 1\}$ and treated as numeric; numeric values are standardized per column, while datetime values use a shared database-wide normalization. Text values and schema names are embedded using the frozen `all-MiniLM-L12-v2` encoder $\mathcal{E} : \text{Text} \to \mathbb{R}^{384}$. The additional schema-based textual information for each cell is given by:

$$\mathbf{s}_i = \mathcal{E}\big(c_i \,\|\, \texttt{ of } \,\|\, t_i\big), \tag{2}$$

where $\|$ denotes string concatenation. For example: `"age of user"`. Let $\mathbf{u}_i$ denote the normalized scalar or text embedding representing the cell value. The initial token embedding is

$$\mathbf{e}_i^{\text{schema}} = \text{RMSNorm}_s\left(W_s \mathbf{s}_i + \mathbf{b}_s\right), \tag{3}$$

$$\mathbf{e}_i^{\text{value}} = \begin{cases} \mathbf{m}_{\tau_i}, & \text{if cell } i \text{ is masked,} \\ \text{RMSNorm}_{\tau_i}\left(W_{\tau_i} \mathbf{u}_i + \mathbf{b}_{\tau_i}\right), & \text{otherwise,} \end{cases} \tag{4}$$

$$\mathbf{x}_i^{(0)} = \mathbf{e}_i^{\text{schema}} + \mathbf{e}_i^{\text{value}} \in \mathbb{R}^{512}. \tag{5}$$

Value encoders and learned mask embeddings $\mathbf{m}_{\tau_i}$ are shared across columns of the same semantic type. Primary and foreign keys specify connectivity rather than contributing value embeddings. No positional or learned row-ID embeddings are used in the model.

Relational attention. Each of the 12 transformer blocks sequentially applies column, feature, and neighbor attention, with 8 heads per module. Let $\mathcal{P}(r)$ denote the parent rows referenced by the foreign keys of row r. For non-padding cells, the masks are

$$M_{ij}^{\text{col}} = \mathbb{I}[c_i = c_j \wedge t_i = t_j], \tag{6}$$

$$M_{ij}^{\text{feat}} = \mathbb{I}[r_i = r_j \vee r_j \in \mathcal{P}(r_i)], \tag{7}$$

$$M_{ij}^{\text{nbr}} = \mathbb{I}[r_i \in \mathcal{P}(r_j)]. \tag{8}$$

These modules respectively exchange information within columns, within rows and their referenced parents, and with child rows referencing the query row. Padding cells are excluded from all attention masks. Each module receives pre-normalized inputs $\mathbf{z}_i = \text{RMSNorm}_m(\mathbf{x}_i)$, with a separate RMSNorm per module m. For head h, bias-free projections and QK normalization [26] give:

$$\widehat{\mathbf{q}}_{ih} = \text{RMSNorm}_Q\left(W_{Q,h} \mathbf{z}_i\right), \tag{9}$$

$$\widehat{\mathbf{k}}_{jh} = \text{RMSNorm}_K\left(W_{K,h} \mathbf{z}_j\right), \tag{10}$$

$$\mathbf{v}_{jh} = W_{V,h} \mathbf{z}_j. \tag{11}$$

Queries and keys are RMS-normalized along the head dimension, with separate learned feature-wise gains shared across heads. Normalized queries are additionally multiplied by a learned head-specific scalar and the logarithm of the valid-key count. For a module with mask M, the attention logits are

$$k_i = \sum_j M_{ij}, \tag{12}$$

$$\ell_{ijh} = \begin{cases} \dfrac{\alpha_h \log(\max(k_i, 1))}{d_h} \widehat{\mathbf{q}}_{ih}^{\top} \widehat{\mathbf{k}}_{jh}, & M_{ij} = 1, \\ -\infty, & M_{ij} = 0, \end{cases} \tag{13}$$

where α_h is a learned head-specific scale. For queries with at least one accessible key,

$$a_{ijh} = \text{softmax}_j(\ell_{ijh}), \tag{14}$$

$$\mathbf{o}_i = \text{Concat}_{h=1}^{8}\left(\sum_j a_{ijh} \mathbf{v}_{jh}\right), \tag{15}$$

$$\mathbf{g}_i = 2\sigma(W_G \mathbf{z}_i), \tag{16}$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + W_O\left(\mathbf{g}_i \odot \mathbf{o}_i\right). \tag{17}$$

Queries with no accessible keys receive a zero attention contribution. Each attention module has independent parameters. After the three attention modules, a pre-normalized residual SwiGLU network completes the block:

$$\mathbf{z}_i = \mathrm{RMSNorm}_{\mathrm{ffn}}(\mathbf{x}_i), \tag{18}$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + W_2\left[\mathrm{SiLU}(W_1\mathbf{z}_i) \odot W_3\mathbf{z}_i\right], \tag{19}$$

with intermediate dimension 2048. All attention and feed-forward projections are bias-free, and all RMSNorm operations use $\epsilon = 10^{-6}$.

Prediction heads. A final RMSNorm and type-specific affine decoders produce

$$\mathbf{h}_i = \mathrm{RMSNorm}_{\mathrm{out}}\left(\mathbf{x}_i^{(12)}\right), \tag{20}$$

$$\widehat{\mathbf{u}}_i = D_{\tau_i}\mathbf{h}_i + \mathbf{b}_{\tau_i}^{\mathrm{out}}. \tag{21}$$

Numeric and datetime heads predict normalized scalars, and the text head predicts the 384-dimensional text embedding. All targets, including binary classification labels, are regressed with the Huber loss ($\delta = 1$, averaged over output dimensions) as ℓ_{τ_i}. For B contexts with target sets $\mathcal{T}_b$, the objective is

$$\mathcal{L} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{\max(|\mathcal{T}_b|, 1)}\sum_{i\in\mathcal{T}_b}\ell_{\tau_i}\left(\widehat{\mathbf{u}}_i, \mathbf{u}_i\right). \tag{22}$$

All prediction-head weights and biases are initialized to zero. The complete model contains 85,562,530 trainable parameters, excluding the frozen text encoder.

G Pretraining Details

In all our experiments we use an 86M-parameter RT model with 12 layers, 512 hidden size and 8 attention heads, incorporating the improvements of [26] (such as QK-Norm) over the original architecture of [23]. We sample context size L from $\mathcal{L} = \{512, 1024, 2048, 4096, 8192\}$, the local context size ℓ from $\mathcal{L}_\ell = \{128, 256, 512, 1024, 2048, 4096, 8192\}$, the BFS width-bound b from $\mathcal{B} = \{8, 16, 32, 64, 128, 256\}$, and the recency parameter ρ from $\mathcal{R} = \{0, 1\}$. The random-walk retriever uses $W = 10{,}000$ walks of length $K = 20$, and only the target cell is masked ($p_{\max} = 0$).

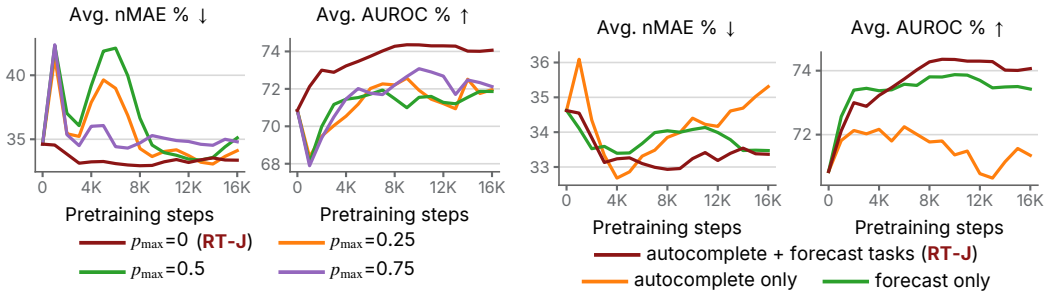
We pretrain in two phases: first on the synthetic PluRel corpus [26] (86,211 tasks over 1,900 databases), then continuing from that checkpoint on a fixed mixture of 13,243 **THE JOIN** tasks drawn from the 523 databases whose *preprocessed* on-disk footprint stays under 5 GB. Each phase is early-stopped once neither mean validation AUROC nor mean validation nMAE (of the live or SWA weights) has improved for 10K optimizer steps (at 18K steps for PluRel and 26K for **THE JOIN**), at a global batch size of 1024 contexts per step using Muon [36] for 2D weight matrices and AdamW for the remaining parameters, at a constant learning rate of 5×10^{-4} (RMS-matched for Muon as in [53]) after a linear warmup over the first 2000 steps. To compensate for a lack of decay phase [54], and following the recommendation in [38], we track the stochastic weight average (SWA) performance during training and select the best SWA checkpoint based on validation performance. Specifically, we use an exponential moving average (EMA) for SWA, with momentum 0.9995, which gives an effective window of $\approx$2000 steps, and find that it produces more stable curves with better results.

Each phase takes about a day or less on a single node with 8×A100 GPUs, with BFloat16 activations and FP32 master weights [55]. The released **RT-J** checkpoint is the SWA weights after 9K steps of the **THE JOIN** phase, selected by mean validation AUROC (74.36); selection runs over the whole SWA trajectory, so the selected step lies well before the step at which the phase stopped. The same single checkpoint is used for every task, classification and regression alike.

H Pretraining Ablations

H.1 Multi-cell masking rate

The original RT recipe [23] masks only a single target cell in a given context of length L. Following prior work on masked pretraining and meta-learning [56–59], the framework also supports masking non-target cells with a hierarchical Bernoulli scheme: per training context i a rate



(a) Multi-cell masking ablation

(b) Pretraining task-mix ablation

Figure 13: Pretraining ablations: multi-cell masking rate and task mix, as validation nMAE / AUROC over pretraining steps. Each arm differs from **RT-J**'s own pretraining run in one knob. (a) Masking non-target cells at any rate $p_{\max} > 0$ costs both metrics; **RT-J** masks only the target cell. (b) Forecasting tasks alone nearly match the mixture on classification but lose on regression, and autocompletion alone trails on both; the mixture is competitive on both.

$p_i \sim \mathrm{Unif}\big([0, p_{\max}]\big)$ is drawn, and each non-target, non-missing cell is masked independently with that rate $m_{i,j} \sim \mathrm{Bernoulli}(p_i)$; **RT-J** is pretrained with $p_{\max} = 0$, i.e., only the target cell is masked.

The hyperparameter $p_{\max}$ of the hierarchical Bernoulli masking rate controls how many non-target cells are masked, and thus how much supervision each context carries. To test whether denser supervision would have been better, we sweep $p_{\max}$ over $\{0, 0.25, 0.5, 0.75\}$, with identical data and context-sampling settings, all arms warm-started from the same checkpoint.

Observations. Figure 13a shows that masking non-target cells hurts both metrics. Over the common window of 10K–18K pretraining steps, single-cell masking averages 74.2 validation AUROC and 33.3 nMAE, while every masked rate is 1.6–2.6 AUROC points and 0.6–1.5 nMAE behind (71.6/33.9 at $p_{\max} = 0.25$, 71.6/34.0 at 0.5, and 72.5/34.8 at 0.75); the gap holds at each arm's own best step (74.4/32.9 against a best of 73.1/33.1 across the masked arms). Masked non-target cells corrupt the very context the model predicts from, and at this model and data scale that cost is not repaid by the extra supervision per step. Denser supervision is therefore not free: it buys gradient signal by degrading the in-context evidence, and at a fixed step budget the trade is a losing one.

H.2 Pretraining task mix

THE JOIN supplies two families of pretraining tasks, forecasting and autocompletion (Section 3), and **RT-J** is pretrained on both. We ablate the mixture by pretraining on each family alone, holding everything else fixed.

Observations. From Figure 13b, the mixture is the only setting competitive on both metrics. Over the same 10K–18K window, the mixed objective reaches 74.2 AUROC and 33.3 nMAE; forecasting alone is close on classification (73.6) but 0.4 nMAE behind, and autocompletion alone trails badly on classification (71.2, three points down) while also being the worst on regression (34.8). The single-family arms are not simply slower: autocompletion alone posts the best regression value of any arm early (32.7 nMAE at 4K steps) and then degrades steadily, which is what one expects when the pretraining distribution stops resembling the RelBench forecasting tasks the model is evaluated on. Mixing the two families is what keeps both metrics improving together, so we keep it in the recipe.

H.3 Compute scaling

Figure 14 tracks validation nMAE and AUROC over pretraining for three runs with the same recipe: from scratch on synthetic PluRel data only, from scratch on **THE JOIN** only, and continued pretraining of RT-PluRel on **THE JOIN**, which yields **RT-J**. Real data scales better than synthetic data: the PluRel-only run stops improving after about 8K steps and degrades on regression thereafter, while the **THE JOIN**-only run keeps slowly improving regression up to 32K steps. Continued pretraining from

Algorithm 1 Random-walk context construction (one task item).

Require: target row $v^\star$ in table τ with timestamp $t^\star$; database $\mathcal{D}$; total cell budget L; per-seed cell budget ℓ; BFS width-bound b; recency flag $\rho \in \{0, 1\}$; walks W; walk length K.
Ensure: cell sequence $\mathbf{x}$ with $|\mathbf{x}| \leq L$.
Stage 1: score same-table candidates by random-walk visits.
$\text{score}[u] \leftarrow 0$ for every row u.
for $w = 1$ **to** W **do**
  $u \leftarrow v^\star$.
  for $k = 1$ **to** K **do**
    Move u to a uniformly random PK/FK neighbor whose timestamp is $\leq t^\star$; if no such neighbor exists, end this walk.
    if $u \neq v^\star$ **and** u is in table τ **then**
      $\text{score}[u] \leftarrow \text{score}[u] + 1$.
    end if
  end for
end for
Stage 2: launch BFS from the ordered candidates.
If $\rho = 1$, order same-table rows by (timestamp, score) lexicographically descending; otherwise order by score descending.
$\mathbf{x} \leftarrow$ empty cell sequence.
for each row s in this order **do**
  if $|\mathbf{x}| \geq L$ **then break**.
  end if
  Append $\text{BFSCollect}(\mathcal{D}, s, \ell, b, t^\star, L - |\mathbf{x}|)$ to $\mathbf{x}$.
end for
return $\mathbf{x}$.

RT-PluRel reaches its best validation metrics within about 8K steps and stays ahead of pretraining on THE JOIN from scratch on regression throughout, and on classification at all but a few steps around 24K, so most of its benefit arrives early.

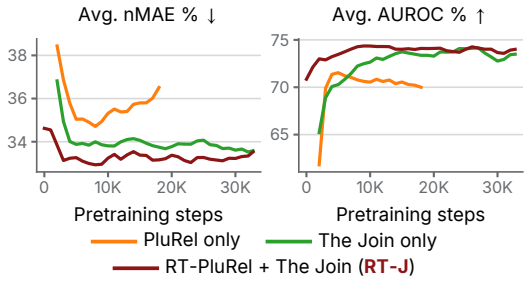


Figure 14: Validation nMAE and AUROC (SWA, 8192-cell context) over pretraining steps. The two runs from scratch are shown from step 2K; the PluRel-only run stops early at 18K steps.

I Random-Walk Context Construction

This appendix gives the full procedure for context construction summarized in Section 4. For one task item, we score same-table candidate rows by random-walk visit counts (Stage 1), then consume the candidates in order—most recent first with visit counts breaking ties when the recency flag $\rho = 1$, or by visit count alone when $\rho = 0$—as launch points for the width-bounded BFS cell collector BFSCollect of the original RT, which we reuse unchanged (Stage 2). Walks step uniformly at random along PK and FK edges, restricted to neighbor rows whose timestamp does not exceed the target's timestamp $t^\star$; this enforces temporal validity at the walk level and avoids leaking future information into the context. Only visits to rows other than the target in the target's table τ contribute, so the score ranks same-table rows by their personalized random-walk proximity to the target.

I.1 Sensitivity to random-walk parameters

The random-walk retriever of Appendix I uses $W = 10{,}000$ walks of length $K = 20$ by default. We evaluate **RT-J** with the default and three cheaper settings on the 8192-row test subsample at $L = 8192$, over three context seeds. Table 9 breaks down the per-example inference cost and Tables 10 and 11 the accuracy. Fewer or shorter walks cut the context-sampling cost from 28.1 to 14.3–17.9 ms per example, while the forward pass stays at about 10 ms. Classification degrades steadily as the walks shrink, from 72.9 to 71.9 mean AUROC, driven by `rel-trial/study-outcome` (59.4 to 52.7–55.8), `rel-stack/user-engagement` and `rel-f1/driver-dnf`. Regression is insensitive on average (33.3–33.6 mean nMAE), although individual tasks move in both directions: `rel-hm/item-sales` improves with shorter walks (14.1 to 11.6 nMAE at $(10\text{K}, 10)$), while `rel-f1/driver-position` is best at the default. We keep the default.

Table 9: Per-example inference cost in ms at batch size 1 and $L = 8192$, on one B200 GPU with one AMD EPYC 9565 CPU: per-task means over 100 timed steps after 5 warmup steps, then mean $\pm$ std. over the 21 RelBench tasks.

Stage	$(10\text{K}, 20)$	$(10\text{K}, 10)$	$(1\text{K}, 20)$	$(1\text{K}, 10)$
Context sampler (CPU)	28.06 ± 10.39	17.87 ± 8.14	14.31 ± 7.99	16.00 ± 8.49
Forward pass (GPU)	10.03 ± 0.38	9.85 ± 0.42	9.81 ± 0.43	9.78 ± 0.45
Total	38.09 ± 10.31	27.72 ± 8.12	24.11 ± 7.99	25.78 ± 8.52

Table 10: AUROC (%, higher is better) of **RT-J** for (W, K) on the 8192-row test subsample at $L = 8192$, mean $\pm$ std. over 3 context seeds; the mean row averages the tasks per seed. Bold: within one standard deviation of the best per row.

Task	$(10\text{K}, 20)$	$(10\text{K}, 10)$	$(1\text{K}, 20)$	$(1\text{K}, 10)$
`rel-amazon/item-churn`	76.67 ± 0.05	76.69 ± 0.06	$\mathbf{76.79} \pm 0.12$	$\mathbf{76.91} \pm 0.14$
`rel-amazon/user-churn`	66.88 ± 0.10	$\mathbf{67.05} \pm 0.11$	66.69 ± 0.14	66.50 ± 0.12
`rel-avito/user-clicks`	$\mathbf{52.31} \pm 0.98$	$\mathbf{52.83} \pm 2.65$	$\mathbf{53.14} \pm 3.41$	$\mathbf{53.87} \pm 2.69$
`rel-avito/user-visits`	61.70 ± 0.22	$\mathbf{62.68} \pm 0.62$	$\mathbf{62.50} \pm 0.62$	$\mathbf{62.42} \pm 0.59$
`rel-event/user-ignore`	$\mathbf{79.91} \pm 1.95$	$\mathbf{79.55} \pm 2.20$	$\mathbf{78.94} \pm 2.16$	77.71 ± 0.84
`rel-event/user-repeat`	$\mathbf{77.91} \pm 1.13$	$\mathbf{77.48} \pm 1.24$	$\mathbf{77.07} \pm 1.23$	75.70 ± 1.73
`rel-f1/driver-dnf`	$\mathbf{82.82} \pm 0.17$	82.43 ± 0.03	81.90 ± 0.13	81.40 ± 0.09
`rel-f1/driver-top3`	$\mathbf{90.75} \pm 0.45$	$\mathbf{91.18} \pm 0.40$	$\mathbf{90.86} \pm 0.26$	$\mathbf{90.95} \pm 0.27$
`rel-hm/user-churn`	62.31 ± 0.21	62.03 ± 0.21	61.76 ± 0.22	$\mathbf{63.48} \pm 0.03$
`rel-stack/user-badge`	$\mathbf{79.74} \pm 0.45$	$\mathbf{79.55} \pm 0.57$	$\mathbf{79.75} \pm 0.71$	$\mathbf{79.43} \pm 0.58$
`rel-stack/user-engagement`	$\mathbf{84.73} \pm 0.43$	82.60 ± 0.55	83.18 ± 0.37	82.12 ± 0.39
`rel-trial/study-outcome`	$\mathbf{59.36} \pm 1.72$	55.83 ± 1.15	52.95 ± 2.20	52.74 ± 3.00
Mean	$\mathbf{72.92} \pm 0.27$	72.49 ± 0.14	72.13 ± 0.28	71.93 ± 0.34

J Additional and Per-Task Results

J.1 Baselines with extended context

Figure 15 repeats the aggregate baselines comparison of Figure 3 on a fixed 8192-row test subsample per task, extending all four baselines to a 131K-cell context ($\sim$4,500 in-context examples on average, $20\times$ what **RT-J** sees at 8K cells)—far past **RT-J**'s 8K training context. The extra context pays off on classification: the TabICLv2 pipelines pass **RT-J**'s 8K-context AUROC by 3.6–4.2 points at the largest context and the LightGBM pipelines by up to about 1 point. On regression, only LLM Agent + TabICLv2 passes **RT-J**, and only from 65K cells on (32.5 vs. 33.4 nMAE at 131K); RDBLearn + TabICLv2 improves steadily with context but stays 0.3–0.5 nMAE behind at the two largest sizes, and the LightGBM pipelines stay 1.7–5.2 nMAE behind at every extended context size (16K–131K cells).

Table 11: nMAE (%, lower is better) of **RT-J** for (W, K) on the 8192-row test subsample at $L = 8192$, mean $\pm$ std. over 3 context seeds; the mean row averages the tasks per seed. Bold: within one standard deviation of the best per row.

Task	(10K, 20)	(10K, 10)	(1K, 20)	(1K, 10)
`rel-amazon/item-ltv`	8.49 ± 0.01	8.25 ± 0.09	8.35 ± 0.03	$\mathbf{8.07} \pm 0.03$
`rel-amazon/user-ltv`	$\mathbf{30.21} \pm 0.10$	30.44 ± 0.07	30.45 ± 0.07	$\mathbf{30.17} \pm 0.11$
`rel-avito/ad-ctr`	$\mathbf{42.17} \pm 0.55$	$\mathbf{42.22} \pm 0.45$	43.07 ± 0.64	43.11 ± 0.59
`rel-event/user-attendance`	$\mathbf{43.46} \pm 1.66$	$\mathbf{43.38} \pm 1.71$	$\mathbf{43.45} \pm 1.70$	$\mathbf{43.78} \pm 1.85$
`rel-f1/driver-position`	$\mathbf{38.53} \pm 0.03$	39.53 ± 0.10	39.97 ± 0.18	39.75 ± 0.07
`rel-hm/item-sales`	14.07 ± 0.20	$\mathbf{11.61} \pm 0.09$	12.56 ± 0.07	13.38 ± 0.06
`rel-stack/post-votes`	15.54 ± 0.04	15.46 ± 0.04	15.46 ± 0.02	$\mathbf{15.21} \pm 0.04$
`rel-trial/site-success`	$\mathbf{91.10} \pm 0.14$	91.94 ± 0.29	91.80 ± 0.31	92.18 ± 0.40
`rel-trial/study-adverse`	$\mathbf{16.39} \pm 0.03$	16.61 ± 0.04	16.76 ± 0.00	16.90 ± 0.05
Mean	$\mathbf{33.33} \pm 0.15$	$\mathbf{33.27} \pm 0.18$	33.54 ± 0.20	33.62 ± 0.20

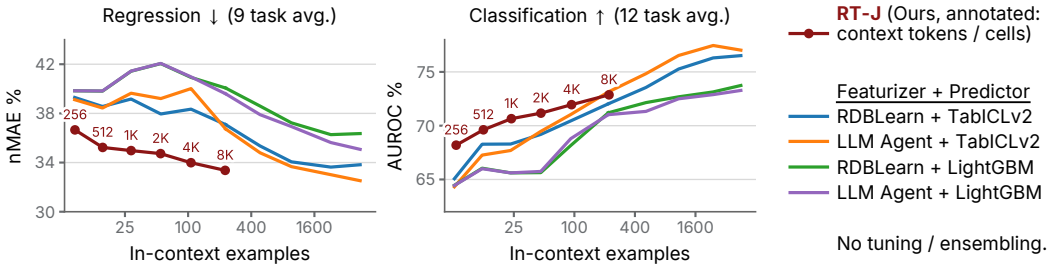


Figure 15: Few-shot relational prediction vs. in-context-learning baselines on a fixed 8192-row-per-task subsample of the RelBench test set, with the baselines extended to 131K-cell contexts. As in Figure 3, the x-axis is the mean number of in-context labeled task rows and the markers on the **RT-J** curve are annotated with the underlying context size (**RT-J** caps at its 8K-cell training context). With $20\times$ more labels than **RT-J** ever sees, the TabICLv2 pipelines lead on classification, and LLM Agent + TabICLv2 passes **RT-J**'s regression error only from 65K cells on.

J.2 Per-task baseline curves

Figures 16 and 17 disaggregate the aggregate baselines comparison of Figure 3 into one panel per RelBench task, plotting test nMAE for the 9 regression tasks and test AUROC for the 12 classification tasks against inference-time context size. The five methods, the context grid, and the full-test evaluation protocol match Section 5; per-task y-axes are auto-scaled.

On regression, **RT-J**'s aggregate dominance in Figure 3 is anchored by wide wins at every context size on tasks whose targets are heavy-tailed or spread across the graph (`rel-stack/post-votes`, `rel-amazon/user-ltv`, `rel-event/user-attendance`, `rel-avito/ad-ctr`), where the flat-feature pipelines stall or get worse with more context. Elsewhere the TabICLv2 pipelines keep improving with context and pass **RT-J** at the largest budgets (`rel-amazon/item-ltv`), lead it at every context size on `rel-hm/item-sales` and `rel-f1/driver-position`, as do the feature-engineering pipelines on `rel-trial/site-success`, while `rel-trial/study-adverse` ends 0.8 nMAE behind RDBLearn + TabICLv2. On classification, **RT-J** leads or ties on nearly every task at small contexts—the label-scarce regime this paper targets. At 8K the picture is mixed: **RT-J** stays ahead on `rel-event/user-repeat`, `rel-amazon/user-churn`, `rel-f1/driver-dnf` and `rel-avito/user-visits` and ties on `rel-hm/user-churn`, while the feature-engineering pipelines overtake it on `rel-avito/user-clicks`, `rel-amazon/item-churn`, the two `rel-stack` tasks, `rel-trial/study-outcome`, `rel-event/user-ignore` and `rel-f1/driver-top3`, mirroring the aggregate convergence at the largest label budget.

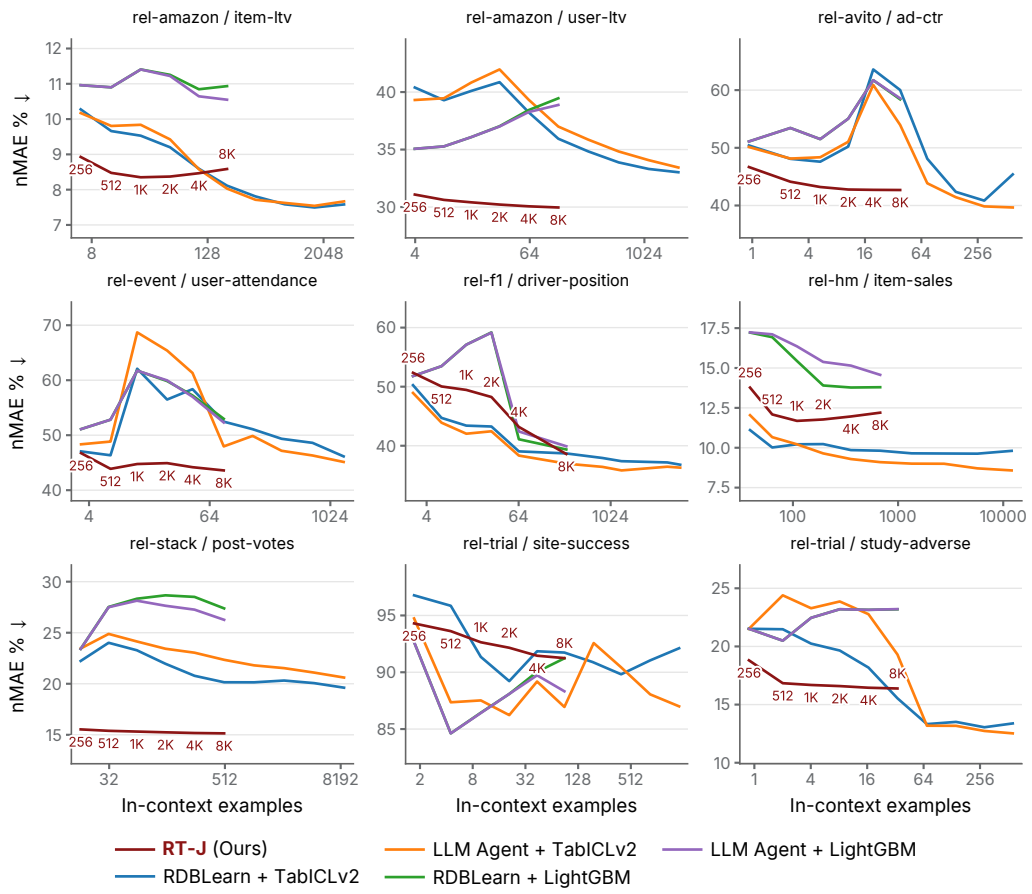


Figure 16: Per-task test nMAE versus the average number of in-context examples on RelBench regression tasks; per-task counterpart of Figure 3 (left).

J.3 Cost of test-time compute scaling

Table 12 compares the cost of context tuning and context ensembling (Section 6), counted in passes of the context sampler and model over an evaluation split, and in labels used beyond the training split from which in-context examples are drawn. See Table 9 for forward pass latencies.

Table 12: Passes over an evaluation split, in total and on the critical path when fully parallelized, and labels used beyond the training split.

Setting	Total passes	Fully parallel	Extra labels
Default context	1 test	1 test	none
Tuning (120 configurations, 4 seeds)	144 val. + 1 test	1 val. + 1 test	$\leq$ 4096 (val.)
Ensembling (16 seeds)	16 test	1 test	none

Tuning scores 120 configurations (L, ℓ, b, ρ) with 4 context seeds each. A single pass samples one context at the largest L and scores every $L \geq \ell$ on its prefixes, so the grid takes 36 passes per seed over the shared (ℓ, b, ρ) settings, i.e. 144 passes over a validation subsample of up to 4096 rows, plus one test pass with the selected configuration. Ensembling needs no labels beyond the training split. Both procedures parallelize fully across configurations and seeds.

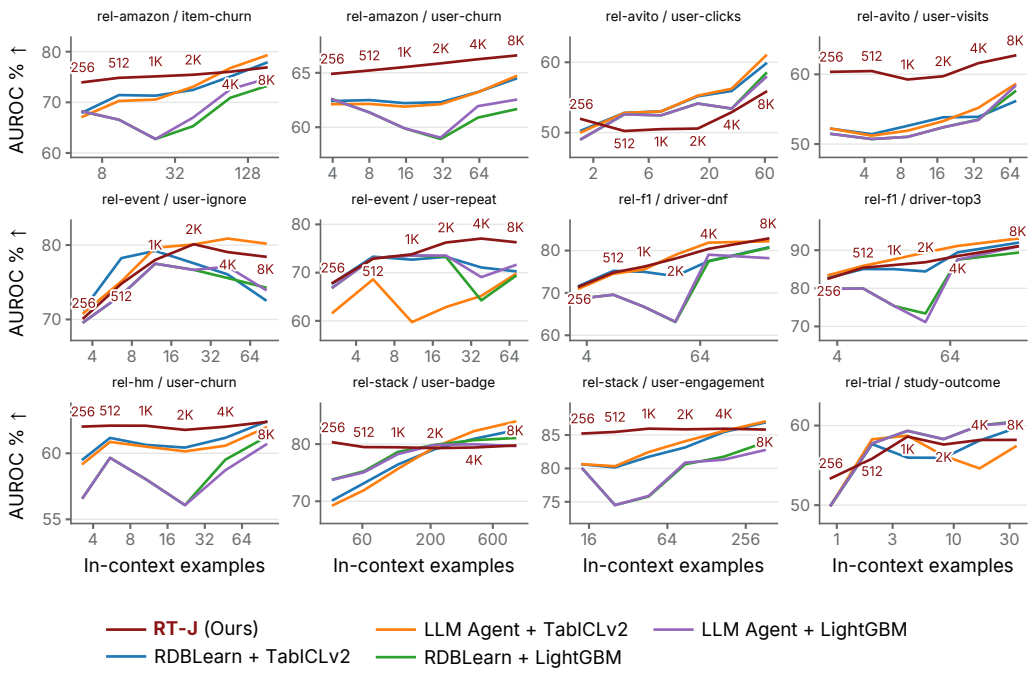


Figure 17: Per-task test AUROC versus the average number of in-context examples on RelBench classification tasks; per-task counterpart of Figure 3 (right).

J.4 Per-task ablation curves

This appendix disaggregates each ablation in Section 7 into per-task panels, one per RelBench task, evaluated on the 8192-row test subsample of Section 7 across context sizes; per-panel y-axes are auto-scaled so the visual ranking of methods within a task is preserved even where its absolute scale differs. These panels disaggregate the aggregate ablation figures (context retrieval, schema semantics) directly.

J.4.1 Context retrieval (per task)

Per-task counterpart of Figure 6. Random walk's advantage at small L is broad: at $L = 256$ it is the best or tied retriever on most regression tasks. Beyond $L = 1024$ RDBLearn-feature similarity leads on most of them (`item-ltv`, `user-ltv`, `ad-ctr`, `item-sales`, `post-votes`, `study-adverse`) and the two are close on `user-attendance` and `driver-position`; on `site-success` the structure-agnostic retrievers stay flat near 94 nMAE, 32-width BFS is far ahead beyond $L = 1024$, random walk drifts back up past $L = 1024$ and RDBLearn-feature similarity is erratic. On classification the same two retrievers lead nearly everywhere, RDBLearn-feature similarity by the wider margin on the churn and driver tasks and random walk on the `rel-event` tasks; the tasks with few labels (`user-clicks`, `user-engagement`, `study-outcome`) are noisy for every retriever. The 256-width BFS variant is a strong outlier at small L on several tasks; it is excluded from per-panel y-limit computation so the other curves remain readable, but its line is still drawn.

J.4.2 Schema semantics (per task)

Per-task counterpart of Figure 5. Schema semantics help on most tasks. The largest classification gaps (`rel-event/user-ignore`, `rel-event/user-repeat`) are worth up to 5.2 AUROC points; shuffled names win at every context size on `rel-avito/user-clicks`, `rel-f1/driver-top3` and `rel-hm/item-sales`, on `rel-amazon/user-ltv` from $L = 512$, and on `rel-f1/driver-position` by up to 5.1 nMAE at small contexts.

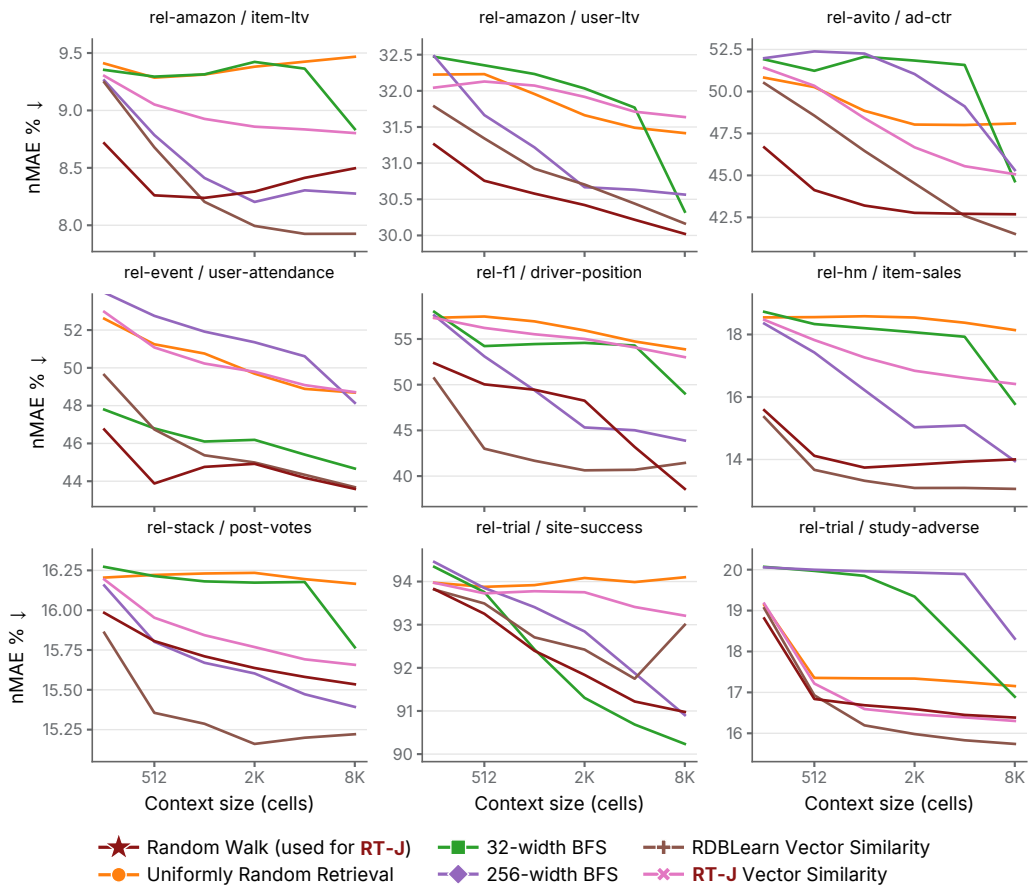


Figure 18: Per-task test nMAE for context-retrieval ablations on RelBench regression tasks; per-task counterpart of Figure 6 (left).

J.4.3 Context hyperparameter selection (per task)

Each task's *tuned* context is chosen by a grid search over the global context length $L \in \{512, \ldots, 8192\}$, the per-seed cell budget $\ell \in \{256, \ldots, 8192\}$ (with $\ell \leq L$), the BFS width $b \in \{8, 32, 128\}$, and the recency flag $\rho \in \{0, 1\}$—120 configurations per task—selecting per task the configuration that maximizes validation performance on a 4096-row validation subsample (four context seeds, per-row predictions averaged before scoring; nMAE for regression, AUROC for classification). Table 13 reports the resulting test metrics on the 8192-subsampled test set, comparing this tuned context against the shared *default* context ($L = 8\text{K}$, $\ell = 256$, $b = 32$, $\rho = 1$) at a single context seed; columns L, ℓ, b, ρ report the selected values.

J.4.4 Context ensembling (per task)

Per-task counterpart of Figure 4, for the default (blue) and per-task-tuned (orange) context. Increasing the number of ensembled context seeds n_{ens} improves classification on most tasks (9 of 12 at the default context and 10 of 12 at the tuned one), but degrades `rel-avito/user-clicks` by 3.0 points at the default context, while several regression tasks (`rel-amazon/user-ltv`, `rel-stack/post-votes`) are insensitive, in line with the flatter aggregate regression curve. The panels also localize the val–test mismatches behind the aggregate classification crossover in Figure 4: on `rel-avito/user-clicks` and `rel-stack/user-engagement` the tuned context is clearly worse than the default at every ensemble size, and on `rel-trial/study-outcome` it starts 2.3 AUROC ahead but ends 0.7 behind at $n_{\text{ens}} = 16$. (On `rel-amazon/user-churn`, tuning selects the default configuration, so the two curves coincide.)

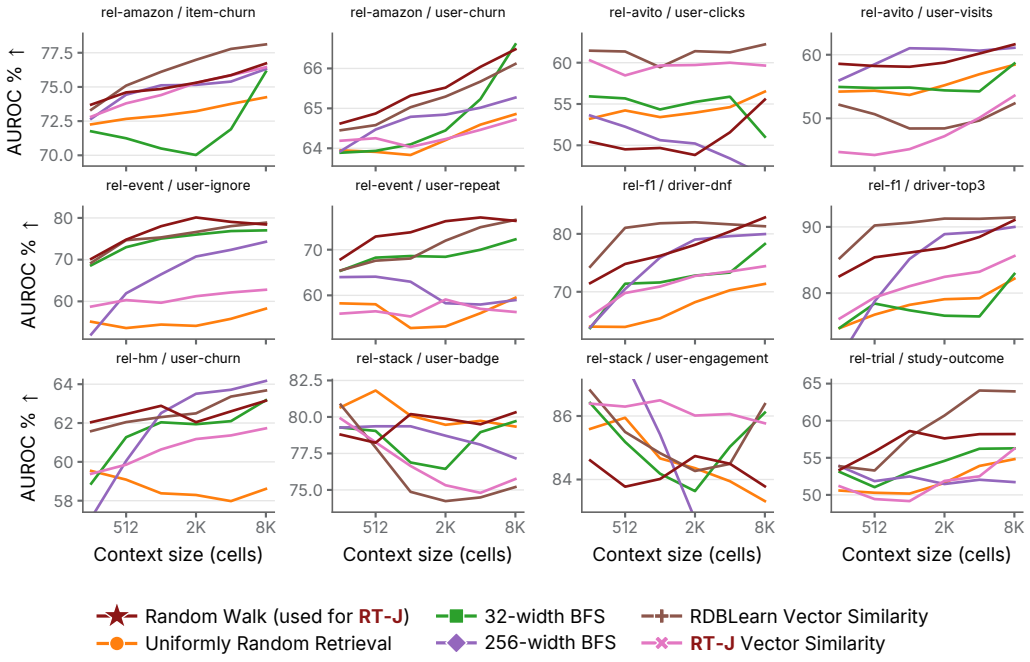


Figure 19: Per-task test AUROC for context-retrieval ablations on RelBench classification tasks; per-task counterpart of Figure 6 (right).

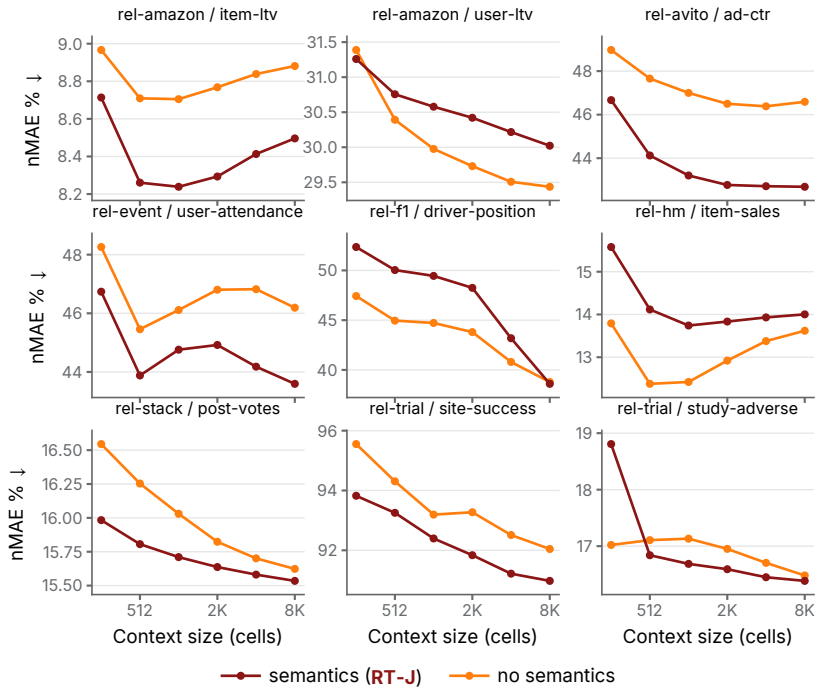


Figure 20: Per-task test nMAE for the schema-semantics ablation on RelBench regression tasks; per-task counterpart of Figure 5 (left).

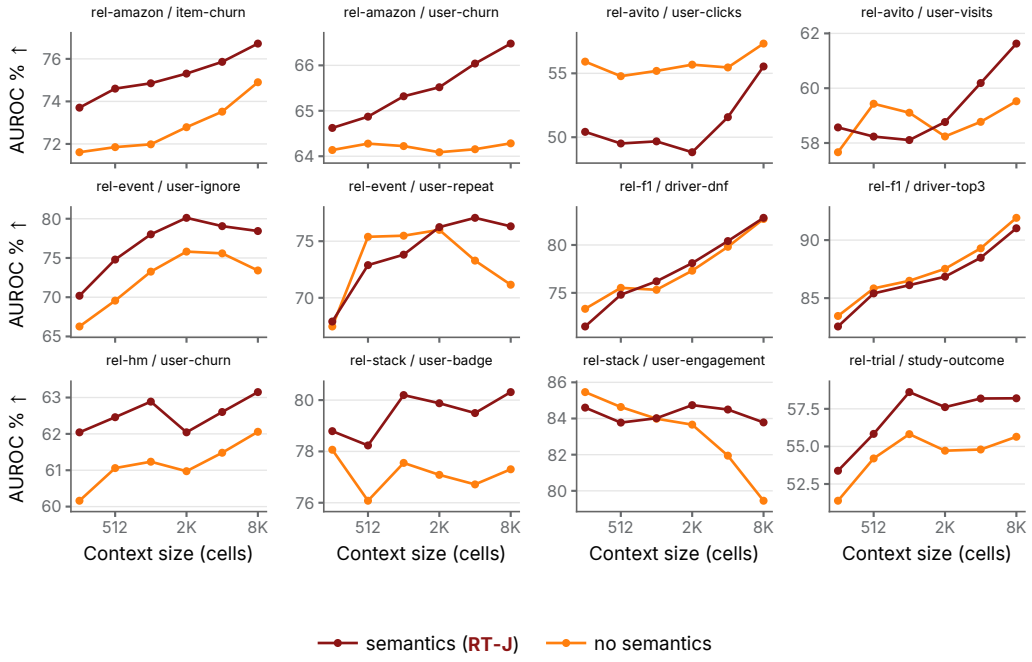


Figure 21: Per-task test AUROC for the schema-semantics ablation on RelBench classification tasks; per-task counterpart of Figure 5 (right).

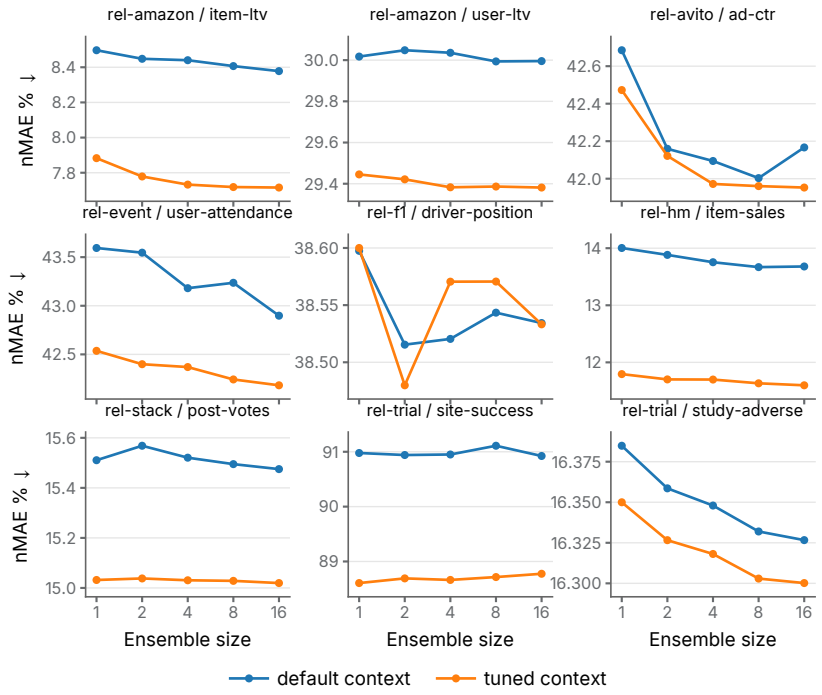


Figure 22: Per-task test nMAE for context ensembling, default vs. per-task-tuned context, on RelBench regression tasks; per-task counterpart of Figure 4 (left).

Table 13: Per-task test metrics on the 8192-subsampled test set: shared *default* context ($L = 8$K, $\ell = 256$, $b = 32$, $\rho = 1$) vs. the per-task *tuned* context. Columns L, ℓ, b, ρ report the configuration selected on the validation split. Single context seed (no ensembling). All metrics shown in percent. Best per row in bold; lower is better for nMAE, higher for AUROC.

DB	Task	L	ℓ	b	ρ	Default	Tuned
Regression (nMAE %, lower is better)							
rel-amazon	item-ltv	4K	2048	8	0	8.50	**7.88**
rel-amazon	user-ltv	4K	2048	8	0	30.02	**29.45**
rel-avito	ad-ctr	8K	1024	8	0	42.69	**42.47**
rel-event	user-attendance	8K	256	8	1	43.59	**42.54**
rel-f1	driver-position	8K	256	8	1	**38.60**	38.60
rel-hm	item-sales	512	256	128	0	14.00	**11.79**
rel-stack	post-votes	8K	2048	8	0	15.51	**15.03**
rel-trial	site-success	8K	4096	32	0	90.98	**88.61**
rel-trial	study-adverse	8K	256	128	1	16.38	**16.35**
Mean						33.36	**32.52**
Classification (AUROC %, higher is better)							
rel-amazon	item-churn	8K	512	128	1	76.73	**77.79**
rel-amazon	user-churn	8K	256	32	1	**66.48**	66.48
rel-avito	user-clicks	8K	2048	128	1	**55.54**	47.63
rel-avito	user-visits	1K	512	128	0	**61.63**	61.33
rel-event	user-ignore	4K	512	8	1	78.43	**80.53**
rel-event	user-repeat	2K	512	8	1	**76.31**	76.29
rel-f1	driver-dnf	8K	256	128	1	82.85	**83.03**
rel-f1	driver-top3	8K	256	128	1	**91.02**	90.75
rel-hm	user-churn	8K	512	128	0	63.14	**65.50**
rel-stack	user-badge	8K	4096	32	1	**80.20**	79.49
rel-stack	user-engagement	512	512	8	1	**83.76**	79.39
rel-trial	study-outcome	8K	2048	32	0	58.20	**60.52**
Mean						**72.86**	72.39

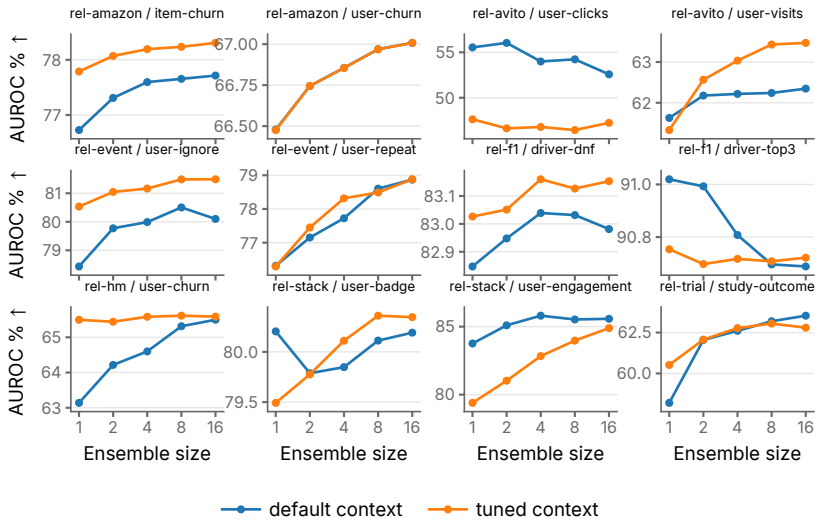


Figure 23: Per-task test AUROC for context ensembling, default vs. per-task-tuned context, on RelBench classification tasks; per-task counterpart of Figure 4 (right).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims made in the abstract and introduction are backed by experimental results (Sections 5 to 7).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in the Limitations paragraph of Section 8.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: We do not make any theoretical claims.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the experimental details for the models, data, and hyperparameters are provided (Sections 3 to 5 and Appendices A, D and G).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All the code and datasets are open-sourced at `https://star-project.stanford.edu/rt-j` (Section 8).

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All the necessary experimental details are provided (Section 5 and Appendices A and G).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported due to the compute cost of repeated pretraining runs. We report means over the 21 RelBench tasks with per-task results in Appendix J.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All the hardware details are provided along with the training setup (Appendix G).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the societal impacts of our work in Section 8.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We discuss our release measures of the collected data in Appendix D.6.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the papers and datasets have been cited, with the license of each source in Table 4 and our compliance practice in Appendix D.6.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All new assets are documented (Section 3 and Appendix D) and are open-sourced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We document the usage of LLMs for curating our data corpus in Section 3.3 and Appendices D.1 and D.4.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.