Exploring the Impacts of Architecture and Scale on GNN Performance on Relational Data

By: Joseph Guman, Atindra Jha, and Christopher Pondoc

Introduction

For our CS 224W project, our team will evaluate the performance of graph neural network (GNN) methods on relational databases using RelBench. In particular, we'll use the open-source implementation of Relational Deep Learning (RDL) as a launching pad and explore several choices around architecture and scale to optimize performance. During this process, we would also like to extend the existing RelBench package to enhance the user experience by allowing users to add their own SQL datasets to the pipeline for easy training and evaluation. The aim is to spark more excitement and investigations into the intersection of GNNs and relational data by adopting RelBench and PyG.

Project Scope

RelBench

RelBench is a benchmark for solving predictive tasks over relational data with GNNs. The project, developed at Stanford, consists of sample relational DB tables and predictive tasks over these tables. To give users an approachable context for our tutorial, we'll focus on the `rel-amazon` Amazon e-commerce database. This database has 24,291,489 rows and three tables — `review`, `customer`, and `product` tables — and contains information about the time a user reviewed an item, its rating, and product metadata.

For this dataset, we'll evaluate the node classification task of `user-churn`, which predicts whether a user will or will not review a product in the next 3 months. `user-churn` is evaluated on the metric area under the receiver operating characteristics (AUROC).

Relational Deep Learning

The RelBench project also provides an open-source implementation of a technique known as Relational Deep Learning (RDL). Whereas older methods for working with relational data have used manual feature engineering and tabular models, RDL employs GNNs alongside deep tabular models to perform predictions. This architecture is optimal for our dataset, enabling

direct learning over multiple tables instead of aggregating database data into a single table. The core pipeline is as follows:

- **Heterogeneous temporal graph**: Each table represents a node type, each row represents a node, and primary-foreign-key relationships represent an edge.
- **Deep learning model**: Deep tabular models encode row-level data into initial node embeddings. We then use a GNN to update node embeddings.
- **Temporal-aware subgraph sampling**: Sample a subgraph around each entity node at a given seed time (to ensure no information leakage). The subgraph is then inputted to GNN to predict the target label.
- **Task-specific prediction head + loss:** Apply an MLP on the entity embedding to make a binary cross-entropy loss prediction.

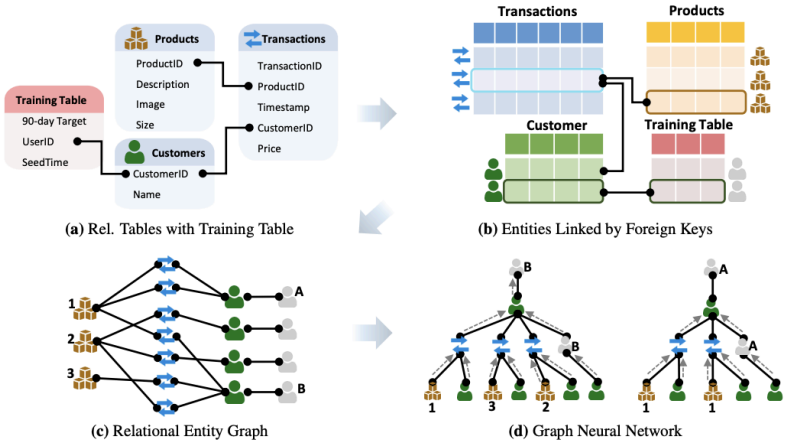

Figure 1: A high-level overview of the Relational Deep Learning pipeline.

Project Design

Baselines

As with the paper, we will provide several baselines to compare against the performance of RDL. While the paper uses a LightGBM classifier for the baseline for entity classification tasks, we will consider employing other methods, such as tabular models without the GNN component and more basic models like Random Forests. These more robust baselines can help provide more nuance as to why the ensemble of RDL is/isn't better for relational data.

Extensions

As noted above, the core implementation is open-sourced by the RelBench team. Thus, instead of just going step-by-step through the RDL implementation, we will also try out combinations of downstream tasks, GNN architectures, aggregation and message passing algorithms, and so

on, and analyze the various performance metrics over them. Below are just some of the areas for exploration:

- **Tabular Models:** Users can experiment with other tabular architectures in the deep learning component. TabNet, while older than the ResNet tabular model used by the team, seems like an apt choice.
    - *About TabNet:* TabNet repeatedly parses over a table's rows for question answering by using sequential attention to selectively find important features from each row it passes over. The architecture employs "instance-wise feature selection," which focuses on different features for each row.

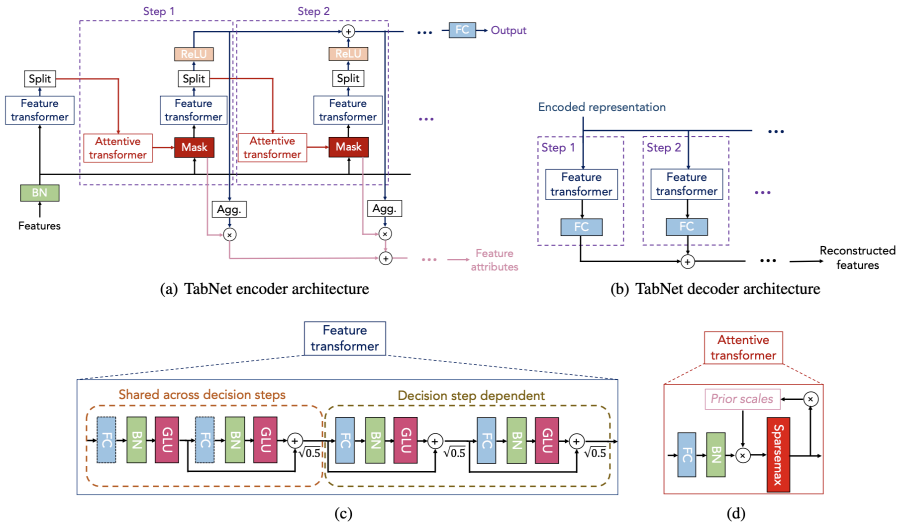

Figure 2: The encoder, decoder, and transformer architecture behind TabNet.

- **GNN Models:** RDL's GNN architecture is GraphSAGE with a sum aggregation function. Some easy experiments can be run here, including graph convolutional networks (GCNs) or graph attention networks (GATs). We can also vary aggregation and message-passing methods to observe their impact on downstream performance.
- **Varying Tasks:** It would be interesting to evaluate the performance of the generated embeddings on a task different from what the architecture was trained for but on the same relational dataset. For example, seeing how well it does on `item-churn` the `rel-amazon` dataset instead of just `user-churn`.

Conclusion

Our goal is to encourage more adoption and experimentation with RelBench by providing more insight into a new way to perform deep learning on relational databases. We believe our tutorial offers a good amount of content without overextending ourselves, and we aim to build the project incrementally such that each ablation is more of a stretch goal for future exploration.