MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

Cardinality Estimation Using Multi Modal ML

Abhinit Mahajan, Dr. Laurent Bindschaedler

Overview

This project focuses on developing a hybrid model architecture to estimate the cardinality of SQL queries accurately. By leveraging both the textual and structural representations of SQL queries, the proposed model aims to provide a robust and efficient solution to the cardinality estimation problem, ultimately improving query execution in database systems. Github

Background and Motivation

Cardinality estimation is a crucial task in optimizing query execution plans. Traditional methods rely on statistical summaries and heuristics, which often fail for complex or novel queries. Recent advancements in machine learning present opportunities to address these limitations. This project seeks to utilize transformer-based models and graph neural networks to capture the semantic and structural aspects of SQL queries, with the motivation to achieve more precise cardinality predictions and enhance database performance.

Research Questions / Hypotheses

The key hypothesis is that combining the structural representation of SQL queries (through a GNN) with their tokenized context (via BERT) will improve cardinality estimation compared to traditional or single-modal approaches. This hybrid approach is expected to provide a more nuanced understanding of query features, resulting in more accurate predictions.

Query Processing

Data Processing Overview

This section outlines the data processing methodologies employed for preparing SQL queries as inputs to the hybrid model. The SQL queries are processed in two formats: as a graph representation for the GNN component and as tokenized text for the transformer (BERT) component.

Graph Neural Network Processing

The **GNN processing** involves converting SQL queries into graph representations that capture both structure and relationships within the query.

Steps involved:

1. **Parsing SQL Query into Graph Representation**: The SQL query is parsed using the `sqlglot` library, resulting in an Abstract Syntax Tree (AST). This tree is further converted into a directed graph using the `networkx` library. Each node in the graph represents an element of the SQL query (e.g., a table or a condition).
2. **Node and Edge Creation**:
   - Nodes represent elements such as expressions, columns, or operators, with features based on node types.
   - Directed edges denote the relationships between elements, effectively capturing the structure of the query.
   - Node features are stored as integer values representing the type of each node, normalized for consistency.
3. **Node Features and Edge Index Extraction**:
   - Node features are extracted and normalized to have zero mean and unit variance.
   - The **edge index** is created to represent the connectivity between nodes.
   - The `torch_geometric` library is used to convert these graph attributes into a format suitable for the GNN.
4. **Output**:
   - A graph data object containing normalized node features and edge indices for the GNN.

Transformer Processing

The **Transformer (BERT) processing** is responsible for transforming SQL queries into a sequence of tokens suitable for processing by the transformer model. The `BertTokenizer` is used to preprocess the SQL text.

Steps involved:

1. **Tokenization and Encoding**:
   - The SQL query is tokenized using (`bert-base-uncased`).
   - The tokenizer converts SQL queries into token IDs and creates attention masks.
2. **Padding and Truncation**:
   - The tokenized inputs are padded for uniform sequence length and truncated to the specified maximum length 512.
   - The data is returned in the form of PyTorch tensors, which are compatible with the BERT model.
3. **Output**:
   - A dictionary containing input IDs and attention masks for each SQL query.

Architecture Outline

The proposed model architecture for SQL Cardinality Estimation combines transformers and graph neural networks (GNNs) in a hybrid design that processes both textual and structural representations of SQL queries. The architecture consists of two components: a transformer-based model (BERT) for SQL query text and a graph-based model (ResidualGNN) for the graph representation of the query. The outputs of these models are fused to predict the estimated cardinality..

Transformer Component: BERT

The transformer model used is BERT (Bidirectional Encoder Representations from Transformers), specifically using a pre-trained version of BERT, `bert-base-uncased`. This component extracts contextual information from the SQL query text. The SQL query is tokenized and passed through BERT, generating token-level embeddings. The `pooler_output` from BERT, which is a 768-dimensional vector summarizing the entire sequence, is used as the output representation of the query text.

Graph Neural Network Component: ResidualGNN

The second component of the architecture is a graph neural network, termed **ResidualGNN**. This model processes the graph representation of the SQL query, where nodes represent various query components, and edges define relationships between these components. The **ResidualGNN** is built using two layers of **Graph Isomorphism Networks (GINConv)**.

The ResidualGNN utilizes the following structure:

- **GIN Layers**: Two GINConv layers extract node-level features, transforming input features into higher-dimensional representations.
- **Residual Connections**: Skip connections maintain the identity of original input features, improving gradient propagation.
- **Batch Normalization**: Each GIN layer is followed by batch normalization to stabilize training.
- **Global Pooling**: Global mean pooling aggregates node-level features into a graph-level embedding.

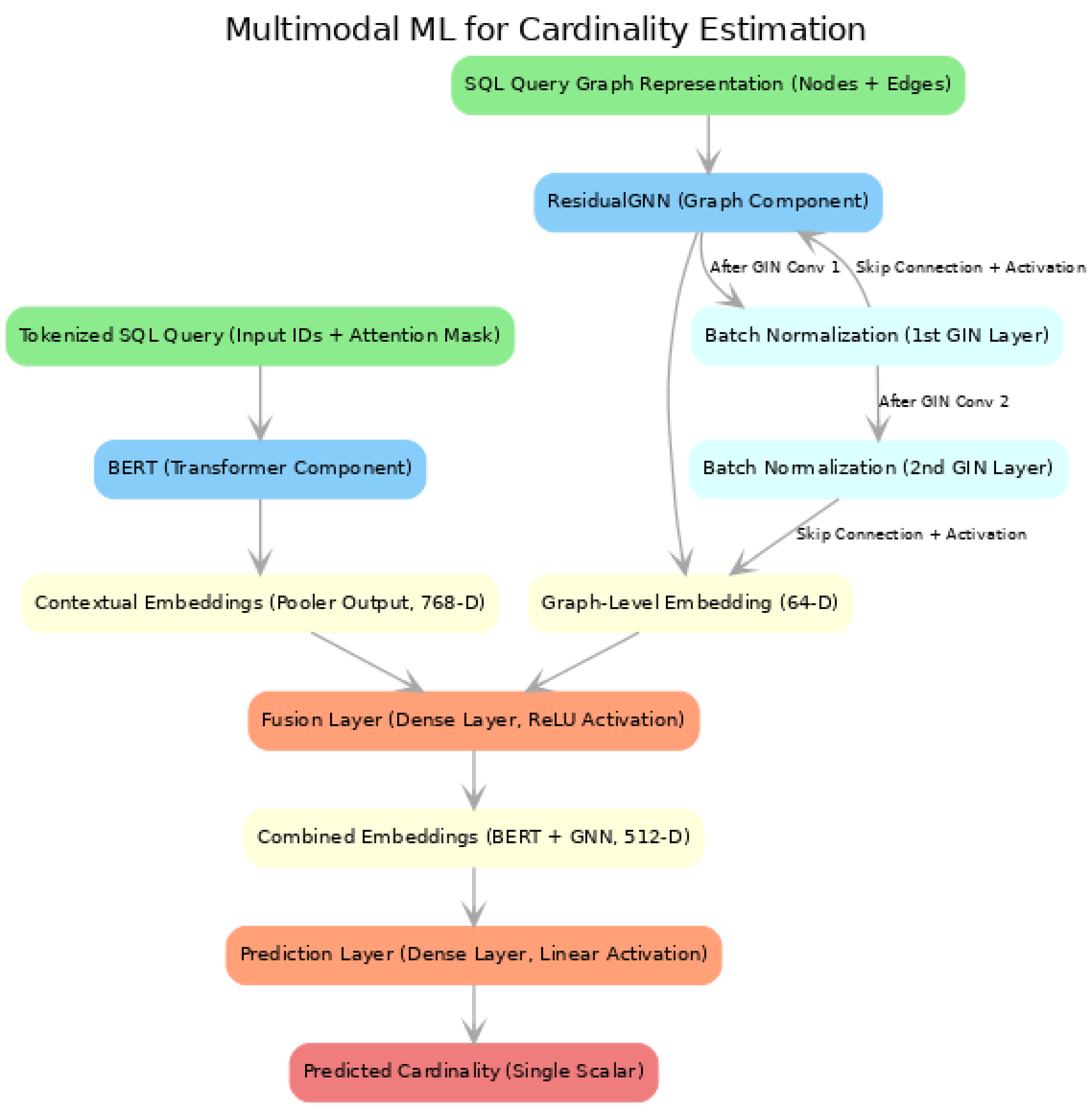
Fusion Layer

The **fusion layer** is responsible for combining the representations produced by **BERT** and **ResidualGNN**. The outputs of both models are concatenated to form a joint representation of the SQL query. Since the BERT output has a dimension of 768, and the GNN output has a dimension of 64, they are concatenated and passed through a dense layer of size 512 with relu activation functions. The final **prediction layer** consists of a fully connected network that outputs the cardinality estimation. The combined feature vector from the fusion layer is passed through this output layer to produce a single scalar value representing the predicted cardinality.

Model Training

The model is trained end-to-end using Mean Squared Error (MSE) as the regression loss function, minimizing the difference between predicted and actual cardinality values.

This hybrid approach leverages both the natural language context of SQL statements and the structural information in graph representations, providing a comprehensive solution to the cardinality estimation problem.



Multimodal ML for Cardinality Estimation
SQL Query Graph Representation (Nodes + Edges)
ResidualGNN (Graph Component)
After GIN Conv 1
Skip Connection + Activation
Tokenized SQL Query (Input IDs + Attention Mask)
Batch Normalization (1st GIN Layer)
After GIN Conv 2
BERT (Transformer Component)
Batch Normalization (2nd GIN Layer)
Skip Connection + Activation
Contextual Embeddings (Pooler Output, 768-D)
Graph-Level Embedding (64-D)
Fusion Layer (Dense Layer, ReLU Activation)
Combined Embeddings (BERT + GNN, 512-D)
Prediction Layer (Dense Layer, Linear Activation)
Predicted Cardinality (Single Scalar)

Results Discussion

The preliminary results from our hybrid model for cardinality estimation indicate a clear trade-off between training and testing performance. When achieving high training accuracy ($R > 0.8$), the model tends to overfit, resulting in significantly lower testing accuracy ($R < 0.3$). However, by constraining training accuracy to a moderate range (R between 0.70 and 0.75), the model generalizes better, achieving an R score of approximately 0.53 on the test set. These results suggest that careful regularization and fine-tuning are essential to balance model complexity and improve generalization performance. Despite the moderate R score, these findings are promising and indicate potential for further refinement and optimization.

Future Direction

To advance this research, several important directions need to be explored. First, a rigorous evaluation of the proposed hybrid model against other state-of-the-art (SOTA) methods for cardinality estimation is crucial. By benchmarking against existing models, we can quantify the performance gains and identify potential areas for improvement. Additionally, the inclusion of a third modality—database characteristics—represents a significant enhancement to the current approach. Traditional methods for cardinality estimation have utilized features such as table sizes, index statistics, and data distributions to inform their predictions. Integrating these database-specific features into our multimodal architecture could provide a more holistic understanding of the underlying data, enhancing the robustness and accuracy of cardinality estimates, particularly for complex queries and diverse database environments. A multimodal ML architecture incorporating query text, query structure, and database characteristics could outperform existing methods by providing a comprehensive representation of the database context.

Another promising direction is to investigate how this methodology can contribute to the generation of synthetic databases. Cardinality estimation is directly linked to data distribution and the relationships between data entities. By leveraging our model's understanding of these relationships, we could potentially generate realistic, representative synthetic data that mimics the properties of real-world databases. This could be invaluable for testing and development purposes, providing a controlled environment to evaluate query optimizers and database management systems without relying on sensitive or proprietary data. Future work could involve developing a pipeline where the estimated cardinality from queries informs a generative model to create synthetic datasets that maintain key statistical properties, making them suitable for use in various benchmarking and testing scenarios.