RoTaR: Efficient Row-Based Table Representation Learning via Teacher-Student Training

Zui Chen
Tsinghua University
Beijing, China
chenzui19@mails.tsinghua.edu.cn

Lei Cao
MIT/U of Arizona
Massachusetts, USA
caolei@mit.edu

Sam Madden
MIT
Massachusetts, USA
madden@csail.mit.edu

Abstract

We propose RoTaR, a row-based table representation learning method, to address the efficiency and scalability issues faced by existing table representation learning methods. The key idea of RoTaR is to generate query-agnostic row representations that could be re-used via query-specific aggregation. In addition to the row-based architecture, we introduce several techniques: cell-aware position embedding, teacher-student training paradigm, and selective backward to improve the performance of RoTaR model.

1 Introduction

Tabular data is one of the most widely used media for storing information. Table representation learning has a wide range of downstream applications such as table question answering, table search, table type detection, etc. It is thus vital to design an effective and practical solution for table representation learning. However, despite its popularity and importance in modern data science, table representation learning is not well addressed, compared to images, texts, or other media.

Most of the previous attempts [20, 11, 8, 7, 14, 19, 1] apply recent progress in natural language processing (NLP), i.e., transformers and large language models (LMs). These works directly serialize the entire table together with a query or related utterance into a sequence as the input to an LM, which is pretrained on a sufficient amount of table corpus. However, as the most common and best practices for table representation learning, this approach suffers from scalability and efficiency issues.

First, serializing a large table containing a large number of rows will result in a long sequence which is hard to process by classical transformer-based models, because the complexity of such models is quadratic to the length of the input sequence. To solve this problem, some works optimize the transformer structure [15], while others use table specific solutions to reduce the complexity of attention computation, such as restricting attention computation to the same row or column [6, 9] or only between the schema and values [4]. However, these approaches do not eliminate the scalability issue, because a pretrained LM is subject to a max sequence length constraint. For example, GPT-3 limits the input length to 2048 tokens, while BERT sets this limit as 512. A table with a small number of rows can easily exceeds it, causing inevitable truncation and thus loss of information.

Second, the serialization process takes the given query as input, leading to query-specific encoding. Because in real-world scenarios many queries concentrate on a few tables, repeatedly computing the table representation for every new incoming query is inefficient.

To address the above problems, we proposed RoTaR which learns a query-agnostic row-based table representation. Rather than serialize the whole table, RoTaR takes each row as input and efficiently produces row level encodings which can be re-used by any queries. It then uses query-specific aggregation to produce the table representation on top of these row encodings.

Table Representation Learning workshop at NeurIPS 2022
2 ROTA R Methodology

2.1 Overall Architecture

Row Independence Observation. Independencies in the table structure are the key to reduce the computational complexity of transformer-based models: irrelevant attention can be saved in transformer-based models without harming their performance. We observe that although the information in different rows should be aggregated to form the representation of the whole table, there is no strong correlation among different rows.

Intuitively, a relational table can be viewed as a set of rows with homogeneous schema, because the order of the rows usually does not matter. The representation of a set is mathematically equivalent to an appropriate aggregation of the representations of each single item in the set \([16, 21]\). Therefore, table representation can be factorized into an aggregation of independent row representations.

Inspired by this observation, ROTA R uses a weight-shared row-based transformer model to encode each row in the table independently and ignores the inter-row correlation in this encoding (Fig. 1).

ROTA R consists of two components: query-agnostic row encoder \(M\) and query-specific aggregation. After training on a dataset, the learned row representations by the row encoder can be preprocessed and stored. Therefore, answering an upcoming query only requires computing the aggregation. It does not have to repeatedly run the row encoder, thus saving a massive amount of time.

Row Encoder. More specifically, ROTA R considers every cell \(T_{i,j}\) in a table \(T\) as a textual cell, i.e., \(T_{i,j} = T_{i,j;1}T_{i,j;2} \cdots T_{i,j;n}\) where each \(T_{i,j;k}\) is a token. Similarly, each attribute in the schema \(A\) is also viewed as textual, i.e., \(A_j = A_{j;1}A_{j;2} \cdots A_{j;m}\), where each \(A_{j;k}\) is a token. ROTA R thus serializes each cell \(c_{i,j}\) by combining the attribute and the cell value \(c_{i,j} = [\text{COL}] A_j [\text{VAL}] T_{i,j}\).

Given a table with \(N\) rows and \(L\) columns, a shared encoder \(M\) encodes each concatenated row \(c_i \coloneqq c_{i,1} \| c_{i,2} \| \cdots \| c_{i,L}\) into a fixed-dimension vector \(v_i = M(c_i)\). Notice that the obtained set of row representations \(\{v_1, v_2, \ldots, v_N\}\) is query-agnostic.

Aggregation. Then for each incoming query \(q\), the resulted table representation can be computed by \(v_q^T = \rho(\{\phi(c_i, q)\}_{i=1}^N)\), where \(\phi\) is a learnable function that given a query \(q\) extracts information from \(c_i\). \(\rho\) is an appropriate aggregation function \([16, 21]\). The function \(\phi\) and \(\rho\) together constitute a query-specific aggregation module. For instance, if a query \(q\) is encoded as a vector \(v_q\) of the same dimension with the row representations, setting \(\phi(c_i, q) = v_i \odot v_q\) (\(\odot\) stands for point-wise multiplication) and \(\rho(X) = \frac{1}{|X|} \sum_{x \in X} x\) yields the table representation as the average row vector projected onto the query vector \(v_q\).

2.2 Query-agnostic Row Encoder

The ROTA R model tackles the first fore-mentioned issue in scalability, i.e., encoding a table with a large number of rows. Because the transformer model is run separately for each row and the aggregation module does not have to use transformer-based models, ROTA R is capable of handling...
any tables with any number of rows. Note the number of columns is not of concern, because the
number of columns is usually much smaller than the number of rows.

The design of the query-agnostic row encoder \( M \) can be very flexible. For example, we can
directly use a general-purpose pretrained LM like BERT [5] or RoBERTa [12], or pre-existing table
representation models like TAPAS [8, 7] if we view each row as a table consisting of a single row. In
addition to directly adopt the existing methods, we also propose two new techniques customized to
row-based table representation learning.

When using a general-purpose pretrained LM as the row encoder \( M \), instead of directly feeding the
serialized row \( c_i = c_{i,1} \mid c_{i,2} \mid \cdots \mid c_{i,n} \) into the LM, we could take advantage of the structure of
row to elaborate the position embedding design. Specifically, the position embedding \( p^T_{i,j:k} \) of each
token \( T_{i,j:k} \) or \( p^A_{i,j:k} \) of each token \( A_{i,j:k} \) can be decomposed into inter-cell position embeddings (based
on \( j \) or \( A_j \)) and intra-cell position embeddings (based on \( k \)) [3], which can then be customized for
different purpose of use (Fig. 2a). For example, in many circumstances the order of attributes should
be irrelevant to the representation of the row. By simply removing the absolute position embedding
and the cell index embedding, the order of attributes is not perceived by the LM and thus the learned
representation is robust against swapping order of columns.

2.3 Query-specific Aggregation

The RoTAR model also tackles the fore-mentioned issue in efficiency, i.e., learning representation re-usable for different queries. While the learned row-based representation is query-agnostic, a query-specific aggregation module is introduced to produce query-specific table representation.

The design of query-specific adaption function \( \phi \) can be very straightforward, for example, \( \phi(v_i, q) = v_i \oplus v_q \) or \( \phi(v_i, q) = \text{MLP}(v_i \oplus v_q) \) (\( \oplus \) stands for concatenation) or even \( \phi(v_i, q) = \text{MLP}(v_i \oplus v_q \oplus (v_i - v_q) \oplus (v_i \odot v_q)) \). Furthermore, notice that \( \phi \) does not have to be differentiable or even numeric, traditional selective \( \phi \) based on the textual input \( c_i \) could also be used. For example, the n-gram similarity weighted embedding \( \phi(c_i, q) = \text{ns}(c_i, q) \cdot v_i = \frac{2 \cdot \text{n-gram}(c_i) \cdot \text{n-gram}(q)}{\text{n-gram}(c_i) + \text{n-gram}(q)} \cdot v_i \), or a hard threshold \( \phi_h(c_i, q) = \left[ \text{ns}(c_i, q) > \alpha \right] \cdot v_i \).

The choice of aggregation function \( \rho \) can be rather arbitrary, for example, Mean, Min, Max, LogSumExp, etc., are all feasible aggregation functions. However, the choice of aggregation naturally influences the table representation quality when handling different queries. Therefore, a learnable aggregation function \( \rho \) could make the model more flexible. For instance, we could learn a multi-
head projection aggregation function, which has a set of learnable parameters \( \Theta = \{\theta_i\}_{i=1}^d \), and
\[
\rho_{\Theta}(X) = \frac{1}{|X|} \sum_{x \in X} \text{Nonlinear}(x \odot \theta_i).
\]

2.4 Training Techniques

2.4.1 Teacher-Student Paradigm

RoTAR simplifies the table representation process by ignoring inter-row interactions. Therefore, it pursues efficiency and scalability at the expense of inevitable but acceptable performance decay. However, with the help of the previous table representation methods during training time, the RoTAR model is able to improve its performance while still being efficient during inference time.
Consider the teacher-student paradigm which is widely used in model distillation \cite{10, 13, 17}. Training the student model to mimic the features generated by the highly performant teacher model can provide more differentiable information and boost the student model’s learning process. The small and efficient student model is then used as a cost effective alternative to the original teacher model.

Technically, instead of only considering the loss function $L = L_{task}(M_S)$ of the downstream task during training, where $M_S$ is the student model, the loss in teacher-student paradigm uses $L = \alpha \cdot L_{task}(M_T) + \beta \cdot L_{task}(M_S) + \gamma \cdot d(M_T, M_S)$, where $M_T$ is the teacher model, $d$ is the mean squared error distance between features or logits generated by the teacher and student model, and $\alpha, \beta, \gamma$ are tunable hyperparameters (Fig. 2b).

### 2.4.2 Selective Backward

In practice, back-propagation through the aggregation of multiple rows could be unacceptable because of the GPU memory limit. However, since the row encoder $M$ is shared, RoTaR is able to only sample some rows to back-propagate. The sampling process could be done either randomly or weighted according to a traditional selective $\phi$, like n-gram similarity.

### 3 Experiments

We conduct preliminary experiments with the table fact verification task on the TabFact \cite{2} dataset. In the table verification task, a query $q$ is a statement to be evaluated based on a table $T$. The TabFact dataset consists of 16K tables obtained from Wikipedia and 118K labeled statements. A common public data split is provided along with the dataset. The results are shown in Tab. 1.

For fair comparison, we use the same model size to bert-base and google/tapas-base. Further, because the TAPAS_{BASE} best result\footnote{The TAPAS_{BASE} and TAPAS_{BASE} (TabFact only) result is reported by \cite{7}, while the Table-BERT result is reported by \cite{2}.} (78.5% ± 0.3%) is pretrained on 6.2M table-text examples obtained from Wikipedia, we compare against the results reported by TAPAS using only TabFact data (69.9% ± 3.8%) and Table-BERT (65.1%).

Note since the RoTaR model prioritizes efficiency and scalability over performance, it slightly sacrifices performance in exchange for big speed up. The performance sacrifice mainly comes from the query-agnostic property instead of the row-independency. In particular, with a an accuracy drop of $\sim 6\%$, the RoTaR model with preprocessed feature vector is $\sim 3.7x$ faster than the TAPAS model, depending on the speed of the query encoder.

Further, we also compare against a TAPAS model that same to RoTaR, is not aware of the queries beforehand. We then finetuned it under the same setting to RoTaR. This TAPAS model achieves an accuracy of 50.3%, which is much lower than the accuracy of our RoTaR (63.6%). This confirms that RoTaR is indeed able to produce query-agnostic representation.

### 4 Conclusion

We propose RoTaR which uses a shared row-encoder to generate query-agnostic row representations and learns instance optimized aggregation function to produce query-specific table representation. Preliminary experiments on TabFact confirm that RoTaR significantly improves the scalability and
efficiency of table representation learning, with limited performance drop. In the further, we will continue to optimize the RoTAR model to improve the speed-accuracy trade-off.

References


A Appendix: Experiment Settings

We use the huggingface implementation of transformer models including BERT and TAPAS. All models share the parameter of virtual batch size 64, learning rate $2 \times 10^{-5}$, weight decay $10^{-5}$, the AdamW optimizer, and cosine annealing with 2 warm-up epochs. The training process uses early stopping of patience 8. All experiments are run in half-precision on a cloud server with a single NVIDIA Tesla V100 Tensor Core GPU.

All features are extended to the same dimension of 2048 by a transformation module, which is a two-layer neural network with hidden size equal to the original feature dimension (768 in case of BASE models), LeakyReLU activation with negative slope 0.01 and dropout probability 0.1. All models share the same downstream binary classifier, which is a three-layer neural network with hidden size 2048, LeakyReLU activation with negative slope 0.01 and dropout probability 0.1. The query encoder is a separate transformer BASE model.