# TABBIE: Pretrained Representations of Tabular Data

**Hiroshi Iida**[†] **Dung Thai**[‡] **Varun Manjunatha**[§] **Mohit Iyyer**[‡]
[†]Sony Corporation [‡]UMass Amherst [§]Adobe Research
hiroshi.iida@sony.com
{dthai,miyyer}@cs.umass.edu
vmanjuna@adobe.com

## Abstract

Existing work on tabular representation-learning *jointly* models tables and associated text using self-supervised objective functions derived from pretrained language models such as BERT. While this joint pretraining improves tasks involving paired tables and text (e.g., answering questions about tables), we show that it underperforms on tasks that operate over tables without any associated text (e.g., populating missing cells). We devise a simple pretraining objective (*corrupt cell detection*) that learns exclusively from tabular data and reaches the state-of-the-art on a suite of table-based prediction tasks. Unlike competing approaches, our model (TABBIE) provides embeddings of all table substructures (cells, rows, and columns), and it also requires far less compute to train. A qualitative analysis of our model's learned cell, column, and row representations shows that it understands complex table semantics and numerical trends.

## 1 Introduction

Large-scale self-supervised pretraining has substantially advanced the state-of-the-art in natural language processing (Peters et al., 2018; Devlin et al., 2018; Liu et al., 2019). More recently, these pretraining methods have been extended to jointly learn representations of *tables* as well as text (Herzig et al., 2020; Yin et al., 2020), which enables improved modeling of tasks such as question answering over tables. However, many practical problems involve semantic understanding of tabular data without additional text-based input, such as extracting tables from documents, retrieving similar columns or cells, and filling in missing information (Zhang and Balog, 2020). In this work, we design a pretraining methodology specifically for tables (**Tab**ular **I**nformation **E**mbedding or TABBIE) that resembles several core tasks in table extraction and decomposition pipelines and allows easy access to representations for different tabular substructures (cells, rows, and columns).

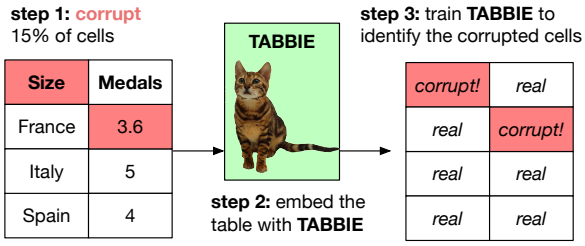



Figure 1: TABBIE is a table embedding model trained to detect corrupted cells, inspired by the ELECTRA (Clark et al., 2020) objective function. This simple pretraining objective results in powerful embeddings of cells, columns, and rows, and it yields state-of-the-art results on downstream table-based tasks.

Existing table representation models such as TaBERT (Yin et al., 2020) and TaPas (Herzig et al., 2020) concatenate tabular data with an associated piece of text and then use BERT's masked language modeling objective for pretraining. These approaches are computationally expensive due to the long sequences that arise from concatenating text with linearized tables, which necessitates truncating the input sequences[1] to make training feasible. We show that TaBERT underperforms on downstream table-based applications that operate independent of external text (e.g., deciding whether cell text was corrupted while extracting a table from a PDF), which motivates us to investigate an approach that preserves the full table during pretraining.

Our TABBIE architecture relies on two Transformers that independently encode rows and columns, respectively; their representations are pooled at each layer. This setup reduces the sequence length of each Transformer's input, which cuts down on its complexity, while also allowing us

[1] Herzig et al. (2020) use a fixed limit of 128 tokens for both text and table, while Yin et al. (2020) drop all but three rows of the table during pretraining.



*Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456
June 6–11, 2021. 

to easily extract representations of cells, rows, and columns. Additionally, TABBIE uses a simplified training objective compared to masked language modeling: instead of predicting masked cells, we repurpose ELECTRA's objective function (Clark et al., 2020) for tabular pretraining by asking the model to predict whether or not each cell in a table is real or corrupted. We emphasize that this pretraining objective is a fundamental task in table structure decomposition pipelines (Nishida et al., 2017; Tensmeyer et al., 2019; Raja et al., 2020), in which incorrectly predicting row/column separators or cell boundaries leads to corrupted cell text. Unlike Clark et al. (2020), we do not require a separate "generator" model that produces corrupted candidates, as we observe that simple corruption processes (e.g., sampling cells from other tables, swapping cells within the same column) yield powerful representations after pretraining.

In a controlled comparison to TaBERT (pretraining on the same number of tables and using a similarly-sized model), we evaluate TABBIE on three table-based benchmarks: column population, row population, and column type prediction. On most configurations of these tasks, TABBIE achieves state-of-the-art performance, outperforming TaBERT and other baselines, while in others it performs competitively with TaBERT. Additionally, TABBIE was trained on **8** V100 GPUs in just over a week, compared to the **128** V100 GPUs used to train TaBERT in six days. A qualitative nearest-neighbor analysis of embeddings derived from TABBIE confirms that it encodes complex semantic properties about textual and numeric cells and substructures. We release our pretrained models and code to support further advances on table-based tasks.[2]

## 2 Model

TABBIE is a self-supervised pretraining approach trained exclusively on tables, unlike prior approaches (Herzig et al., 2020; Yin et al., 2020) that jointly model tables and associated text snippets. At a high level, TABBIE encodes each cell of a table using two different Transformer models (Vaswani et al., 2017), one operating across the rows of the table and the other across columns. At each layer, the representations from the *row* and *column* Transformers are averaged and then passed as input to the next layer, which produces a contextualized representation of each cell within the table. We place a binary classifier over TABBIE's final-layer cell representations to predict whether or not it has been *corrupted*, or replaced by an intruder cell during preprocessing, inspired by the ELECTRA objective of Clark et al. (2020). In the remainder of this section, we formalize both TABBIE's model architecture and pretraining objective.

### 2.1 Model Architecture

TABBIE takes an $M \times N$ table as input and produces embeddings $\boldsymbol{x}_{ij}$ for each cell (where $i$ and $j$ are row and column indices, respectively), as well as embeddings for individual columns $\boldsymbol{c}_j$ and rows $\boldsymbol{r}_i$.

**Initialization:** We begin by initializing the cell embeddings $\boldsymbol{x}_{ij}$ using a pretrained BERT model (Devlin et al., 2018).[3] Specifically, for each cell $(i, j)$, we feed its contents into BERT and extract the 768-$d$ [CLS] token representation. This step allows us to leverage the powerful semantic text encoder of BERT to compute representations of cells out-of-context, which is important because many tables contain cells with long-form text (e.g., *Notes* columns). Additionally, BERT has been shown to encode some degree of numeracy (Wallace et al., 2019), which helps represent cells with numerical content. We keep this BERT encoder fixed during training to reduce computational expense. Finally, we add learned positional embeddings to each of the [CLS] vectors to form the initialization of $\boldsymbol{x}_{ij}$. More specifically, we have two sets of positional embeddings, $p_i^{(r)} \in \mathbb{R}^H$ and $p_j^{(c)} \in \mathbb{R}^H$, which model the position of rows and columns, respectively, and are randomly initialized and fine-tuned via TABBIE's self-supervised objective.

**Contextualizing the cell embeddings:** The cell embeddings we get from BERT are uncontextualized: they are computed in isolation of all of the other cells in the table. While methods such as TaBERT and TaPaS contextualize cell embeddings by linearizing the table into a single long sequence, we take a different and more computationally manageable approach. We define a *row* Transformer, which encodes cells across each row of the table, as well as a *column* Transformer, which does the same across columns.

Concretely, assume row $i$ contains cell embeddings $\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \ldots, \boldsymbol{x}_{i,N}$. We pass this se-

[2] https://github.com/SFIG611/tabbie

[3] We use the BERT-base-uncased model in all experiments.

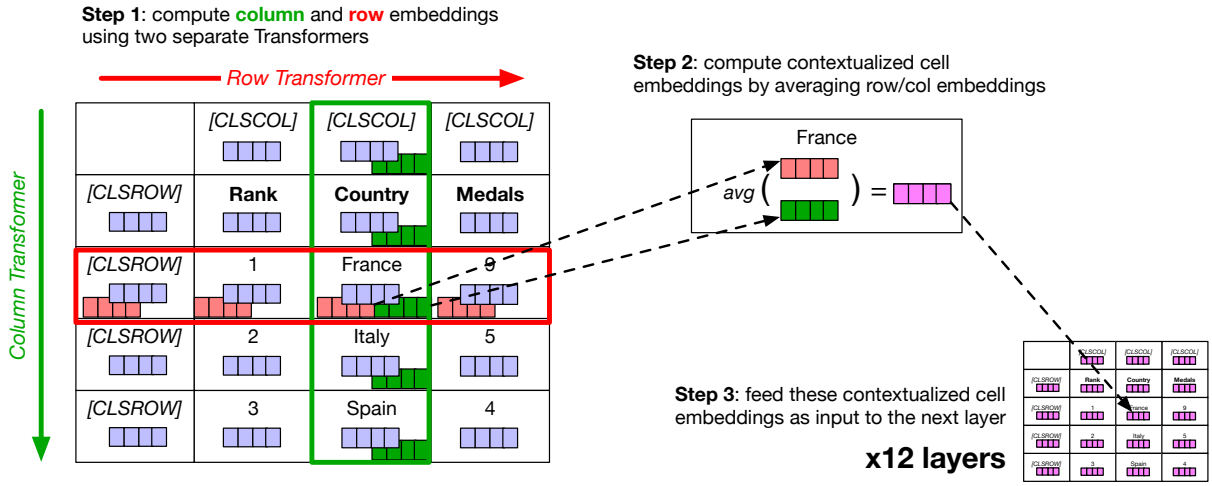



Figure 2: TABBIE's computations at one layer. For a given table, the row Transformer contextualizes the representations of the cells in each row, while the column Transformer similarly contextualizes cells in each column. The final cell representation is an average of the row and column embeddings, which is passed as input to the next layer. [CLS] tokens are prepended to each row and column to facilitate downstream tasks operating on table substructures.

quence of embeddings into a *row* Transformer block, which uses self-attention to produce contextualized output representations $\boldsymbol{r}_{i,1}, \boldsymbol{r}_{i,2}, \ldots, \boldsymbol{r}_{i,N}$. Similarly, assume column $j$ contains cell embeddings $\boldsymbol{x}_{1,j}, \boldsymbol{x}_{2,j}, \ldots, \boldsymbol{x}_{M,j}$; the *column* Transformer produces contextualized representations $\boldsymbol{c}_{1,j}, \boldsymbol{c}_{2,j}, \ldots, \boldsymbol{c}_{M,j}$. After running the two Transformers over all rows and columns, respectively, each cell $(i, j)$ of a table is associated with a row embedding $\boldsymbol{r}_{i,j}$ as well as a column embedding $\boldsymbol{c}_{i,j}$.

The final step of cell contextualization is to compose the row and column embeddings together before feeding the result to the next layer. Intuitively, if we do not aggregate the two sets of embeddings together, subsequent layers of the model will only have access to information from a specific row or column, which prevents contextualization across the whole table. We implement this aggregation through simple averaging: specifically, at layer $L$ of TABBIE, we compute cell embeddings as:

$$\boldsymbol{x}_{i,j}^{L+1} = \frac{\boldsymbol{r}_{i,j}^{L} + \boldsymbol{c}_{i,j}^{L}}{2} \quad (1)$$

The new cell representations $\boldsymbol{x}_{i,j}^{L+1}$ are then fed to the row and column Transformers at the next layer $L + 1$.

**Extracting representations of an entire row or column:** The row and column Transformers defined above produce separate representations for every cell in a particular row or column. However, many table-related downstream tasks (e.g., retrieve similar columns from a huge dataset of tables to some query column) can benefit from embeddings that capture the contents of an entire row or column. To enable this functionality in TABBIE, we simply prepend [CLSROW] and [CLSCOL] tokens to the beginning of each row and column in an input table as a preprocessing step. After pretraining, we can extract the final-layer cell representations of these [CLS] tokens to use in downstream tasks.

## 2.2 Pretraining

Having described TABBIE's model architecture, we turn now to its training objective. We adapt the self-supervised ELECTRA objective proposed by Clark et al. (2020) for text representation learning, which places a binary classifier over each word in a piece of text and asks if the word either is part of the original text or has been corrupted. While this objective was originally motivated as enabling more efficient training compared to BERT's masked language modeling objective, it is especially suited for tabular data, as corrupt cell detection is actually a fundamental task in table structure decomposition pipelines such as (Nishida et al., 2017; Tensmeyer et al., 2019; Raja et al., 2020), in which incorrectly predicted row/column separators or cell boundaries can lead to corrupted cell text.

In our extension of ELECTRA to tables, a binary classifier takes a final-layer cell embedding as input to decide whether it has been corrupted. More concretely, for cell $(i, j)$, we compute the

corruption probability as

$$P_{\text{corrupt}}(\text{cell}_{i,j}) = \sigma(\boldsymbol{w}^\intercal \boldsymbol{x}^L_{i,j}) \quad (2)$$

where $L$ indexes TABBIE's final layer, $\sigma$ is the sigmoid function, and $\boldsymbol{w}$ is a weight vector of the same dimensionality as the cell embedding. Our final loss function is the binary cross entropy loss of this classifier averaged across all cells in the table.

### 2.3 Cell corruption process

Our formulation diverges from Clark et al. (2020) in how the corrupted cells are generated. In ELECTRA, a separate generator model is trained with BERT's masked language modeling objective to produce candidate corrupted tokens: for instance, given *Jane went to the* [MASK] *to check on her experiments*, the generator model might produce corrupted candidates such as *lab* or *office*. Simpler corruption strategies, such as randomly sampling words from the vocabulary, cannot induce powerful representations of text because local syntactic and semantic patterns are usually sufficient to detect obvious corruptions. For tabular data, however, we show that simple corruption strategies (Figure 3) that take advantage of the intra-table structure actually do yield powerful representations without the need of a separate generator network. More specifically, we use two different corruption strategies:

- **Frequency-based cell sampling**: Our first strategy simply samples corrupt candidates from the training cell frequency distribution (i.e., more commonly-occurring cells are sampled more often than rare cells). One drawback of this method is that oftentimes it can result in samples that violate a particular column type (for instance, sampling a textual cell as a replacement for a cell in a numeric column). Despite its limitations, our analysis in Section 4 shows that this strategy alone results in strong performance on most downstream table-based tasks, although it does not result in a rich semantic understanding of intra-table semantics.

- **Intra-table cell swapping**: To encourage the model to learn fine-grained distinctions between topically-similar data, our second strategy produces corrupted candidates by swapping two cells in the same table (Figure 3c, d). This task is more challenging than the frequency-based sampling strategy above, especially when the swapped cells occur within the same column. While it underperforms frequency-based sampling on downstream tasks, it qualitatively results in more semantic similarity among nearest neighbors of column and row embeddings.

(a) original table

| Rank | Country | Gold |
|---|---|---|
| 1 | France | 9 |
| 2 | Italy | 5 |
| 3 | Spain | 4 |

(b) sample cells from other tables

| Rank | Size | Gold |
|---|---|---|
| 1 | France | 3.6 |
| 2 | Italy | 5 |
| 3 | Spain | 4 |

(c) swap cells on the same row

| Rank | Country | Gold |
|---|---|---|
| 1 | France | 9 |
| 2 | 5 | Italy |
| 3 | Spain | 4 |

(d) swap cells on the same column

| Rank | Country | Gold |
|---|---|---|
| 1 | France | 9 |
| 3 | Italy | 5 |
| 2 | Spain | 4 |

Figure 3: The different cell corruption strategies used in our experiments.

### 2.4 Pretraining details

**Data:** We aim for as controlled of a comparison with TaBERT (Yin et al., 2020) as possible, as its performance on table QA tasks indicate the strength of its table encoder. TaBERT's pretraining data was not publicly released at the time of our work, but their dataset consists of 26.6M tables from Wikipedia and the Common Crawl. We thus form a pretraining dataset of equivalent size by combining 1.8M Wikipedia tables with 24.8M preprocessed Common Crawl tables from Viznet (Hu et al., 2019).[4]

**Experimental settings:** We train TABBIE for seven epochs for just over a week on 8 V100 GPUs using mixed precision. TABBIE has 12 layers and a hidden dimensionality of 768 for both row and column Transformers, in an effort to be comparably-sized to the TaBERT-Base model.[5] Before computing the initial cell embeddings using BERT, we truncate each cell's contents to the first 300 characters, as some cells contain huge amounts of text. We also truncate tables to 30 rows and 20 columns to avoid memory issues (note that this is much larger than the three rows used by TaBERT), and

[4] The vast majority of text in these tables is in English.

[5] TABBIE is slightly larger than TaBERT-Base (170M to 133M parameters) because its row and column Transformers are the same size, while TaBERT places a smaller "vertical" Transformer over the output of a fine-tuned BERT model.

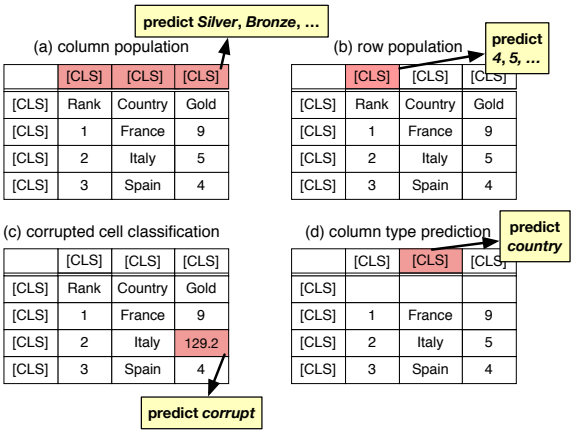



Figure 4: The inputs and outputs for each of our table-based prediction tasks. Column type prediction does not include headers as part of the table.

our maximum batch size is set at 4,800 cells (on average, 104 tables per batch). We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-5.

We compared two pretrained models trained with different cell corruption strategy for downstream tasks. The first strategy (**FREQ**) uses exclusively a frequency-based cell sampling. The second strategy is a 50/50 mixture (**MIX**) of frequency-based sampling and intra-table cell swapping, where we additionally specify that half of the intra-table swaps must come from the same row or column to make the objective more challenging.

## 3 Experiments

We validate TABBIE's table representation quality through its performance on three downstream table-centric benchmarks (column population, row population, and column type prediction) that measure semantic table understanding. In most configurations of these tasks, TABBIE outperforms TaBERT and other baselines to set new state-of-the-art numbers. Note that we do *not* investigate TABBIE's performance on table-and-text tasks such as WikiTableQuestions (Pasupat and Liang, 2015), as our focus is not on integrating TABBIE into complex task-specific pipelines (Liang et al., 2018), although this is an interesting avenue for future work.

### 3.1 Fine-tuning TABBIE

In all of our downstream experiments, we apply essentially the same fine-tuning strategy to both TABBIE and TaBERT: we select a subset of its final-layer representations (i.e., cell or column representations) that correspond to the tabular substructures used in the downstream task, and we place a classifier over these representations to predict the training labels. We select task-specific hyperparameters based on the size of each dataset (full details in Table 1) and report the test performance of the best-performing validation checkpoint. For both models, we backpropagate the downstream error signal into all of the model's parameters (i.e., we do not "freeze" our pretrained model).

| Task | Batch size | LR | Max epochs |
|---|---|---|---|
| Column population | 12 | 1e-05 | 20 |
| Row population | 48 | 2e-05 | 30 |
| Col. type prediction | 12 | 2e-05 | 15 |

Table 1: Fine-tuning hyperparameters of each downstream task for TABBIE and TaBERT.

### 3.2 Column Population

In the column population task, which is useful for attribute discovery, tabular data augmentation, and table retrieval (Das Sarma et al., 2012), a model is given the first $N$ columns of a "seed" table and asked to predict the remaining column headers. Zhang and Balog (2017) compile a dataset for this task comprising 1.6M tables from Wikipedia with a test set of 1,000 tables, formulated as a multi-label classification task with 127,656 possible header labels. Importantly, we remove all of the tables in the column population test set from our pretraining data to avoid inflating our results in case TABBIE memorizes the missing columns during pretraining.[6]

To fine-tune TABBIE on this task, we first concatenate the column [CLSCOL] embeddings of the seed table into a single vector and pass it through a single linear and softmax layer, training with a multi-label classification objective (Mahajan et al., 2018). Our baselines include the generative probabilistic model (**GPM**) of Zhang and Balog (2017) as well as a word embedding-based extension called Table2VecH (**TH**) devised by Deng et al. (2019). As fine-tuning on the full dataset is extremely expensive for TABBIE and TaBERT, we fine-tune on a random subset of 100K training examples; as a further disadvantage to these, we do not use table captions (unlike GPM and GPM+TH) during training. Nevertheless, as Table 2 shows, TABBIE and TaBERT substantially outperform both

[6]Note that TaBERT's pretraining data likely includes the test set tables, which may give it an advantage in our comparisons.

| N | Method | MAP | MRR | Ndcg-10 | Ndcg-20 |
|---|---|---|---|---|---|
| 1 | GPM | 25.1 | 37.5 | - | - |
| | GPM+TH | 25.5 | 0.38.0 | 27.1 | 31.5 |
| | TaBERT | 33.1 | 41.3 | 35.1 | 38.1 |
| | TABBIE (FREQ) | **37.9** | **49.1** | **41.2** | **43.8** |
| | TABBIE (MIX) | 37.1 | 48.7 | 40.4 | 43.1 |
| 2 | GPM | 28.5 | 40.4 | - | - |
| | GPM+TH | 33.2 | 44.0 | 36.1 | 41.3 |
| | TaBERT | 51.1 | 60.1 | 54.7 | 56.6 |
| | TABBIE (FREQ) | **52.0** | **62.8** | **55.8** | **57.6** |
| | TABBIE (MIX) | 51.7 | 62.3 | 55.6 | 57.2 |
| 3 | GPM | 28.5 | 35.5 | - | - |
| | GPM+TH | 40.0 | 50.8 | 45.2 | 48.5 |
| | TaBERT | 53.3 | 60.9 | 56.9 | 57.9 |
| | TABBIE (FREQ) | **54.5** | **63.3** | **57.9** | **58.9** |
| | TABBIE (MIX) | 54.1 | 62.3 | 57.4 | 58.7 |

Table 2: TABBIE outperforms all methods on the column population task, with the biggest improvement coming with just a single seed column ($N = 1$). Despite its simplicity, the **FREQ** corruption strategy yields better results than **MIX**.

baselines, and TABBIE consistently outperforms TaBERT regardless of how many seed columns are provided, especially with only one seed column. This result indicates that TABBIE encodes more semantics about headers and columns than TaBERT.

### 3.3 Row Population

The row population task is more challenging than column population: given the first $N$ rows of a table in which the first column contains entities (e.g., "Country"), models must predict the remaining entries of the first column. Making reasonable predictions of which entities best fill the column requires understanding the full context of the seed table. The Zhang and Balog (2017) dataset also contains a split for row population, which we use to evaluate our models. Again, since the dataset is too large for our large embedding models, we sample a subset of tables for fine-tuning.[7] Our label space consists of 300K entities that occur at least twice in Wikipedia tables, and we again formulate this problem as multi-label classification, this time on top of the first column's [CLSCOL] representation.[8]

On this task, TaBERT and TABBIE again outperform the baseline Entitables model (which uses external information in the form of table captions). When given only one seed row, TaBERT slightly outperforms TABBIE, but with more seed rows, TABBIE exhibits small improvements over TaBERT.

| N | Method | MAP | MRR | Ndcg-10 | Ndcg-20 |
|---|---|---|---|---|---|
| 1 | Entitables | 36.8 | 45.2 | - | - |
| | TaBERT | **43.2** | **55.7** | **45.6** | **47.7** |
| | TABBIE (FREQ) | 42.8 | 54.2 | 44.8 | 46.9 |
| | TABBIE (MIX) | 42.6 | 54.7 | 45.1 | 46.8 |
| 2 | Entitables | 37.2 | 45.1 | - | - |
| | TaBERT | 43.8 | 56.0 | 46.4 | 48.8 |
| | TABBIE (FREQ) | **44.4** | **57.2** | **47.1** | **49.5** |
| | TABBIE (MIX) | 43.7 | 55.7 | 46.2 | 48.6 |
| 3 | Entitables | 37.1 | 44.6 | - | - |
| | TaBERT | 42.9 | 55.1 | 45.6 | 48.5 |
| | TABBIE (FREQ) | **43.4** | **56.5** | **46.6** | **49.0** |
| | TABBIE (MIX) | 42.9 | 55.5 | 45.9 | 48.3 |

Table 3: TABBIE outperforms baselines on row population when provided with more seed rows $N$, although TaBERT is superior given just a single seed row. Again, the **FREQ** strategy produces better results than **MIX**.

### 3.4 Column Type Prediction

While the prior two tasks involve predicting missing elements of a table, the column type prediction task involves predicting a high-level *type* of a particular column (e.g., *name*, *age*, etc.) without access to its header. This task is useful when indexing tables with missing column names, which happens relatively often in practice, or for schema matching(Hulsebos et al., 2019; Rahm and Bernstein, 2001), and like the other tasks, requires understanding the surrounding context. We evaluate our models on the same subset of VizNet Web Tables (Hu et al., 2019)[9] created by Zhang et al. (2019) to evaluate their column type predictor, SATO[10]. They formulate this task as a multi-class classification problem (with 78 classes), with a training set of 64,000 tables and a test set consisting of 16,000 tables. We set aside 6,400 training tables to form a validation for both TABBIE and TaBERT, and we fine-tune each of these models with small random subsets of the training data (1000 and 10000 labeled tables) in addition to the full training set to evaluate their performance in a simulated low-resource setting.

Along with TaBERT, we compare with two recently-proposed column type prediction meth-

[7] We sample all tables that have at least five entries in the left-most column, which results in roughly 200K tables.

[8] Due to the large number of labels, we resort to negative sampling during training instead of the full softmax to cut down on fine-tuning time. Negative samples are formed by uniform random sampling on the label space.

[9] Again, we ensure that none of the test tables in this dataset occur in TABBIE's pretraining data.

[10] https://github.com/megagonlabs/sato

| Method | $n$=1000 | $n$=10000 | $n$=all |
|---|---|---|---|
| Sherlock | - | - | 86.7 |
| SATO | - | - | 90.8 |
| TaBERT | **84.7** | 93.5 | **97.2** |
| TABBIE (FREQ) | **84.7** | **94.2** | 96.9 |
| TABBIE (MIX) | 84.1 | 93.8 | 96.7 |

Table 4: Support-weighted F1-score of different models on column type prediction. TaBERT and TABBIE perform similarly in low resource settings ($n$=1000) and when the full training data is used ($n$=all).

ods: Sherlock (Hulsebos et al., 2019), which uses a multi-input neural network with hand-crafted features extracted from each column, and the aforementioned SATO (Zhang et al., 2019), which improves Sherlock by incorporating table context, topic model outputs, and label co-occurrence information. Table 4 shows the support-weighted F1-score for each method. Similar to the previous two tasks, TABBIE and TaBERT significantly outperform the prior state-of-the-art (SATO). Here, there are no clear differences between the two models, but both reach higher F1 scores than the other baselines even when given only 1,000 training examples, which demonstrates the power of table-based pretraining.

## 4 Analysis

The results in the previous section show that TABBIE is a powerful table representation method, outperforming TaBERT in many downstream task configurations and remaining competitive in the rest. In this section, we dig deeper into TABBIE's representations by comparing them to TaBERT across a variety of quantitative and qualitative analysis tasks, including our own pretraining task of corrupt cell classification, as well as embedding clustering and nearest neighbors. Taken as a whole, the analysis suggests that TABBIE is able to better capture fine-grained table semantics.

### 4.1 Corrupt Cell Detection

We first examine how TaBERT performs on TABBIE's pretraining task of corrupt cell detection, which again is practically useful as a post-processing step after table structure decomposition (Tensmeyer et al., 2019; Raja et al., 2020) because mistakes in predicting row/column/cell boundaries (sometimes compounded by OCR errors) can lead to inaccurate extraction. We fine-tune TaBERT on 100K tables using the **MIX** corruption strategy for ten epochs, and construct a test set of 10K tables that are unseen by both TaBERT and TABBIE during pretraining. While TABBIE of course sees an order of magnitude more tables for this task during pretraining, this is still a useful experiment to determine if TaBERT's pretraining objective enables it to easily detect corrupted cells.

| Corruption | Method | Prec. | Rec. | F1 |
|---|---|---|---|---|
| *Intra-row swap* | TaBERT | 85.5 | 83.0 | 84.2 |
| | TABBIE (FREQ) | 99.0 | 81.4 | 89.4 |
| | TABBIE (MIX) | 99.6 | 95.8 | **97.7** |
| *Intra-column swap* | TaBERT | 31.2 | 19.0 | 23.7 |
| | TABBIE (FREQ) | 90.9 | 22.3 | 35.8 |
| | TABBIE (MIX) | 91.5 | 55.0 | **68.8** |
| *Intra-table swap* | TaBERT | 81.2 | 69.5 | 74.9 |
| | TABBIE (FREQ) | 98.2 | 73.3 | 84.0 |
| | TABBIE (MIX) | 98.4 | 86.2 | **91.9** |
| *Random FREQ cell* | TaBERT | 86.7 | 87.0 | 86.8 |
| | TABBIE (FREQ) | 99.3 | 98.2 | **98.8** |
| | TABBIE (MIX) | 99.1 | 98.1 | 98.6 |
| *All* | TaBERT | 75.6 | 65.2 | 70.0 |
| | TABBIE (FREQ) | 98.2 | 69.5 | 81.4 |
| | TABBIE (MIX) | 97.8 | 84.1 | **90.5** |

Table 5: A fine-grained comparison of different models on corrupt cell detection, with different types of corruption. TaBERT struggles on this task, especially in the challenging setting of *intra-column swaps*. Unlike our downstream tasks, the **MIX** strategy is far superior to **FREQ** here.

As shown in Table 5, TaBERT performs significantly worse than TABBIE on all types of corrupt cells (both random corruption and intra-table swaps). Additionally, intra-column swaps are the most difficult for both models, as TABBIE achieves a **68.8** F1 on this subset compared to just **23.7** F1 by TaBERT. Interestingly, while the **MIX** strategy consistently performs worse than **FREQ** for the TABBIE models evaluated on the three downstream tasks in the previous section, it is substantially better at detecting more challenging corruptions, and is almost equivalent to detecting random cells sampled by **FREQ**. This result indicates that perhaps more complex table-based tasks are required to take advantage of representations derived using **MIX** corruption.

### 4.2 Nearest neighbors

We now turn to a qualitative analysis of the representations learned by TABBIE. In Figure 6 (top), we display the two nearest neighbor columns from our validation set to the *date* column marked by the red box. TABBIE is able to model the similarity of *feb.*

(a) Input table

| # | name | year |
|---|---|---|
| 15 | allysia | junior |
| 18 | maria | senior |
| 17 | emily | sophomore |
| 16 | hydn | sophomore |
| 19 | hayley | sophomore |
| 20 | michelle | graduate |

(b) TABBIE (MIX)

| # | name | year |
|---|---|---|
| 0.0% | 0.1% | 0.0% |
| 100% | 0.0% | 0.0% |
| 0.0% | 0.0% | 0.0% |
| 99.9% | 0.0% | 0.0% |
| 0.0% | 0.0% | 0.0% |
| 0.0% | 0.0% | 0.0% |

(c) TaBERT

| # | name | year |
|---|---|---|
| 2.6% | 1.6% | 8.9% |
| 3.2% | 2.6% | 1.9% |
| 4.3% | 7.6% | 5.2% |
| 2.2% | 0.3% | 0.5% |
| 3.3% | 3.3% | 1.5% |
| 4.0% | 6.6% | 2.9% |

Figure 5: In this figure, (b) and (c) contain the predicted corruption probability of each cell in (a). Only TABBIE **MIX** is able to reliably identify violations of numerical trends in columns.

*16* and *saturday. february 5th* despite the formatting difference, while TaBERT's neighbors more closely resemble the original column. Figure 6 (bottom) shows that TABBIE's nearest neighbors are less reliant on matching headers than TaBERT, as the neighbors all have different headers (*nom*, *nombre*, *name*).

### 4.3 Clustering

Are the embeddings produced by TABBIE useful for clustering and data discovery? To find out, we perform clustering experiments on the FinTabNet dataset from Zheng et al. (2021). This dataset contains ∼110K tables from financial reports of corporations in the S&P-500. We use the [CLS] embedding at the $(0, 0)$ position (i.e., the top left-most cell in the table), extracted from a TABBIE model trained with the **FREQ** strategy, as a representative embedding for each table in the dataset. Next, we perform $k$-means clustering on these embeddings using the FAISS library (Johnson et al., 2017), with $k$=1024 centroids. While the FinTabNet dataset is restricted to the homogenous domain of financial tables, these tables cluster into sub-types such as *consolidated financial tables*, *jurisdiction tables*, *insurance tables*, etc. We then examine the contents of these clusters (Figure 7) and observe that TABBIE embeddings can not only be clustered into these sub-types, but also that tables from reports of the same company, but from different financial years, are placed into the same cluster.

### 4.4 Identifying numeric trends

Next, we analyze how well TABBIE understands trends in numerical columns by looking at specific examples of our corrupt cell detection task. The first column of the table in Figure 5 contains jersey numbers sorted in ascending order. We swap two cells in this column, *16* and *18*, which violates

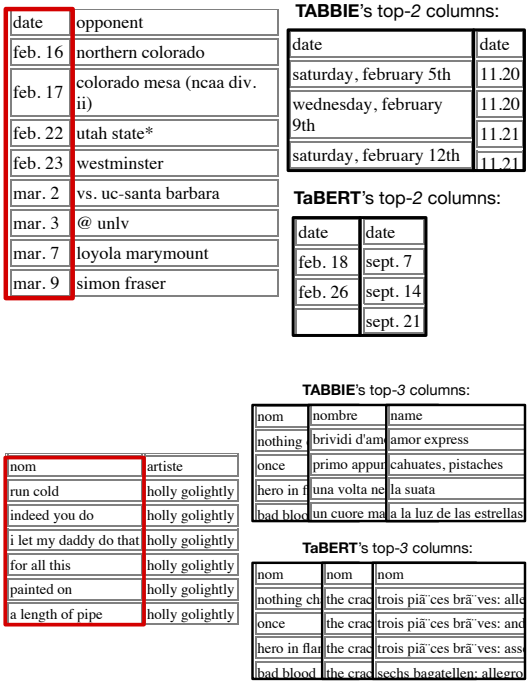



Figure 6: Nearest neighbors of the *date* and *nom* columns from the tables on the left, from both TABBIE and TaBERT. TABBIE's nearest neighbors exhibit more diverse formatting and less reliance on the header, which is an example of its semantic representation capability.

the increasing trend. Both TaBERT (fine-tuned for corrupt cell detection) and TABBIE **FREQ** struggle to identify this swap, while TABBIE **MIX** is almost certain that the two cells have been corrupted. This qualitative result is further evidence that the **MIX** model has potential for more complex table-based reasoning tasks.

## 5 Related work

The staggering amount of structured relational data in the form of tables on the Internet has attracted considerable attention from researchers over the past two decades (Cafarella et al., 2008; Limaye et al., 2010; Venetis et al., 2011; Suchanek et al., 2007; Embley et al., 2006), with applications including retrieval (Das Sarma et al., 2012), schema-matching (Madhavan et al., 2001, 2005), and entity linking (Zhang et al., 2020).

Similar to popular large-scale language models pretrained on tasks involving unstructured natural language(Peters et al., 2018; Devlin et al., 2018; Liu et al., 2019), our work is part of a recent trend of self-supervised models trained on structured tabular data. TaBERT (Yin et al., 2020) and TaPaS (Herzig et al., 2020) jointly model tables

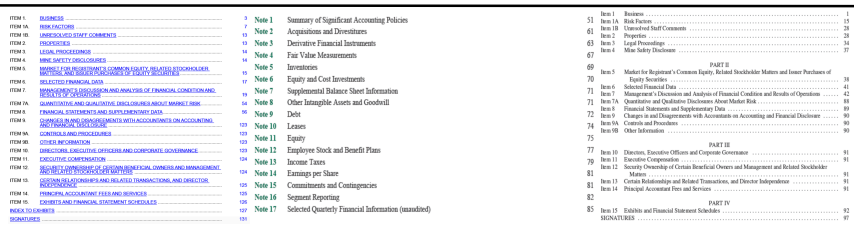
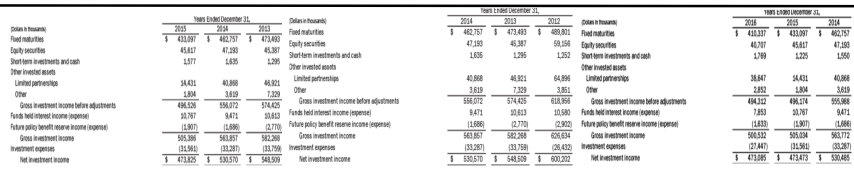
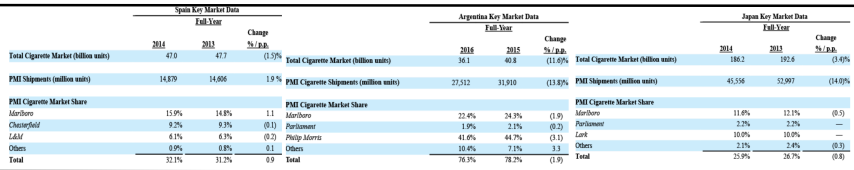
| Semantic type | Sample Tables | Centroid No. |
|---|---|---|
| Table of contents |  | 23 |
| Investment income table for *Everest Re Group* |  | 190 |
| Market share table for *Phillip Morris International* |  | 295 |

Figure 7: Sample tables from clusters obtained by running $k$-means on TABBIE's [CLS] embeddings on the FinTabNet dataset. TABBIE not only clusters embeddings into reasonable semantic types, such as *Table of Contents* (first row), but it also places tables of the same type from the same company into the same cluster (second and third rows). We provide the source images of the corresponding tables in this figure.

with text (typically captions or questions), and are thus more suited for tasks like question answering (Pasupat and Liang, 2015). For pretraining, TaBERT attempts to recover the name and datatype of masked column headers (masked column prediction), in addition to contents of a particular cell (cell value recovery). The pretraining objectives of TaPaS, on the other hand, encourage tabular textual entailment. In a concurrent work, the TUTA model (Wang et al., 2020) uses masked language modeling, cell-level cloze prediction, and table-context retrieval as pretraining objectives. Further, in addition to traditional position embeddings, this work accounts for the hierarchical nature of tabular data using tree-based positional embeddings. Similiarly, in Deng et al. (2020), the authors perform a variant of MLM called masked entity recovery. In contrast, TABBIE is pretrained strictly on tabular data and intended for more general-purpose table-based tasks, and uses corrupt-cell classification as its pretraining task.

## 6 Conclusion

In this paper, we proposed TABBIE, a self-supervised pretraining method for tables without associated text. To reduce the computational cost of training our model, we repurpose the ELECTRA objective for corrupt cell detection, and we use two separate Transformers for rows and columns to minimize complexity associated with sequence length. On three downstream table-based tasks, TABBIE achieves competitive or better performance to existing methods such as TaBERT, and an analysis reveals that its representations include a deep semantic understanding of cells, rows, and columns. We publicly release our TABBIE pretrained models and code to facilitate future research on tabular representation learning.

## 7 Ethics Statement

As with any research work that involves training large language models, we acknowledge that our work has a negative carbon impact on the environment. A cumulative of 1344 GPU-hours of computation was performed on Tesla V100 GPUs. Total emissions are estimated to be 149.19 kg of $CO_2$ per run of our model (in total, there were two runs). While this is a significant amount (equivalent to $\approx$ 17 gallons of fuel consumed by an average motor vehicle[11]), it is lower than TaBERT's cost per run by more than a factor of 10 assuming a similar computing platform was used. Estimations were conducted using the Machine Learning Impact calculator presented in Lacoste et al. (2019).

[11] https://www.epa.gov/greenvehicles/

## Acknowledgements

We thank the anonymous reviewers for their useful comments. We thank Christopher Tensmeyer for helpful comments and pointing us to relevant datasets for some of our experiments. We also thank the UMass NLP group for feedback during the paper writing process. This work was made possible by research awards from Sony Corp. and Adobe Inc. MI is also partially supported by award IIS-1955567 from the National Science Foundation (NSF).

## References

Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 817–828, New York, NY, USA. Association for Computing Machinery.

Li Deng, Shuo Zhang, and Krisztian Balog. 2019. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of SIGIR 2019*.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: Table understanding through representation learning. *Proc. VLDB Endow.*, 14(3):307–319.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

D. Embley, Matthew Hurst, D. Lopresti, and G. Nagy. 2006. Table-processing paradigms: a research survey. *International Journal of Document Analysis and Recognition (IJDAR)*, 8:66–86.

Jonathan Herzig, P. Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *ACL*.

Kevin Zeng Hu, Snehalkumar (Neil) S. Gaikwad, Madelon Hulsebos, Michiel A. Bakker, Emanuel Zgraggen, César A. Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çagatay Demiralp. 2019. Viznet: Towards A large-scale visualization learning and benchmarking repository. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*.

M. Hulsebos, K. Hu, M. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, T. Kraska, cCaugatay Demiralp, and C'esar A. Hidalgo. 2019. Sherlock: A deep learning approach to semantic data type detection. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*.

Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1):1338–1347.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Halevy. 2005. Corpus-based schema matching. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, page 57–68, USA. IEEE Computer Society.

Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. 2001. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, page 49–58, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

D. Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Y. Li, Ashwin Bharambe, and L. V. D. Maaten. 2018. Exploring the limits of weakly supervised pretraining. In *ECCV*.

Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Erhard Rahm and Philip A. Bernstein. 2001. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350.

Sachin Raja, Ajoy Mondal, and C. V. Jawahar. 2020. Table structure recognition using top-down and bottom-up cues. In *Computer Vision – ECCV 2020*, pages 70–86, Cham. Springer International Publishing.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*.

C. Tensmeyer, V. I. Morariu, B. Price, S. Cohen, and T. Martinez. 2019. Deep splitting and merging for table structure decomposition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 114–121.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ¥L ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 4(9):528–538.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Empirical Methods in Natural Language Processing*.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2020. Structure-aware pre-training for table understanding with tree-based transformers. *ArXiv*, abs/2010.12537.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. 2019. Sato: Contextual semantic type detection in tables.

Shuo Zhang and Krisztian Balog. 2017. Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Trans. Intell. Syst. Technol.*

Shuo Zhang, Edgar Meij, Krisztian Balog, and Ridho Reinanda. 2020. Novel entity discovery from web tables. In *Proceedings of The Web Conference 2020*, WWW '20, page 1298–1308, New York, NY, USA. Association for Computing Machinery.

Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. 2021. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 697–706.