Teaching Old DB Neu(ral) Tricks: Learning Embeddings on Multi-tabular Databases

Garima Gaur*
IIT Delhi
Delhi, India
garimag@cse.iitd.ac.in

Rajat Singh*
IIT Delhi
Delhi, India
rajat.singh@cse.iitd.ac.in

Siddhant Arora
CMU
Pennsylvania, USA
siddhantarora1806@gmail.com

Vinayak Gupta
IIT Delhi
Delhi, India
vinayak.gupta@cse.iitd.ac.in

Srikanta Bedathur
IIT Delhi
Delhi, India
srikanta@cse.iitd.ac.in

ABSTRACT

The lack of robust representation learning techniques tailored for relational data has led to the underwhelming application of ML models to handle database relevant downstream tasks. Recent works that attempt to embed tabular data into a low dimensional latent space have focused solely on Web tables. A relational database is quite different from a Web table corpus and is way more sophisticated. Existing approaches cannot handle the intricacy of relational databases and often fail to learn meaningful embeddings. To this end, we propose an attention based novel learning technique called RelBert, that intelligently computes the context of entities and learns column semantic aware embeddings. We have implemented an end-to-end system, RelBert, and have evaluated its performance for two tasks, missing value imputation (MVI) and instance classification. RelBert has reduced the mean rank by ~40% on an average for MVI task compared to the state-of-the-art approach.

KEYWORDS

Relational database, attention model, representation learning

ACM Reference Format:
Garima Gaur, Rajat Singh, Siddhant Arora, Vinayak Gupta, and Srikanta Bedathur. 2023. Teaching Old DB Neu(ral) Tricks: Learning Embeddings on Multi-tabular Databases. In *6th Joint International Conference on Data Science & Management of Data (10th ACM IKDD CODS and 28th COMAD) (CODS-COMAD 2023), January 4–7, 2023, Mumbai, India.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3570991.3571041

1 INTRODUCTION

The majority of valuable data is still stored in traditional relational databases. In particular, data-centric applications like e-commerce, finance, and organization management systems are powered by enterprise database engines like IBM DB2, Apache Derby, Monet DB, Oracle DB, etc. However, the research question of employing sophisticated deep learning based models to improve the standard database issues like missing value imputation, data integration, classification, and so on is under-explored. It can be explained by the scarcity of robust representation learning approaches for relational databases. The success of deep-learning (DL) models in addressing the same issue in the domain of NLP [2, 12, 27, 32], computer vision[13, 18, 21, 23], and graphs [17, 20, 33] has led to significant improvement in downstream tasks like question-answering, co-reference resolution, text generation, summarization, object detection, link prediction, node classification, and so on. However, the inherent characteristics of relational databases such as their domain-specific vocabulary, the presence of explicit constraints on data, and the semantic types associated with entities, etc., result in the ineffectiveness of using these models off-the-shelf.

There have been a few recent efforts to tailor the DL models for handling the tabular data [1, 3, 7–11, 24, 35]. Based on which DL model is adapted, these can be categorized into two classes – transformer-based models and GNN-based models. The majority of works using transformer encoders work only with Web tables [9, 11, 24, 35]. Such tables are reasonably different from relational databases and are not as sophisticated as databases. Importantly, Web tables are derived from large encyclopedias like Wikipedia, where tokens (cell values) of Web tables are often associated with an entity in the knowledge source. For instance, in a Web table listing all UN members, the token *USA* can be associated with the entity *United_States_of_America* of Wikipedia. Existing solutions [11, 35] are heavily dependent on this association and restrict their focus to only tokens with an associated entity. General databases do not comply with this strong restriction and often have tokens that are part of the domain-specific vocabulary of the database. GNN-based solutions [3, 10] are more relational database-centric as they consider database schema while transforming tables to a graph. However, most of them rely on feature engineering as a pre-processing step.

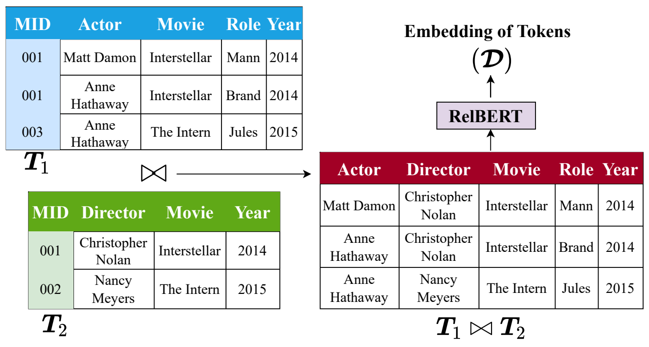
1.1 Challenges

We identify the following key challenges for the use of deep learning based methods for solving a variety of tasks in database settings:

Domain Specific Vocabulary: Many of the existing works [9, 11, 35] leverage the knowledge gathered by large language models (LLMs) like BERT [12], GPT[27], T5 [28], etc. These approaches

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CODS-COMAD 2023, January 4–7, 2023, Mumbai, India
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-9797-1/23/01…$15.00
https://doi.org/10.1145/3570991.3571041



MID	Actor	Movie	Role	Year
001	Matt Damon	Interstellar	Mann	2014
001	Anne Hathaway	Interstellar	Brand	2014
003	Anne Hathaway	The Intern	Jules	2015

MID	Director	Movie	Year
001	Christopher Nolan	Interstellar	2014
002	Nancy Meyers	The Intern	2015

Actor	Director	Movie	Role	Year
Matt Damon	Christopher Nolan	Interstellar	Mann	2014
Anne Hathaway	Christopher Nolan	Interstellar	Brand	2014
Anne Hathaway	Nancy Meyers	The Intern	Jules	2015

Figure 1: Schematic diagram of embedding learning procedure of RelBert for two tables from the IMDb dataset.

fine-tune these LLMs with their tabular data. However, a domain-specific database has its dedicated vocabulary, and language models are not trained on those. For instance, MIMIC is a data warehouse of anonymized hospitalization information of patients. Multiple tables in the database have columns that contain special medical codes that represent different medical emergencies, diagnosis groups, etc. In such cases, an LLM-based solution may turn out to be not helpful.

Web tables collection is not a database: Web tables and a database (collection of tables) can be differentiated on two factors that are relevant to this work, (i) coherence among tables and (ii) the table size (number of tuples).

- A database stores information about a few principal entities and relationships among them. These relationships are often expressed either using a pair of primary key, foreign key (*PK-FK* pair), or relationship tables that associate two primary entity tables. For instance, in the IMDb database, *Movie* and *Director* are the principal entities, and relationship table *Movies_Directors* relates each movie to its director(s). In essence, the information about a principal entity is spread across multiple tables. While on the other hand, each Web table is a standalone piece of information. Therefore, relational databases solution has an additional requirement of capturing the full context of the principal entities.
- Generally, database tables are orders of magnitude larger than Web tables. Existing attention-based models [9, 11, 35] serialize the entire table, one tuple followed by the next, and feed this sequence as an input to the model. Usually, attention-based models can take an input of size 512, and a flattened relational table is likely to exceed this limit by a huge margin. Thus, flattening a table for input construction is impractical in database settings.

Same token different semantics: Tabular data is more structured than textual data, and each token in a table is tightly coupled with a semantic type. For example, entity *New_Delhi* can have multiple semantics, like *geo-location*, *capital*, *union_territory*, associated to it, but the column to which the token *New_Delhi* belongs dictates its semantic type. Further, a single token can appear in different columns, but its semantics can be different based on the column. E.g., in the IMDb database, token *1991* under the *date_of_birth* column of an actor is different from its presence in *release_year* column in *movie* table. The same token under different columns has different types. Therefore, capturing the semantics of tokens in the database is essential.

1.2 Contributions

In this work, we are focusing on the use of transformer-based models with relational databases and make the following key contributions:

- We propose an end-to-end self-supervised learning system, called RelBert, that employs an attention-based encoder tailored for relational databases. The novel learning strategy of embedding tokens in a column to a single latent space effectively captures the token type information. To the best of our knowledge, this is the first work to provide a solution at the cross-section of attention models and relational databases.
- Similar to [10], we utilize specified referential integrity constraints of a database to define the complete context of a principal entity in a database. Particularly, we work with *Primary Key-Foreign Key* joins to capture the context and then generate the input sequences for pre-training an attention-based encoder from scratch .
- We assess the quality of embedding using two database-specific tasks, *viz.*, missing value imputation and instance classification, over two real-world datasets –IMDb and MIMIC-III– and observe a significant improvement.

Our codebase is made public at: https://github.com/data-iitd/relbert.

2 RELATED WORK

This section will provide an overview of the research surrounding the projection of tabular data into latent space. Numerous efforts have been made to learn the mapping of tabular data to latent space [1, 3, 5, 6, 8–11, 26, 34, 37, 38]. Each method focuses on data at a different granularity of detail, from representing the entire table as single vector [37, 38] to mapping each token of the table to a vector [5, 8]. From considering tables as graph [3, 10] to considering them as linear sentences [1, 8, 9]. From treating a tuple in the table as a sentence [1] to constructing sentences using random walk on tables [8].

With the increasing research on tabular data, are addressing various table-related tasks, such as [37, 38] learning the table embeddings with the intent of finding related tables. Recently, the focus has shifted to addressing downstream tasks such as missing value imputation, join prediction, schema matching, question-answering, and link prediction.

Bai et al. [3] and Cvitkovic [10] use GNN-based technique to learn tabular data embeddings. Typically, These models initially associate the graphical ideas of nodes, edges, and vertices with the table's defining attributes of rows, columns, and cell entities. [3] transforms multiple tables with relationships into a heterogeneous hyper-graph where vertices are joinable attributes and tuples are treated as the hyper-edges. Furthermore, [10] transforms relational tables into a directed multi-graph in which rows are viewed as nodes and foreign key references are handled as directed vertices. These graphs are later fed into GNN, which learns embeddings from the respective graph. Graph-based models helped in solving challenges that conventional NNs were unable to address adequately. However, extensive feature engineering is required for these models.

The universal model TURL [11], is a transformer-based framework for learning deep contextualized representations of table entities. It learns embeddings for each entity during pre-training and uses a visibility matrix to capture intra-row and intra-column relations of entities in the table. Furthermore, to get the final embedding, it uses type embedding, position embedding, mention representation, and entity embedding of the entities and fed them to a structure-aware transformer with Masked Language Model (MLM) and Masked Entity Recovery (MER) as the learning objective. TURL not only uses a huge amount of metadata for pre-training but also requires an external Knowledge Base (KB) for capturing the semantics of entities in the table. It is hard to find external KB for the datasets with domain-specific vocabulary.

Bordawekar and Shmueli [5] and Bordawekar and Shmueli [6] focused on answering cognitive queries over relational data . Their approach involves interpreting rows of tables as natural language (NL) sentences and then training a word2vec model to embed entities in a latent space. [26] aimed to use learned embeddings to identify nontrivial patterns from the database that helps in predicting appropriate policing matters. This way of simply interpreting table rows as NL sentences fail to capture the semantic relations between tables, which are often expressed via foreign key and primary key (FK-PK) pairs. We leverage such implicit information about data that is encoded in the database schema.

The state-of-the-art model EMBDI [8] builds a tripartite graph with each entity connected to the *row_ids* and *column_ids* of the table in which it resides. Furthermore, it generates sentences using a random walk over the tripartite graph and learns the embeddings of the entities. Sentences generated by random walks on a tripartite graph may contain entities that are not directly present in the same row or column of the table. EMBDI learns embeddings over the generated sentence corpus by using the standard NLP-based embedding method, word2vec.

Most of the approaches [9, 11] either work with Web tables and, thus, do not deal with the intricacies of a relational database. Even when they work with the relational data, they fail to fully leverage the implicit knowledge a database schema captures about the stored data. In this work, we do not treat entities just as simple words in NL sentence; rather, we focus on their semantics. For instance, if a syntactically same entity is present in two different columns, we treat them as two separate entities mapped to two different column aware latent spaces.

3 PRELIMINARY

This section presents background on self-attention models, the training tasks in RELBERT, and various notations used in this paper.

3.1 Attention based Encoders

Self-attention [32] based models have shown tremendous prowess in *sequence-to-sequence* tasks with applications in natural language [4, 12, 30], recommendation systems [19, 22, 31], and time series [15, 16, 29, 36, 39]. Specifically, the *transformer* architecture introduced in [32] involves an Encoder-Decoder architecture. An encoder is used to map an input sequence to a D dimensional hidden representation. This representation is later fed into the decoder for sequence generation. The encoder and decoder consist of multi-head attention for obtaining a weighted aggregation of all entities. Specifically, it uses dot-product attention defined as:

$$f_{\mathrm{Attn}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{D}}\right)\boldsymbol{V}, \quad (1)$$

Where f_{Attn} is the function to calculate attention weights, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ represent queries, keys, and values respectively. In more detail, these matrices are linear formulations of the input sequence.

3.2 Training Tasks

The goal of RELBERT is to model the inter-relationships between entities of a database system. To achieve this, RELBERT must be able to capture the interaction between entities in a tuple as well as across tuples. Therefore, inspired by BERT model [12], we formulate two tasks for RELBERT to optimize – masked language model(MLM) and next sentence prediction (NSP).

Masked Language Model (MLM). Buoyed by the efficacy of Cloze task in training natural language models [14], here we mask entities from a tuple in a relation and then predict the masked entities by jointly conditioning on the *left* and *right* context. These contexts are derived from a bidirectional self-attention mechanism [12]. The final vector obtained from the bidirectional self-attention model is fed into an output softmax over all entities with the same attribute.

Next Sentence Prediction (NSP). The task of NSP is designed to model the relationship between different sentences, *e.g.*, a question and the respective answer. Specifically, we feed both sentences into the self-attention model with a *start* and *end* vectors. Later, we use the soft-max output at indices of these vectors to predict the relationship between the sentences. NSP is necessary for RELBERT to learn the inter-tuple relationships between relations with *PK-FK* joins. Note that although in this work we use the PK-FK join relation between tuples from different tables, there can be other mechanisms for the same. We will leave this for future exploration.

3.3 Notations

- **Database:** We consider an RDBMS as a collection of coherent tables, denoted by $\mathcal{D} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_N\}$. Here, $\boldsymbol{T}_\bullet$ denotes the individual tables in the system. Moreover, we assume that every RDBMS includes a *schema* that specifies its structural characteristics, including the column headers, column data types, data-related constraints, properties of the columns of all tables, *etc.*
- **Table:** Each table $\boldsymbol{T}_j$ can be viewed as a grid with columns $c_m^j, \ldots c_n^j$, and each row $t_k^j = (val_{c_m}^k, \ldots, val_{c_n}^k)$ is an instance defined by value $val_\bullet$ corresponding to each column. When the table under consideration is clear from the context, we denote columns as $c_\bullet$ and rows as $t_\bullet$.
- **Token:** A *token* is the value stored in a cell of a table, i.e. the value $val_{c_\bullet}^\bullet$ corresponding to column $c_\bullet$ in row $t_\bullet$ of a table is referred as a *token.* Tokens are the basic unit of a database. In other words, tokens constructs rows, rows construct tables and tables lead to a database.

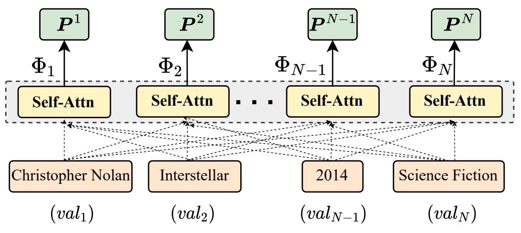


Figure 2: Schematic diagram of the embedding procedure in RelBert.

• **Primary Key:** A primary key is a column(s) that uniquely identifies each row of a table, i.e. $val^i_{c_{key}} = val^j_{c_{key}}$ *iff* $t_i = t_j$, where c_{key} is the primary key of the table.

• **Foreign Key:** Suppose tables $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ in database and c^i_{key} is the primary key of $\boldsymbol{T}_i$, then column c^j_f is called foreign key of table $\boldsymbol{T}_j$ if tokens $val^{\bullet}_{c_f}$ in table $\boldsymbol{T}_j$ are derived from tokens $val^{\bullet}_{c_{key}}$ in table $\boldsymbol{T}_i$, i.e. $val^{\bullet}_{c^j_f} \in TOK(c^i_{key})$, where $TOK(c_\bullet)$ is a set of all tokens corresponding to column $c_\bullet$. Note that using the *PK-FK* pair (c^i_{key}, c^j_f), we can join tables $\boldsymbol{T}_i$ and T_j. We denote the resultant joined table as $\boldsymbol{T}_{ij}$.

4 PROPOSED MODEL

In this section, we will describe in detail the architecture of our model RelBert.

4.1 Table Encoder

For any table $\boldsymbol{T}_\bullet$, our model incorporates a self-attention model for encoding the entities in a table. Rows of the table are used to generate sentences, which are then fed sequentially into the transformer. However, unlike a natural language sentence, each word in a table row belongs to a column in that table, and each column inherently contains distinct entity information. In Figure-1, for instance, an entity in the *actor* column captures entirely different semantics as compared to the identical entity in the *director* column. Thus, a homogeneous embedding initialization function will fail to incorporate the column heterogeneity. Therefore, we use distinct embedding spaces corresponding to each column in the table denoted by $\Phi^\bullet$ (Figure-2). We calculate the embedding matrix for the entities in the table as follows:

$$\boldsymbol{P}^k_i = \Phi^k(val^i_k), \qquad \forall val^i_k \in \boldsymbol{T}, \tag{2}$$

where Φ^k and i denote embedding function for k-th column and the row (or sentence) index in the relational table, respectively. Following [12], we utilize masked-language-based learning where we randomly mask one of the entities in a sentence and then predict those masked entities based on the transformer output. We compute the output embeddings for each entity through an encoder network consisting of multiple attention layers. We denote the output embeddings for each sentence indexed by i as $\boldsymbol{O}_i = \texttt{TEncoder}(\boldsymbol{P}_i)$, where $\texttt{TEncoder}(\bullet)$ denotes the base-transformer module. However, since each column k has a different embedding space, computed by Φ^k, we determine the candidate entities for the *masked* entity in the sentence using an output-softmax over the entities in the column of the masked entity. We learn the parameters for the horizontal transformer by optimizing the cross-entropy loss function for each sentence in the table $\boldsymbol{T}$ as:

$$\mathcal{L}_{mlm} = \sum_{i=1}^{|\boldsymbol{T}|} \texttt{CrossEntropy}(\boldsymbol{o}^m_i, \boldsymbol{P}^{mask}_i), \qquad \boldsymbol{o}^m_i \in \boldsymbol{O}, \tag{3}$$

where, $\boldsymbol{o}^m_i$, $\boldsymbol{P}^{mask}$ denote the output embedding corresponding to the masked index and the column entities, respectively. In addition, we exclude positional embeddings from the transformer model as a table is inherently *permutation-invariant*. We initialize each entity in a row of the table using standard word2vec [25] computed over the table by simply treating each row as a sentence. For cells with multi-word attributes, we concatenated all the words using '_' and converted them into a single word.

4.2 RelBert

RelBert learns column-aware semantic representations for each entity in a relational table. However, a relational table is organized into multiple tables, either to efficiently enforce integrity constraints or to enhance the overall database efficiency using the *data normalization* process. Therefore, in order for RelBert to learn semantic information between two linked entities spread across the tables, it requires information from multiple tables. For instance, *actor-genre* in the actor table and *director-genre* in the director table contribute to modelling the genre preference for both entities, which could assist a director in finding a plausible actor for his/her movie.

However, learning entity representations across multiple tables incurs a high computational cost due to the need to compute (explicitly or implicitly) all pairs of FK-PK joins between tables, commonly known as *denormalization* of the database. Secondly, a denormalized table degrades the data quality since it may merge unrelated attributes and dramatically increase table size by repeating the rows of the table multiple times.

To circumvent these bottlenecks, we model the table joins as a *next-sentence prediction* problem. Specifically, consider two tables $\boldsymbol{T}_\alpha$ and $\boldsymbol{T}_\beta$ that can be joined via the primary-key column $\boldsymbol{c}^\alpha_A$ in table $\boldsymbol{T}_\alpha$, and the corresponding *foreign-key* column $\boldsymbol{c}^\beta_{\hat{A}}$ in table $\boldsymbol{T}_\beta$. If a row containing entity $val^i_{c_\alpha} \in \boldsymbol{c}^\alpha_A$ is linked to the row containing entity $val^j_{c_\beta} \in \boldsymbol{c}^\beta_{\hat{A}}$, then the latter is considered as the *next sentence* for the current row. This will help the model learn the semantics of the PF-FK join between two or more related tuples when these tuples are present in different tables without necessitating their join materialization. We generated sentence pairs from these rows by separating them with [SEP] token and appending [CLS] and [SEP] tokens at the beginning and end, respectively. This sentence pair is later fed into RelBert to learn inter-table information for an entity. We utilize the output embedding at the `[CLS]` token to estimate the probability of a sentence being the *next* sentence, $v_{t^\alpha_i, t^\beta_j}$ and use a negatively sampled row $t'^\beta_j \in \boldsymbol{T}_\beta$. The loss for NSP is calculated via:

$$\mathcal{L}_{nsp} = -\sum_{i=1}^{|\boldsymbol{T}_\alpha|} \sum_{t^\alpha_i \in \boldsymbol{T}_\alpha} \left[\log\left(\sigma(v_{t^\alpha_i, t^\beta_j})\right) + \log\left(1 - \sigma(v_{t^\alpha_i, t'^\beta_j})\right) \right], \tag{4}$$

where t_i^{α} and t_j^{β} are i^{th} and j^{th} row in $\boldsymbol{T}_{\alpha}$ and $\boldsymbol{T}_{\beta}$ tables respectively and σ denotes the *sigmoid* function. Finally, we combine this next sentence prediction task with the masked language model learning of the predefined table encoder.

4.3 Optimization

Training RelBert for table MLM enables us to learn relationships between attributes, whereas the NSP task helps to aggregate inter-table information i.e., join between the tables. By introducing the next sentence prediction task, we develop the first table missing value imputation approach that avoids the bottlenecks of a denormalized table. RelBert parameters are optimised by minimising the net loss, which is the sum of the respective MLM and NSP cross entropy losses.

$$\mathcal{L}_A = \mathcal{L}_{mlm} + \mathcal{L}_{nsp}, \tag{5}$$

However, joining different tables at each training iteration would lead to an information loss over multiple iterations. Therefore, our training strategy consists of a two-step approach, with the first step as a general *pre-training* step, which is followed by task-specific *fine-tuning*.

Pre-training: To understand the relationships between tables and attributes during pre-training, we perform task-independent learning of all entities in the database. Specifically, we construct pairwise sentences between tables using the PK-FK joins in the relational schema and train the model parameters using standard MLM-based optimization.

Fine-Tuning: The pre-training step is followed by task-specific *fine-tuning* of the model. That is, for our task of table missing value imputation, we jointly fine-tune the RelBert model to predict the *masked* entity for each row and simultaneously the join-based next sentence by optimizing the net loss $\mathcal{L}_A$.

5 EXPERIMENTS

5.1 Setup

Datasets. We evaluated the performance of our model using the two fairly complex real databases, IMDb and MIMIC-III (Table-1).

• IMDb[1]: This database serves as an encyclopedia of different movies, tv series, documentaries, etc. We focused particularly on *movies*. The dataset has a total of 5 tables capturing different aspects of a movie, like its director(s), actor(s), movie genres, and so on.

• MIMIC-III[2]: It is a domain-specific database holding information about patients' visits to Beth Israel Deaconess medical center over a decade. Out of 40 total tables, we selected 3 tables that are concerned with the patient's demographics, diagnosis, and drugs prescribed at each visit to the hospital. It has a record of total 58976 patient visits.

Baselines. Recall, in Section 1, we categorize existing solutions in two classes based on how the tabular data is interpreted, either language model (LM) style training with the table as a sequence of tokens or GNN training with an entire database along with schema as a graph. Our proposed solution RelBert belongs to the former category, and thus, we select solutions from the same category as baselines. While the recent solutions in the LM category are developed for Web tables and cannot handle relational tables due to their size (discussed in Section 1.1). Effectively, we chose the following solutions as baselines that perform LM-style training and are robust enough to handle decent size relational tables,

• Table2Vec [5]: Bordawekar and Shmueli proposed a fairly simple and effective approach of training a word2vec model over a corpus of sentences constructed by treating each row of a table as a sentence. The intent of the solution is to handle cognitive queries over a relational database.

• Embdi [8]: It is a sophisticated framework that captures intra-column dependency by constructing a sentence corpus by performing multiple random walks over a tripartite graph. The graph is constructed such that the tokens in the same column or same row can be present within a sentence. Using the sentence corpus, they train a word2vec model to learn contextual embeddings of tokens in a table.

Evaluation tasks. We test different tabular data representation learning approaches for two relevant downstream tasks, *missing value imputation (MVI)* and *instance classification*.

• **Missing Value Imputation (MVI)**: We translate the problem of imputing a missing cell value to a token prediction task, were given a tuple with an empty cell, our aim is to predict the correct token in the cell. For each input sequence, we randomly chose a column (out of the columns shaded blue in Figure-3), and marked the token corresponding to the column as missing. The marked token is effectively the ground truth, and the task is to predict the value of the missing token.

• **Instance Classification**: For classification, we interpret one of the categorical columns as the class of the row. For IMDb, we used *movie_genre* as the class of each movie. There is a total of 22 unique movie genres. Each movie can have multiple genres; thus it is a multi-label classification problem. However, for MIMIC-III, we categorize patients based on the 5 different type of insurance; thus column *insurance* token served as the class. Unlike IMDb, this is a multi-class classification problem as a patient can hold a single health insurance.

Metrics. For MVI, all the baselines, along with RelBert, rank the tokens based on their softmax scores; therefore, for evaluation, we measure the performance using 3 metrics, mean rank (MR), mean reciprocal rank (MRR) and Hits@k. Mathematically,

$$MR = \frac{1}{n}\sum_{i=1}^{n} rank(token_i)$$

$$MRR = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{rank(token_i)}$$

$$Hit@k = \frac{|\{token : rank(token) < k\}|}{n}$$

where, n is size of the test set, and $|\{\bullet\}|$ computes the number of items in the set.

For the classification task, we used standard classification metrics – micro mean precision and recall. Mathematically,

$$\text{Micro Avg Precision} = \frac{\sum_{c\in C} TP}{\sum_{c\in C} TP + \sum_{c\in C} FP}$$

[1]https://relational.fit.cvut.cz/dataset/IMDb
[2]https://mimic.physionet.org/

Name	Year	Rank	Genre	Director_Name

(a) IMDb: `Movies` ⋈ `Movies_Genre` ⋈ `Movies_Director` ⋈ `Director`.

Adm_Time	Disch_Time	Adm_Type	Adm_Loc	Disch_Loc	Insurance	Language	Religion	M_Status	Ethnicity	Diagnosis	DRG_Code	DRG_Type	DRG_Severity	DRG_Mortality	ICD9_Code

(b) MIMIC-III: `Admissions`⋈ `DRGCodes` ⋈ `Diagnoses_ICD`.

Figure 3: Schema of the joined table with principle entity columns as underlined.

Table 1: Sentence corpus sizes for different models.

Approach	Sentence Count	
	IMDb	MIMIC-III
Table2Vec	66, 254	58, 976
Embdi	299, 004	297, 924
RelBert	87, 386	1, 536, 008

Table 2: MVI task performance comparison of RelBert against other baselines over IMDb and MIMIC-III.

Approach	IMDb			MIMIC-III		
	MR	MRR	Hit@10	MR	MRR	Hit@10
Table2Vec	44.252	0.0631	0.176	657.022	0.206	0.663
Embdi	52.302	0.036	0.0792	1036.53	0.175	0.6197
RelBert	24.976	0.16	0.321	205.576	0.64	0.906

$$\text{Micro Avg Recall} = \frac{\sum_{c\in C} TP}{\sum_{c\in C} TP + \sum_{c\in C} FN}$$

where, C is the set of classes, and TP, FN and FP stands for *true positive*, *false negative* and *false positive*.

Training Setup and Implementation. The (joined) tables that we used to construct the input sequence corpus of RelBert, as well as baseline systems has columns shown in Figure-3. The underlined column headers are the ones belonging to the principal entity tables, *movies* and *patient_admission* in IMDb and MIMIC-III, respectively. For Embdi, we perform 4 random walks per node for sentence generation for both IMDb and MIMIC-III datasets. Using the number of columns in the joined table as a heuristic, we generate walks of length 6 and 18 for IMDb and MIMIC-III, respectively. The sentence corpus size of each baseline model and RelBert is reported in Table-1.

For training RelBert, we split the sentence corpus into three partitions, train set (70%), validation set (15%), and test set (15%). Using the Adam optimizer with learning rate 0.001, we train the model for 50 epochs with a block size of 8. [3]

5.2 Performance Comparison

Missing Value Imputation (MVI). Recall that we address the issue of missing value imputation via cell value prediction, i.e., the model predicts the token corresponding to an empty cell. We report the performance of baseline systems and our proposed framework RelBert in Table-2. Note that for MIMIC-III dataset, while the mean rank (MR) of Embdi is poor as compared to that of Table2Vec, their $Hit@10$ percent is comparable. This points towards the disparity in the prediction capability of Embdi. While it is performing better in certain cases, it is ranking the other ones badly. In contrast, RelBert has a consistent performance as its MR is lowest in comparison to all other systems, and for almost one-third of the predictions, the answers are listed correctly in the top-10 predicted tokens. RelBert showcased a reduction of about 52% and 79% in the mean rank from that of Embdi over IMDb and MIMIC-III datasets, respectively.

Interestingly, the performance of RelBert over IMDb and MIMIC-III datasets is not comparable, i.e., RelBert is not clearly performing better on one dataset than the other. While the *mean rank* for IMDb is quite low as compared to that for MIMIC-III, the $Hits@10$ for MIMIC-III is significantly high in comparison to IMDb. This pattern indicates that RelBert favors some columns over other. The low MR for IMDb indicates the column (tokens) in IMDb are equally easy to predict, whereas MIMIC-III columns are either very easy or too tough to predict correctly. This can explain the undesirable high MR and desirable high $Hit@10$ on the MIMIC-III database. Later in Section 5.3, we have closely investigated the impact of different columns on the RelBert.

Instance Classification. For IMDb and MIMIC-III instance classification, we chose *movie_genre* and *insurance* columns, respectively, as the class labels. The choice of *movie_genre* resulted in a multi-label classification problem. In the case of MIMIC-III, the decision of treating *insurance* tokens as labels is dictated by the balanced distribution of unique values (tokens) of *insurance* column. We report the micro averaged precision ($P@k$) and recall ($R@k$) at different values of k in Table-3 and -4. We reported precision and recall at k=5, 10, 15, 20 for IMDb as it contain 22 classes (*movie_genre*). In contrast, we reported precision and recall at k=1, 3, 5 for MIMIC-III as it contain total 5 classes (*insurance*). For IMDb dataset, while the RelBert performs better than Embdi, interestingly, our simple baseline Table2Vec outperforms both Embdi and RelBert. In the case of Embdi, it can be attributed to the difference in sentence formation technique, i.e., possibly the sentence generated by random walks confused the model and led to a weak association between movies and their genres. However, the performance of all the models for the multi-label classification task is subpar. However, for MIMIC-III dataset, the performance is reasonable. RelBert has better precision as well as recall than that of Embdi.

Notice the difference in *precision-recall* trend across the datasets for all the 3 approaches. For IMDb, the precision and recall are increasing as the number of predicted tokens (k) under consideration is increasing. In contrast, for MIMIC-III, the trend is inverse, viz precision is decreasing, and the recall is increasing. The increase in both precision and recall indicates the gradual increase of *true positive* count. Intuitively, as more predicted tokens are considered, the chances of the presence of true labels (genres) of a movie in the predicted token list increase. On the other hand, the MIMIC-III

[3]https://github.com/data-iitd/relbert

Table 3: Classification task performance (micro precision $P@k$ and recall $R@k$) of RelBert against other baselines over IMDb.

Approach	k=5		k=10		k=15		k=20	
	P	R	P	R	P	R	P	R
Table2Vec	0.07	0.20	0.07	0.40	0.08	0.64	0.09	1
Embdi	0.03	0.07	0.04	0.24	0.06	0.47	0.09	1
RelBert	0.048	0.08	0.065	0.24	0.09	0.56	0.11	1

Table 4: Classification task performance (micro precision $P@k$ and recall $R@k$) of RelBert against other baselines over MIMIC-III.

Approach	k=1		k=3		k=5	
	P	R	P	R	P	R
Table2Vec	0.22	0.22	0.21	0.64	0.20	1
Embdi	0.44	0.44	0.28	0.84	0.20	1
RelBert	0.58	0.58	0.32	0.97	0.2	1

classification is a multi-class problem, and the decreasing precision can be viewed as the side-effect of the increasing value of k. The number of predicted tokens (k) is effectively the sum of the number of *true positives* and *false positives*. Therefore, with the increasing count of (true and false) positives, the precision decreases. The recall follows the expected pattern and is increasing with the value of k.

5.3 Ablation Study

We performed the following three exploratory studies to get insight into RelBert,

E1 *Can language models handle relational tabular data?*: Fine-tuning language model (BERT) Vs relational data tailored pre-training (RelBert)
E2 *Can different latent space capture semantic type information?*: Single Latent Space Vs Separate embedding space
E3 *Do columns impact the performance of RelBert?*: Drilling down for column-specific performance

E1: Fine-tuning Vs Pre-training. We finetuned BERT (base-uncased) model available at Huggingface[4]. For sentence corpus construction, we used a sentence template with placeholders for different column values (tokens). For each tuple in our joined table, we replaced the placeholders with the respective tokens and converted the tuple into a natural sentence. We report the performance of this finetuned model and RelBert in Table-5 for both datasets IMDb and MIMIC-III. Our model outperformed the finetuned BERT by a significant margin. Interestingly, for IMDb dataset finetuned BERT lists the correct token in $top10$ predicted tokens for around 20% of test instances, whereas, on MIMIC-III dataset, it never lists the correct token in $top-10$ tokens. This observation highlights the inability of pre-trained LM to handle domain-specific vocabulary.

E2: Single Latent space Vs Column-type driven embedding. With the strategy of mapping each column to a different latent space, we expect RelBert to capture the semantics of different columns in the database. To understand the impact of this innovative way

[4]https://huggingface.co/bert-base-uncased

Table 5: Performance comparison of (a) A model that fine-tunes the pretrained BERT with natural language sentences constructed using the tabular data; (b) RelBert-SS that embeds all the tokens to a same latent space; and (c) RelBert.

Approach	IMDb			MIMIC-III		
	MR	MRR	Hit@10	MR	MRR	Hit@10
Finetuned BERT	1182.308	0.086	0.207	1105.55	0.07	–
RelBert-SS	56.46	0.069	0.154	500.802	0.229	0.544
RelBert	24.976	0.16	0.321	205.576	0.64	0.906

of learning, we compare RelBert to the conventional approach of mapping all tokens to a single space.

While keeping all the hyperparameters the same as that of RelBert, we trained an attention encoder such that all the tokens are mapped to the same embedding space. We refer to this model as RelBert-SS. We report the performance of RelBert-SS and RelBert over IMDb and MIMIC-III datasets in Table-5. We observed a significant performance improvement of RelBert over the RelBert-SS. For both datasets, the MR is reduced by 50%, and $Hit@10$ is increased by 100%. This huge gap in the performance implies that the idea of different latent space for each column is critical to the success of our proposed model.

E3: Column specific performance assessment. The column-wise performance of RelBert, along with column statistics, are reported in Table-6 and Table-7 for IMDb and MIMIC-III datasets, respectively. Here, *cardinality* of a column gives the total number of unique values (tokens) in the column. We observed a clear trend across datasets that the lower cardinality columns are easier to predict, viz, column cardinality and its MR are directly proportional. In the case of IMDb, this relation is almost linear; for e.g., the ratio of cardinality (5.26) of column *year* to *genre* is the same as that of their respective mean ranks. For MIMIC-III, we did not observe any such strong pattern.

Notice that for columns *Language*, *Religion*, and *Ethnicity*, the respective MR is quite low despite their comparatively large cardinalities. A plausible explanation of this trend is the inter-column dependency, i.e., the value of one column is dependent on the other column. For instance, given the ethnicity of a person, it is possible to predict her language. The low MR in such cases implies that RelBert is capable of capturing inter-column dependencies.

5.4 Discussion

In summary, our empirical results led to the following observations:

• Despite the small training set size (70% of corpus), attention encoder based RelBert outperformed the word2vec based Embdi and Table2Vec. While this shows that attention is useful, it alone can not explain the superior performance of RelBert since RelBert outperforms fine-tuned BERT by quite a margin.

• Pre-trained language models are incapable of learning meaningful representation for domain-specific databases. Particularly, the results for MIMIC-III dataset highlighted the failure of pre-trained language models.

• As expected, RelBert performed better for lower cardinality columns.

• Finally, the key idea of embedding each column to a different latent space has led to the success of RelBert as we observed

Table 6: Column-wise performance of RelBert on IMDb.

Column	Cardinality	MR	Hit@5	Hit@10	Hit@20
Year	120	32.233	0.197	0.267	0.41
Rank	91	21.959	0.138	0.283	0.553
Genre	22	5.882	0.602	0.805	1

Table 7: Column-wise performance of RelBert on MIMIC-III.

Column	Cardinality	MR	Hit@1	Hit@3	Hit@5	Hit@10
Adm_Type	4	1.25	0.823	0.986	1	1
Adm_Loc	9	2.117	0.38	0.843	1	1
Disch_Loc	17	3.274	0.232	0.619	0.858	0.988
Insurance	5	1.591	0.578	0.967	1	1
Language	76	1.767	0.73	0.932	0.964	0.984
Religion	21	2.791	0.372	0.721	0.912	0.983
M_Status	8	2.127	0.442	0.84	0.984	1
Ethnicity	41	2.22	0.724	0.865	0.914	0.969
Diagnosis	15692	1826.313	0.085	0.116	0.157	0.229

a straight jump of 100% in $Hit@10$ of RelBert as compared to RelBert-SS.

6 CONCLUSION

In this work, we have presented a relational database tailored representation learning technique. Our model RelBert leverages database-specific features (referential integrity constraints) to intelligently construct input sequences from the tabular data. RelBert employs an innovative learning approach of embedding different columns to different latent space so as to capture column semantics. We extensively evaluated the performance of RelBert over two real datasets and observed a significant improvement as compared to that of state-of-the-art model Embdi. Further, the experimental results have reaffirmed the superiority of attention models for the embedding learning task and further validated our key idea of embedding different columns to different latent spaces.

Despite its promising results, RelBert still has limited scalability for handling practical databases with large number of rows and tables. Apart from scalability, RelBert still lacks the ability to work with non-categorical cell values. Our immediate future direction of research includes the incorporation of continuous numerical values in the table and the evaluation of these column-aware multi-tabular embeddings for other tasks, such as table question answering.

REFERENCES

[1] Sercan Ö Arik and Tomas Pfister. 2021. Tabnet: Attentive interpretable tabular learning. In *AAAI*.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[3] Jinze Bai, Jialin Wang, Zhao Li, Donghui Ding, Ji Zhang, and Jun Gao. 2021. ATJ-Net: Auto-Table-Join Network for Automatic Learning on Relational Databases. In *WWW*.

[4] Ivan Bilan and Benjamin Roth. 2018. Position-aware Self-attention with Relative Positional Encodings for Slot Filling. *arXiv preprint arXiv:1807.03052* (2018).

[5] Rajesh Bordawekar and Oded Shmueli. 2017. Using Word Embedding to Enable Semantic Queries in Relational Databases. In *DEEM*.

[6] Rajesh R. Bordawekar and Oded Shmueli. 2019. Exploiting Latent Information in Relational Databases via Word Embedding and Application to Degrees of Disclosure. In *CIDR*.

[7] Shaofeng Cai, Kaiping Zheng, Gang Chen, H. V. Jagadish, Beng Chin Ooi, and Meihui Zhang. 2021. ARM-Net: Adaptive Relation Modeling Network for Structured Data. In *SIGMOD*.

[8] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *SIGMOD*.

[9] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. TabFact : A Large-scale Dataset for Table-based Fact Verification. In *ICLR*.

[10] Milan Cvitkovic. 2020. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046* (2020).

[11] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2021. TURL: Table Understanding through Representation Learning. In *VLDB*.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

[13] Yazan Abu Farha, Alexander Richard, and Juergen Gall. 2018. When Will You Do What? - Anticipating Temporal Occurrences of Activities. In *CVPR*.

[14] William Fedus, Ian Goodfellow, and Andrew M. Dai. 2018. MaskGAN: Better Text Generation via Filling in the _. In *ICLR*.

[15] Vinayak Gupta and Srikanta Bedathur. 2022. ProActive: Self-Attentive Temporal Point Process Flows for Activity Sequence. In *KDD*.

[16] Vinayak Gupta, Srikanta Bedathur, and Abir De. 2022. Learning Temporal Point Processes for Efficient Retrieval of Continuous Time Event Sequences. In *AAAI*.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[18] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. 2017. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In *CVPR*.

[19] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*.

[20] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*.

[22] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*.

[23] Tahmida Mahmud, Mahmudul Hasan, and Amit K. Roy-Chowdhury. 2017. Joint prediction of activity labels and starting times in untrimmed videos. In *ICCV*.

[24] Kushal Majmundar, Sachin Goyal, Praneeth Netrapalli, and Prateek Jain. 2022. MET: Masked Encoding for Tabular Data. *arXiv preprint arXiv:2206.08564* (2022).

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *NeurIPS*.

[26] Apoorva Nitsure, Rajesh R. Bordawekar, and Jose Neves. 2020. Unlocking New York City Crime Insights using Relational Database Embeddings. *arXiv preprint arXiv:2005.09617* (2020).

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* (2019).

[28] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR* (2020).

[29] Karishma Sharma, Yizhou Zhang, Emilio Ferrara, and Yan Liu. 2021. Identifying Coordinated Accounts on Social Media through Hidden Influence and Group Behaviours. In *KDD*.

[30] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL-HLT*.

[31] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[34] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. 2012. InfoGather: Entity Augmentation and Attribute Discovery By Holistic Matching with Web Tables. In *SIGMOD*.

[35] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian" Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *ACL*.

[36] Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. 2020. Self-attentive Hawkes processes. In *ICML*.

[37] Shuo Zhang and Krisztian Balog. 2018. Ad Hoc Table Retrieval Using Semantic Similarity. In *WWW*.

[38] Shuo Zhang and Krisztian Balog. 2019. Recommending Related Tables. *arXiv preprint arXiv:1907.03595* (2019).

[39] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. 2020. Transformer Hawkes Process. In *ICML*.