From Features to Structure: Task-Aware Graph Construction for Relational and Tabular Learning with GNNs

Tamara Cucumides
University of Antwerp
tamara.cucumidesfaundez@uantwerp.be

Floris Geerts
University of Antwerp
floris.geerts@uantwerp.be

ABSTRACT

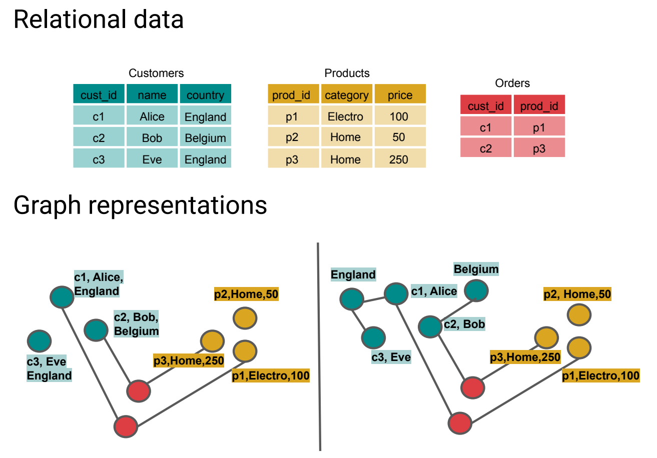
Tabular and relational data remain the most ubiquitous formats in real-world machine learning applications, spanning domains from finance to healthcare. Although both formats offer structured representations, they pose distinct challenges for modern deep learning methods, which typically assume flat, feature-aligned inputs. Graph Neural Networks (GNNs) have emerged as a promising solution by capturing structural dependencies within and between tables. However, existing GNN-based approaches often rely on rigid, schema-derived graphs—such as those based on primary-foreign key links—thereby underutilizing rich, predictive signals in non key attributes. In this work, we introduce auGraph, a unified framework for *task-aware* graph augmentation that applies to both tabular and relational data. auGraph enhances base graph structures by selectively promoting attributes into nodes, guided by scoring functions that quantify their relevance to the downstream prediction task. Empirically, auGraph outperforms schema-based and heuristic graph construction methods by producing graphs that better support learning for relational and tabular prediction tasks.

VLDB Workshop Reference Format:
Tamara Cucumides and Floris Geerts. From Features to Structure: Task-Aware Graph Construction for Relational and Tabular Learning with GNNs. VLDB 2025 Workshop: Tabular Data Analysis (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/TamaraCucumides/auGraph/.

1 INTRODUCTION

Relational databases are a foundational data storage paradigm, widely used to manage structured data across industry and science. Their normalized schema—multiple interlinked tables connected via primary-foreign key relationships—supports modularity and efficient querying. However, this structure poses challenges for machine learning methods that assume flat, feature-complete inputs. The prevailing workaround, by doing manual joins, feature engineering and aggregations, is not only laborious and memory intensive due to redundancy from denormalization, but also eliminates the relational structure that underpins the data, potentially leading to information loss and bias [11, 15].



Customers

cust_id	name	country
c1	Alice	England
c2	Bob	Belgium
c3	Eve	England

Products

prod_id	category	price
p1	Electro	100
p2	Home	50
p3	Home	250

Orders

cust_id	prod_id
c1	p1
c2	p3

Figure 1: Graph representations of relational data.

This has motivated research into automated, structure-aware methods for learning directly from relational without collapsing it into a single table [1, 3, 9, 14]. Even in the tabular setting, where data is already in a single table, graph-based approaches are used to model implicit structure [12]. These approaches construct graphs based on co-occurrence statistics [1], feature-wise multiplex connectivity [9], attribute-level hypergraphs [3], or contextual linkage for specific tasks like conversational Question Answering [14]. Yet, they generally lack task-aware mechanisms to guide graph construction.

Recent advances in *Relational Deep Learning* (RDL) provide an alternative to traditional feature engineering. RDL frameworks transform relational databases into heterogeneous graphs—*relational entity graphs*—where rows become nodes and key-based links define edges [6]. GNNs are then applied to learn task-specific representations over these structures in an end-to-end fashion [2, 6, 16]. This approach has achieved strong performance across multi-table prediction tasks, and has recently been extended with relational graph transformers [4] and foundation models for in-context learning over arbitrary databases [7].

At the core of RDL and related GNN-based tabular learning lies a persistent challenge: *how to construct a graph that captures meaningful statistical dependencies?* Schema-derived graphs rely on key-based connectivity, often resulting in sparse or semantically limited structures. In single-table settings, various graph constructions exist, but they are typically task-agnostic and not optimized for downstream performance.

To illustrate the challenge, consider the Customer–Order–Product relational snippet in Figure 1 (top). Beyond primary-key constraints, each Order row carries foreign-key references into the Customer and Product tables. If we simply translate this to the standard relational entity graph (Figure 1, bottom left), each tuple becomes a vertex and every foreign-key relationship becomes an edge.

Although this schema-derived graph captures referential structure, it can miss other useful connections. For example, suppose we extract the country attribute from Customer and create two new vertices—"England" and "Belgium"—then link each customer to the vertex corresponding to their country (Figure 1, bottom right). This augmentation reveals that customers from the same country share a common node, introducing edges that reflect shared attribute values rather than just keys.

In many graph-learning tasks—whether using graph neural networks or graph transformers—these additional edges can be critical for downstream predictions. We call the systematic enrichment of a relational entity graph with nodes and edges derived from attribute values graph augmentation, and investigating this process is the focus of our paper.

Our contribution. We present auGraph, a method that systematically integrates attribute-level signals into the graph structure through a task-guided augmentation process. While auGraph is broadly applicable, in this work we focus on node-level prediction tasks, where the goal is to improve learning for a given target relation. Unlike approaches that rely solely on fixed schema links or indiscriminate feature inclusion, auGraph evaluates and incorporates attributes based on their utility for the prediction task. This results in graph structures that tailored to the learning objective, applicable across both relational databases and single tables.

2 PRELIMINARIES

We recall the relational data model and then define the corresponding relational entity graphs. We conclude by describing the general graph learning approach to relational (deep) learning.

2.1 Databases

We recall some textbook definitions. Let Dom be an arbitrary *domain* of data values, and let $\mathcal{R} = \{R_1, \ldots, R_n\}$ be a *schema*, i.e., a finite set of *relations*. Each relation $R \in \mathcal{R}$ is associated with a finite set of *attributes* $\mathsf{att}(R) \subseteq \mathsf{Att}$, where Att is a fixed collection of *attribute names*. A *table* T for relation R consists of a finite set of *tuples* (or *rows*) $r = (r[A_1], \ldots, r[A_d])$ assigning values from Dom to all attributes $A_1, \ldots, A_d \in \mathsf{att}(R)$.

We enrich the schema with key and foreign-key constraints. First, we assume that each relation R has a designated *primary key* $K_R \subseteq \mathsf{att}(R)$. The corresponding *primary-key constraint* on the table T of R requires that for all $r, s \in T$, if $r[K_R] = s[K_R]$, then $r = s$. Second, let $\mathcal{L} \subseteq \{(R_i, A_j, R_k) \mid R_i, R_k \in \mathcal{R},\ A_j \in \mathsf{att}(R_i)\}$ be a set of *foreign-key links*. Each $(R_i, A_j, R_k) \in \mathcal{L}$ imposes a constraint on the tables: for every tuple r in the table T_i of R_i, there must exist a tuple s in the table T_k of R_k such that $r[A_j] = s[K_{R_k}]$. In other words, the A_j-attribute of R_i references the primary key of R_k. For simplicity, we assume single attributes foreign-key links only.

We refer to the pair $(\mathcal{R}, \mathcal{L})$ as a *relational schema*, and to a collection of tables $D = (T_1, \ldots, T_n)$ that satisfies all key and foreign-key constraints as a *database* for that schema.

Example 2.1. Consider a schema with the following relations:

- R_1 = Customers($\underline{\textit{cust\_id}}$, *name*, *country*)
- R_2 = Orders($\underline{\textit{cust\_id}, \textit{prod\_id}}$)
- R_3 = Products($\underline{\textit{prod\_id}}$, *category*, *price*),

in which their attributes are listed. The primary key attributes are underlined. Furthermore, *cust_id* in Orders is a foreign key referencing *cust_id* in Customers, and *prod_id* in Orders is a foreign key referencing *prod_id* in Products. Thus, the set of relations is $\mathcal{R} = \{R_1, R_2, R_3\}$ and the foreign-key set is:

$$\mathcal{L} = \{(R_2, \textit{cust\_id}, R_1),\ (R_2, \textit{prod\_id}, R_3)\}.$$

This models a typical customer–order–product schema, where each order links a customer to a product. We have shown an example database of this schema in Figure 1 (top). ◇

▷ *Remark.* We observe that *tabular data*, as typically understood in tabular learning, can be regarded as a special instance of a database, consisting of a single table T of schema $\mathcal{R} = \{R\}$ and with no foreign-key links ($\mathcal{L} = \emptyset$).

2.2 Relational Entity Graphs

We next associate a graph to a relational schema and database. Given a schema $(\mathcal{R}, \mathcal{L})$ and a database D over that schema, we define its *relational entity graph* [6] as $G_{\text{REG}} = (V, E, \phi, \tau)$, where

- the vertex set V is defined as $V := \{v_r \mid r \in T_i,\ R_i \in \mathcal{R}\}$, i.e., V contains one vertex v_r for each tuple r in each table in D;
- the edge set E is given by $E := \{(v_r, v_s) \mid (R_i, A_j, R_k) \in \mathcal{L},\ r \in T_i,\ s \in T_k,\ r[A_j] = s[K_{R_k}]\}$, i.e., E contains an edge (v_r, v_s) whenever the corresponding tuples satisfy a foreign-key link $(R_i, A_j, R_k) \in \mathcal{L}$; and
- ϕ and τ label each vertex v_r by its tuple $r \in T_i$ and relation name R_i of T_i, respectively, i.e., $\phi(v_r) := r$ and $\tau(v_r) := R_i$.

In other words, for each primary key value we have one vertex, which ϕ labels with the corresponding record. In addition, foreign-key links on the schema, are instantiated as edges between the vertices (records) in the graph.

Example 2.2. Continuing with our example database, we show the corresponding relational entity graph in Figure 1 (bottom, left). Each row in the tables becomes a vertex in the graph. Based on the foreign-key set, edges are added from each order to its associated customer and product. In particular, we have edges between the red vertex (corresponding to (c_1, p_1) and the vertices corresponding to rows c_1 and p_1 in Customers and Products, respectively. Similarly, we have edges between the other red vertex (corresponding to (c_2, p_3) to the vertices corresponding to rows c_2 and $p32$, respectively. We also remark that this yields a heterogeneous graph with typed nodes (depending on the relation) and different kinds of foreign-key-based edges (depending on relations and attributes involved). ◇

▷ *Remark.* It should be clear that we can move freely between schemas and databases, and relational entity graphs. That is, they carry the same information. Of course, the connection between tuples is explicit in the graph representation, whereas it is implicit in the database representation.

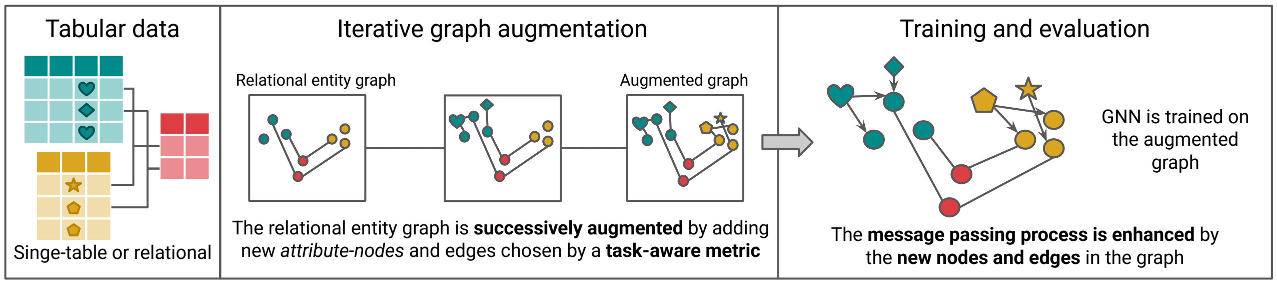

Tabular data
Singe-table or relational
Iterative graph augmentation
Relational entity graph
Augmented graph
The relational entity graph is **successively augmented** by adding new *attribute-nodes* and edges chosen by a **task-aware metric**
Training and evaluation
GNN is trained on the augmented graph
The **message passing process is enhanced** by the **new nodes and edges** in the graph

Figure 2: Overview of AUGRAPH. Given a tabular dataset and a task, AUGRAPH constructs a task-aware graph through iterative attribute promotion and scoring. The resulting graph is then used by a downstream GNN to perform the learning task.

2.3 Relational Deep Learning

We focus on *tuple-level* (or node-level) *prediction tasks*. Given a relation $R_i \in \mathcal{R}$ with corresponding table T_i in a database D, the goal is to learn a function

$$f : T_i \to \mathcal{Y},$$

where $\mathcal{Y}$ denotes the space of prediction targets. In typical applications, $\mathcal{Y}$ may be a finite set of discrete class labels—such as `will buy` or `will not buy` in transaction data, or disease categories in medical diagnostics—or a subset of the real numbers in regression tasks, such as predicting a customer's credit risk or forecasting future revenue.

We adopt the graph-based approach to relational learning [6], and thus aim to model f as a function defined on nodes of a graph. To enable this, we use – as expected – *relational entity graphs*. More specifically, we extend the relational schema by introducing a new relation $R_{\text{train}}(K, Y)$, whose table T_{train} contains pairs $(r[K], f(r))$, where $r[K]$ is the key of a tuple $r \in T_i$ and $f(r) \in \mathcal{Y}$ is its label. To link training labels to data, we also introduce a foreign-key link $(R_{\text{train}}, K, R_i) \in \mathcal{L}$, ensuring that the key attribute K in R_{train} references the primary key of R_i.

As a result, the relational entity graph $G_{\text{REG}} = (V, E, \phi, \tau)$ includes a set of *training vertices*

$$V_{\text{train}} := \{v_r \mid r \in T_i \text{ and } (r[K], f(r)) \in T_{\text{train}}\},$$

where each training vertex v_r is labeled by $\phi(v_r) = (r[K], y)$, representing both the tuple's key and its label. For notational convenience, we will write $f(v)$ to refer to the true label y of a training vertex v, rather than accessing it via $\phi(v)[Y]$.

We now define or learn f using a graph neural network (GNN), abstracted as a function $f_\theta(G, v) \in \mathcal{Y}$, which takes the relational entity graph G and a vertex $v \in V$ (typically from the same relation R_i) as input, and returns a prediction in $\mathcal{Y}$. The function is parameterized by $\theta \in \Theta$, representing learnable parameters such as weights and biases.

Training the model then amounts to minimizing an empirical loss over the labeled training vertices:

$$\min_{\theta \in \Theta} \sum_{v \in V_{\text{train}}} \ell\big(f_\theta(G, v), f(v)\big),$$

for a suitable loss function ℓ, such as cross-entropy (for classification) or squared error (for regression).

3 FEATURE INFORMED GRAPH CONSTRUCTION

We introduce AUGRAPH, a principled and scalable framework for *task-aware graph construction* from relational and tabular data. Our key insight is to treat graph construction itself as a pre-processing step tuned for learning—augmenting the base entity graph by promoting high-value attributes into new structural components. These attribute-derived nodes and edges are selected via scoring functions that capture statistical, structural, and model-based signals of predictive relevance. By iteratively augmenting the graph based on these signals, AUGRAPH tailors the graph topology to better support the target task, all while avoiding brute-force enumeration of attribute subsets. As shown in Figure 2, this graph augmentation process enhances message passing and improves GNN performance, producing graphs that are compact, interpretable, and aligned with the learning objective. We now describe the main components of AUGRAPH: how attribute promotion augments the graph structure, the scoring functions used to evaluate candidate attributes, and how these are combined in the iterative construction procedure.

3.1 Augmenting Graphs with Attribute Nodes

In terms of databases, relational entity graph augmentation corresponds to extracting a new unary table of constants from a chosen attribute and linking it back to the original graph. More precisely, let $(\mathcal{R}, \mathcal{L})$ be a relational schema with database D. Fix $R_i \in \mathcal{R}$ and an attribute $A \in \text{att}(R_i)$. We introduce a new unary relation R_A with table $T_A := \pi_A(T_i) = \{\, r[A] \mid r \in T_i\}$, and add the foreign-key link (R_A, A, R_i) to $\mathcal{L}$. In graph terms, starting from the relational entity graph $G_{\text{REG}} = (V, E, \phi, \tau)$ of $(\mathcal{R}, \mathcal{L})$ and D, the augmented graph $G_{\text{REG}}^{(A)}$ is obtained by

- adding vertices $V_A = \{v_a : a \in T_A\}$, i.e., one vertex per each attribute value;
- for each original vertex $v \in V$ with $\phi(v) = r$, $\tau(v) = R_i$ and $r[A] = a$, adding an edge (v_a, v); and finally,
- setting $\phi(v_a) := a$ and $\tau(v_a) := R_A$.

If multiple attributes $A_1, A_2, \ldots$ are promoted, the resulting graph is denoted $G_{\text{REG}}^{(A_1, A_2, \ldots)}$ and is obtained by sequential augmentation.

3.2 Attribute Scoring Metrics

To guide graph augmentation, AUGRAPH uses task-aware scoring functions that rank attributes by their potential to improve downstream prediction. Each score captures a different signal: from raw statistical relevance to structural and model-informed criteria. We recall that V_{train} is the set of vertices in our graph, corresponding to the tuples in the relation for which we have the actual labels and on which we want to do learning.

Given a non-key attribute A, we evaluate its relevance for the learning task and its potential to be used for graph augmentation, via the following metrics:

Statistical Signal. We begin with a classical feature selection metric only looking at the data represented by the underlying graph:
▷ *Mutual Information.* We define the *fully joined training table* of D as $D_{\text{train}} = T_1 \bowtie \cdots \bowtie T_n \bowtie T_{\text{train}}$, where the joins are computed based on the foreign-key links[1], and compute the mutual information between the attribute A and the label $\mathcal{Y}$:

$$s_{\text{MI}}(A; G) = \sum_{(a,y) \in \pi_{A,Y}(D_{\text{train}})} \hat{P}(a, y) \log\left(\frac{\hat{P}(a, y)}{\hat{P}(a)\hat{P}(y)}\right),$$

where $\hat{P}(a, y)$ denotes the empirical joint probability of observing value a for attribute A and label y in D_{train}, and the empirical marginals are given by $\hat{P}(a) = \sum_y \hat{P}(a, y)$ and $\hat{P}(y) = \sum_a \hat{P}(a, y)$. This score quantifies the dependence between the attribute and the label, and is computed directly from the tabular data.

Graph-aware Signal. We next assess whether A improves the topology of the graph G for learning.
▷ *Entropy Gain.* The entropy gain metric measures how graph augmentation with attribute A creates more homogeneous neighborhoods with respect to the target labels in $\mathcal{Y}$

For a node v, let $\mathsf{N}^d_G(v)$ denote its d-hop neighborhood in graph G, and let $\mathsf{N}^d_{G,\text{train}}(v) = \mathsf{N}^d_G(v) \cap V_{\text{train}}$ be the subset of training nodes within it. We define the empirical label distribution over a set $S \subseteq V_{\text{train}}$ as $\hat{P}_S(y) = \frac{1}{|S|} \sum_{u \in S} \mathbb{I}[f(u) = y]$, and its entropy:

$$H(S) = -\sum_y \hat{P}_S(y) \log \hat{P}_S(y).$$

Then, the entropy gain score can be written as:

$$s_{\text{ent}}(A; G) = \frac{1}{|V_{\text{train}}|} \sum_{v \in V_{\text{train}}} \left[H\big(\mathsf{N}^d_{G,\text{train}}(v)\big) - H\big(\mathsf{N}^d_{G^{(A)},\text{train}}(v)\big) \right].$$

▷ *Path Disagreement.* Let $\mathsf{Path}_A(u, v) = 1$ if u and v in V_{train} are connected through a shared attribute-vertex in V_A, and 0 otherwise. Then we define the path disagreement metric as:

$$s_{\text{dis}}(A; G) = \frac{1}{Z} \sum_{u,v \in V_{\text{train}}} \mathsf{Path}_A(u, v) \cdot \mathbb{I}[f(u) \neq f(v)],$$

where Z is a normalization constant. Lower values suggest that A tends to connect nodes with the same label.

Model-based Signal. We also include a GNN-based score.

▷ *GNN Gain.* A model f_θ, which maps nodes in a graph to predictions, is trained once on G and then kept fixed. We assess how augmenting the graph with attribute A affects its performance over the validation nodes:

$$s_{\text{GNN}}(A; G) = \mathsf{Eval}_{f_\theta}(G^{(A)}) - \mathsf{Eval}_{f_\theta}(G),$$

[1]We use natural (inner) joins to ensure that mutual information is computed only on well-defined, observed attribute-label pairs. Outer joins or full disjunctions would introduce null values and undefined combinations, which are incompatible with standard information-theoretic measures.

where $\mathsf{Eval}_{f_\theta}(G)$ denotes the aggregated validation performance of f_θ on graph G, using a task-specific metric (e.g., accuracy or F1 score). This provides a lightweight proxy for the utility of attribute A: if its inclusion improves message passing, performance should increase, even without retraining.

While some scores are purely statistical, and others are structure or model-aware, all are designed to promote attributes that support downstream learning.

3.3 Graph Augmentation Procedure

Rather than selecting a fixed top-k set of attributes in one pass, AUGRAPH builds the graph incrementally through a selection loop. As shown in Algorithm 1, starting from the base graph G_{REG} (line 1), it repeatedly (loop, line 3–11) selects the highest scoring attribute (according to a chosen scoring function, line 5) and promotes it into the graph. After each promotion, scores are recomputed to reflect the updated structure (line 4), favoring attributes that are complementary rather than redundant. This process continues until a fixed budget k is reached or the best available score falls below an early-stopping threshold τ (line 6). The resulting graph G_{aug} can then be used as input to any downstream GNN model.

Algorithm 1 Iterative Attribute-Based Graph Augmentation with Early Stopping

```
Require: Relational schema and labels $f : V \to \mathcal{Y}$, scoring function $s(A; G)$, maximum attributes $k$, threshold $\tau$
Ensure: Augmented graph $G_{\text{aug}}$
 1: Initialize $G_{\text{aug}} \leftarrow G_{\text{REG}}$
 2: Let $C$ be the set of non-key categorical attributes
 3: for $i = 1$ to $k$ do
 4:     Compute $s(A; G_{\text{aug}})$ for each $A \in C$
 5:     Let $A^* = \arg\max_{A \in C} s(A; G_{\text{aug}})$
 6:     if $s(A^*; G_{\text{aug}}) < \tau$ then
 7:         break {Early stopping if no informative attribute remains}
 8:     end if
 9:     $G_{\text{aug}} \leftarrow G^{(A^*)}_{\text{aug}}$
10:     Remove $A^*$ from $C$
11: end for
12: return $G_{\text{aug}}$
```

4 EXPERIMENTS

We test AUGRAPH in three settings, focusing on:

(Q1) **Attribute discovery**: Can AUGRAPH identify task-relevant attributes, and does promoting them improve performance?
(Q2) **Relational augmentation**: Can AUGRAPH enhance schema-derived graphs in multi-table relational databases?
(Q3) **Tabular graph quality**: Does AUGRAPH outperform heuristic graph construction baselines on single-table data?

4.1 Setup

We evaluate node classification performance across three regimes: (i) synthetic relational data, (ii) single-table tabular benchmarks, and (iii) multi-table relational databases. In each case, data is transformed into a graph via a designated construction strategy, and a fixed GNN model is trained for supervised learning. Only the graph structure varies between runs.

The GNN used is a two-layer heterogeneous architecture built using `HeteroConv` and `SAGEConv` from PyTorch Geometric [8], with mean aggregation [10]. This architecture is kept constant across all datasets and experiments to isolate the effect of the graph construction itself. We intentionally avoid tuning or comparing GNN architectures, as our goal is not to find the optimal model for a specific dataset but rather to assess how task-aware graph augmentation influences learning outcomes within a fixed GNN setting.

We compare auGraph by training and evaluating a GNN over the following graph constructions[2]:

- **REG**[3]: Relational Entity Graph, as defined in Section 2.2 [6].
- **All-promote**: Graphs in which all attributes are promoted [19].
- **Random-k**: Graphs built by promoting k random attributes.
- **kNN Graph**: Instance similarity graphs constructed via k-nearest neighbors in feature space [5].

We evaluate models using accuracy, F1-score, and ROC-AUC. Accuracy and F1 reflect classification quality; ROC-AUC captures ranking performance and robustness to class imbalance.

To isolate the contribution of each signal, we run the graph augmentation pipeline separately for each scoring function and report results using the best-performing one per setting. We do not use a combined selector in our experiments, though auGraph supports combining scores (e.g., via weighted aggregation), which we leave for future exploration.

Datasets. We use three different data regimes:

- **Synthetic:** A relational schema modeled after `relbench-h&m` [6], with a binary label defined by a known subset of attributes. This tests auGraph ability to recover relevant structure (Q1) and improve over relational entity graphs (Q2).
- **Relational:** The `hepatitis` dataset from the CTU Relational Datasets [13], where the task is to classify entities in a multi-table schema. We compare auGraph against relational entity graph, all-promote, and random baselines (Q2).
- **Tabular:** The UCI `mushroom` dataset [17], treated as a flat table without schema. This setting evaluates whether auGraph can construct effective graph structure from purely tabular data (Q3).

4.2 Results and Discussion

To evaluate (Q1), we focus on synthetic data, where the ground-truth label depends on three specific attributes. We perform graph augmentation using all scoring metrics and examine which attributes each method promotes. With a promotion budget of $k = 3$, the `path-disagreement` metric identifies two of the three ground-truth attributes but ranks third in downstream performance, suggesting that recovering relevant features does not necessarily yield the most effective graph structure for learning. The `gnn-gain` metric, while promoting only one relevant attribute, achieves the highest performance by aligning directly with message passing (see Figure 3). The `entropy-gain` score plateaus after two promotions and stops, never degrading performance—suggesting greater robustness. Notably, all metrics outperform the minimal relational entity graph, showing their general utility for guiding augmentation.

Table 1: Performance in test data. Results are reported for $k = 3$ and the Random-k is averaged over 3 runs. auGraph results are from the metric that performs best on the validation set.

Model	F1-score	Accuracy	ROC-AUC
Synthetic Data (Relational)			
REG	0.734	0.784	0.834
All-promote	0.668	0.728	0.726
Random-k	0.699	0.765	0.786
auGraph (top-k, s_{GNN})	**0.773**	**0.816**	**0.836**
Relational Data			
REG	0.930	0.936	0.987
All-promote	0.894	0.904	0.985
Random-k	0.917	0.924	**0.988**
auGraph (top-k, s_{GNN})	**0.939**	**0.944**	0.987
Tabular Data			
All-promote	0.969	0.970	**0.992**
Random-k	0.944	0.946	0.987
kNN Graph (k=10)	0.937	0.936	0.979
auGraph (top-k, s_{MI})	**0.985**	**0.985**	**0.992**

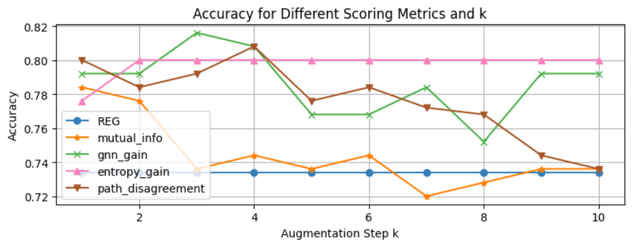


Figure 3: Accuracy on augmented graph with different scoring methods versus promotion budget k. Synthetic data. Best viewed in color.

To address (Q2) and (Q3), we refer to Table 1, which compares performance across relational and tabular benchmarks.

In the relational setting, across both synthetic and real datasets, auGraph consistently outperforms the baseline relational entity graph. Naively promoting all attributes degrades performance, often underperforming even the minimal graph, while random promotion yields modest gains. These results show that structural complexity alone is insufficient; task-aware augmentation is crucial. auGraph's consistent improvements demonstrate that its scoring functions successfully identify attributes that induce useful inductive bias.

In the tabular setting, where the base graph is often empty or poorly structured, promoting random attributes proves ineffective, likely due to arbitrary edge formation. While promoting all attributes helps, it also brings redundancy and noise. auGraph again

[2]Our goal is to isolate the impact of graph construction strategies on graph-based learning. For this reason, we do not compare against traditional non-graph baselines such as XGBoost or CatBoost, which do not make use of the graph structure.

[3]The **REG** baseline is not included for tabular data, as it would result in a set of disconnected nodes.

achieves the best performance, showing that feature-informed selective augmentation enables compact yet expressive graphs that improve downstream learning.

We also note that the computational cost of auGraph is dominated by score computation during each promotion step, rather than by graph construction itself. When an attribute A is promoted, one node per unique value in A is added to the graph, along with edges connecting these to the existing nodes that contain those values. However, the scoring metrics are designed to favor attributes that improve task-relevant connectivity, typically those with a moderate number of shared values between instances. As a result, high-cardinality attributes (which would introduce a large overhead without aiding message passing) tend to receive low scores and are rarely promoted. This acts as an implicit regularizer, keeping both runtime and memory usage practical throughout the iterative process. In practice, we observe that even a small number of promoted attributes often yields meaningful performance improvements (cf. Figure 3), making auGraph applicable even in constrained settings.

Finally, we emphasize the importance of the promotion budget k and the early stopping threshold τ. As seen in Table 1, promoting low-quality attributes (as in the All-promote baseline) can obscure the signal, but auGraph's controlled, iterative augmentation mitigates this risk. Moreover, as shown in Figure 3, performance tends to plateau after a few high-quality promotions, with further additions offering diminishing or negative returns. Rather than running until convergence, risking overfitting or unnecessary complexity, we select k and τ through validation performance, ensuring efficient and robust graph construction.

5 CONCLUSIONS AND FUTURE DIRECTIONS

We have presented auGraph, a principled, task-aware alternative to schema-based or heuristic graph construction for tabular and relational data. By promoting informative attributes into the graph topology, auGraph consistently improves downstream performance. To our knowledge, this is the first framework to pose graph construction as a task-aware feature selection problem.

Looking ahead, an important direction is to extend auGraph to *temporal relational data*. While current scores apply to temporally-aware graphs, they fail to capture unique dynamics—such as evolving attributes, cross-time entity links, or timestamped dependencies. Designing temporal-sensitive scores could yield stronger structural priors and more expressive graphs.

A second and more foundational direction is to better understand the *expressivity and generalization* of task-aware graph construction. A compelling goal is to align augmentation more directly with labels—e.g., by maximizing agreement between 1-WL colourings and targets. This could define a notion of *task-optimal graph construction*, offering both a theoretical target and a practical guide. Understanding how structural properties shape generalization remains an open challenge, especially given the weak empirical link between expressivity and performance in GNNs [18]. We view this as a promising and underexplored research avenue.

Acknowledgment. Tamara Cucumides is supported by FWO project G019222N.

REFERENCES

[1] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 1335–1349. https://doi.org/10.1145/3318464.3389742

[2] Milan Cvitkovic. 2020. Supervised Learning on Relational Databases with Graph Neural Networks. *arXiv preprint arXiv:2002.02046* (2020). https://doi.org/10.48550/arXiv.2002.02046 arXiv:2002.02046 Presented at the ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds.

[3] Kounianhua Du, Weinan Zhang, Ruiwen Zhou, Yangkun Wang, Xilong Zhao, Jiarui Jin, Quan Gan, Zheng Zhang, and David Wipf. 2022. Learning Enhanced Representation for Tabular Data via Neighborhood Propagation. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 16373–16384. https://proceedings.neurips.cc/paper_files/paper/2022/file/67e79c8e9b11f068a7cafd79505175c0-Paper-Conference.pdf

[4] Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. 2025. Relational Graph Transformer. *arXiv* (2025). https://arxiv.org/abs/2505.10960

[5] Federico Errica. 2023. On Class Distributions Induced by Nearest Neighbor Graphs for Node Classification of Tabular Data. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 28910–28940. https://proceedings.neurips.cc/paper_files/paper/2023/file/5c1863f711c721648387ac2ef745facb-Paper-Conference.pdf

[6] Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. 2024. Position: Relational Deep Learning – Graph Representation Learning on Relational Databases. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Vol. 235. PMLR, 13592–13607. https://proceedings.mlr.press/v235/fey24a.html

[7] Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen, and Jure Leskovec. 2025. KumoRFM: A Foundation Model for In-Context Learning on Relational Data. Kumo.ai whitepaper. https://kumo.ai/research/kumo_relational_foundation_model.pdf Accessed May, 2025.

[8] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. *arXiv* (2019). https://arxiv.org/abs/1903.02428

[9] Xiawei Guo, Yuhan Quan, Huan Zhao, Quanming Yao, Yong Li, and Weiwei Tu. 2021. TabGNN: Multiplex Graph Neural Network for Tabular Data Prediction. *arXiv* (2021). https://arxiv.org/abs/2108.09127 Presented at the KDD 2021 Workshop on Deep Learning Practice for High-Dimensional Sparse Data.

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 1024–1034. https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[12] Cheng-Te Li, Yu-Che Tsai, Chih-Yao Chen, and Jay Chiehen Liao. 2024. Graph Neural Networks for Tabular Data Learning: A Survey with Taxonomy and Directions. *arXiv* (2024). https://arxiv.org/abs/2401.02143

[13] Jan Motl and Oliver Schulte. 2015. The CTU Prague Relational Learning Repository. *arXiv preprint arXiv:1511.03086* (2015). https://doi.org/10.48550/arXiv.1511.03086 arXiv:1511.03086

[14] Thomas Müller, Francesco Piccinno, Peter Shaw, Massimo Nicosia, and Yasemin Altun. 2019. Answering Conversational Questions on Structured Data without Logical Forms. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 5901–5909. https://doi.org/10.18653/v1/D19-1603

[15] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table union search on open data. *Proc. VLDB Endow.* 11, 7 (March 2018), 813–825. https://doi.org/10.14778/3192965.3192973

[16] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *Proceedings of the 15th International Conference on The Semantic Web (ESWC) (Lecture Notes in Computer Science)*, Vol. 10843. Springer, 593–607. https://doi.org/10.1007/978-3-319-93417-4_38

[17] UCI Machine Learning Repository. 2024. Mushroom Dataset. https://doi.org/10.24432/C5959T Accessed May 2025.

[18] Antonis Vasileiou, Stefanie Jegelka, Ron Levie, and Christopher Morris. 2025. Survey on Generalization Theory for Graph Neural Networks. *arXiv* (2025). https://arxiv.org/abs/2503.15650

[19] Han Zhang, Quan Gan, David Wipf, and Weinan Zhang. 2023. GFS: Graph-based Feature Synthesis for Prediction over Relational Databases. *arXiv* (2023). https://arxiv.org/abs/2312.02037 Presented at the VLDB 2024 Workshop on Tabular Data Analysis (TaDA).