Annotating Columns with Pre-trained Language Models

Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang
Megagon Labs
{yoshi,jinfeng,yuliang,dan_z}@megagon.ai

Çağatay Demiralp*
Sigma Computing
cagatay@sigmacomputing.com

Chen Chen†
Megagon Labs
chen@megagon.ai

Wang-Chiew Tan*
Meta AI
wangchiew@fb.com

ABSTRACT

Inferring meta information about tables, such as column headers or relationships between columns, is an active research topic in data management as we find many tables are missing some of this information. In this paper, we study the problem of annotating table columns (i.e., predicting column types and the relationships between columns) using only information from the table itself. We develop a multi-task learning framework (called Doduo) based on pre-trained language models, which takes the entire table as input and predicts column types/relations using a single model. Experimental results show that Doduo establishes new state-of-the-art performance on two benchmarks for the column type prediction and column relation prediction tasks with up to 4.0% and 11.9% improvements, respectively. We report that Doduo can already outperform the previous state-of-the-art performance with a minimal number of tokens, only 8 tokens per column. We release a toolbox[1] and confirm the effectiveness of Doduo on a real-world data science problem through a case study.

CCS CONCEPTS

• **Information systems** → **Information integration**.

KEYWORDS

table understanding, language models, multi-task learning

ACM Reference Format:
Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating Columns with Pre-trained Language Models. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3514221.3517906

*Work done while the author was at Megagon Labs.
†Deceased.
[1]https://github.com/megagonlabs/doduo

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3517906

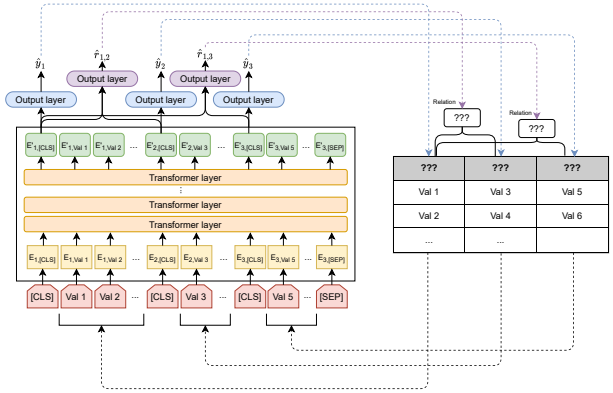


Figure 1: Overview of Doduo's model architecture. Doduo serializes the entire table into a sequence of tokens to make it compatible with the Transformer-based architecture. To handle the column type prediction and column relation extraction tasks, Doduo implements two different output layers on top of column representations and a pair of column representations, respectively.

1 INTRODUCTION

Meta information about tables, such as column types and relationships between columns (or column relations), is crucial to a variety of data management tasks, including data quality control [40], schema matching [37], and data discovery [7]. Recently, there is an increasing interest in identifying *semantic* column types and relations [20, 21, 60]. Semantic column types such as "population", "city", and "birth_date" provide contain finer-grained, richer information than standard DB types such as integer or string. Similarly, semantic column relations such as a binary relation "is_birthplace_of" connecting a "name" and a "city" column can provide valuable information for understanding semantics of the table. For example, commercial systems (e.g., Google Data Studio [17] , Tableau [41]) leverage such meta information for better table understanding. However, semantic column types and relations are typically missing in tables while annotating such meta information manually can be quite expensive. Thus, it is essential to build models that can automatically assign meta information to tables.

Figure 2 shows two tables with missing column types and column relations. The table in Figure 2(a) is about animation films and the corresponding directors/producers/release countries of the

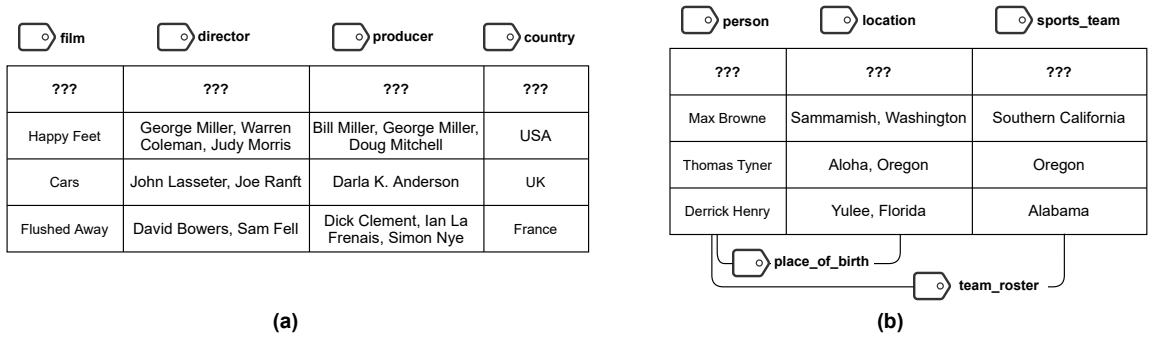


Figure 2: Two example tables from the WikiTable dataset. (a) The task is to predict the column type of each column based on the table values. (b) The task is to predict both column types and relationships between columns. The column types (the column relations) are depicted at the top (at the bottom) of the table. This example also shows that column types and column relations are inter-dependent and hence, our motivation to develop a unified model for predicting both tasks.

films. In the second and third columns, person names will require context, both in the same column and the other columns, to determine the correct column types. For example, `George Miller`[2] appears in both columns as a director and a producer, and it is also a common name. Observing other names in the column helps better understand the semantics of the column. Furthermore, a column type is sometimes dependent on other columns of the table. Hence, by taking contextual information into account, the model can learn that the topic of the table is about (animation) films and understand that the second and third columns are less likely to be `politician` or `athlete`. To sum up, this example shows that the table context and both intra-column and inter-column context can be very useful for column type prediction.

Figure 2(b) depicts a table with predicted column types and column relations. The column types `person` and `location` are helpful for predicting the relation `place_of_birth`. However, it will still need further information to distinguish whether the location is `place_of_birth` or `place_of_death`.

The example above shows that column type and column relation prediction tasks are intrinsically related. Thus it will be synergistic to solve the two tasks simultaneously using a single framework. To combine the synergies of column type prediction and column relation prediction tasks, we develop Doduo that: (1) learns column representations, (2) incorporates table context, and (3) uniformly handles both column annotation tasks. Most importantly, our solution (4) shares knowledge between the two tasks.

Doduo leverages a pre-trained Transformer-based language models (LMs) and adopts *multi-task learning* into the model to appropriately "transfer" shared knowledge from/to the column type/relation prediction task. The use of the pre-trained Transformer-based LM makes Doduo a data-driven representation learning system[3] (i.e., feature engineering and/or external knowledge bases are not needed) (Challenge 1.) Pre-trained LM's contextualized representations and our table-wise serialization enable Doduo to naturally incorporate table context into the prediction (Challenge 2) and to handle different tasks using a single model (Challenge 3.) Lastly, training such a *table-wise* model via multi-task learning helps "transfer" shared knowledge from/to different tasks (Challenge 4.)

Figure 1 depicts the model architecture of Doduo. Doduo takes as input values from multiple columns of a table after serialization and predicts column types and column relations as output. Doduo considers the table context by taking the serialized column values of all columns in the same table. This way, both *intra-column* (i.e., co-occurrence of tokens within the same column) and *inter-column* (i.e., co-occurrence of tokens in different columns) information is accounted for. Doduo appends a dummy symbol `[CLS]` at the beginning of each column and uses the corresponding embeddings as *learned column representations* for the column. The output layer on top of a column embedding (i.e., `[CLS]`) is used for column type prediction, whereas the output layer for the column relation prediction takes the column embeddings of each column pair.

Contributions Our contributions are:

- We develop Doduo, a unified framework for both column type prediction and column relation prediction. Doduo incorporates table context through the Transformer architecture and is trained via multi-task learning.
- Our experimental results show that Doduo establishes new state-of-the-art performance on two benchmarks, namely the WikiTable and VizNet datasets, with up to 4.0% and 11.9% improvements compared to TURL and Sato.
- We show that Doduo is data-efficient as it requires less training data or less input data. Doduo achieves competitive performance against previous state-of-the-art methods using less than half of the training data or only using 8 tokens per column as input.
- We release the codebase and models as a toolbox, which can be used with just a few lines of Python code. We test the performance of the toolbox on a real-world data science problem and verify the effectiveness of Doduo even on out-domain data.

2 RELATED WORK

Existing column type prediction models enjoyed the recent advances in machine learning by formulating column type prediction as a multi-class classification task. Hulsebos et al. [21] developed a deep learning model called Sherlock, which applies neural networks

[2]In this context, `George Miller` refers to an Australian filmmaker, but there exist more than 30 different Wikipedia articles that refer to different George Miller.
[3]In other words, Doduo relies on the general knowledge obtained from text corpora (e.g., Wikipedia) and a training set of tables annotated with column types and relations.

on multiple feature sets such as word embeddings, character embeddings, and global statistics extracted from individual column values. Zhang et al. [60] developed Sato, which extends Sherlock by incorporating table context and structured output prediction to better model the nature of the correlation between columns in the same table. Other models such as ColNet [8], HNN [9], Meimei [44], C^2 [23] use external Knowledge Bases (KBs) on top of machine learning models to improve column type prediction. Those techniques have shown success on column type prediction tasks, outperforming classical machine learning models.

While those techniques identify the semantic types of individual columns, another line of work focuses on *column relations* between pairs of columns in the same table for better understanding tables [3, 12, 26, 27, 30, 49]. A column relation is a semantic label between a pair of columns in a table, which offers more fine-grained information about the table. For example, a relation `place_of_birth` can be assigned to a pair of columns `person` and `location` to describe the relationship between them. Venetis et al. [49] use an Open IE tool [57] to extract triples to find relations between entities in the target columns. Muñoz et al. [30] use machine learning models to filter triple candidates created from DBPedia. Cannaviccio et al. [3] use a language model-based ranking method [59], which is trained on a large-scale web corpus, to re-rank relations extracted by an open relation extraction tool [31]. Cappuzo et al. [4] represent table structure as a graph and then learn the embeddings from the descriptive summaries generated from the graph.

Recently, pre-trained Transformer-based Language Models (LMs) such as BERT, which were originally designed for NLP tasks, have shown success in data management tasks. Li et al. [25] show that pre-trained LMs is a powerful base model for entity matching. Macdonald et al. [27] proposed applications for entity relation detection. Tang et al. [45] propose RPTs as a general framework for automating human-easy data preparation tasks like data cleaning, entity resolution and information extraction using pre-trained masked language models. The power of Transformer-based pre-trained LMs can be summarized into two folds. First, using a stack of Transformer blocks (i.e., self-attention layers), the model is able to generate contextualized embeddings for structured data components like table cells, columns, or rows. Second, models pre-trained on large-scale textual corpora can store "semantic knowledge" from the training text in the form of model parameters. For example, BERT might know that George Miller is a director/producer since the name frequently appears together with "directed/produced by" in the text corpus used for pre-training. In fact, recent studies have shown that pre-trained LMs store a significant amount of factual knowledge, which can be retrieved by template-based queries [22, 36, 38].

Those pre-trained models have also shown success in data management tasks on tables. TURL [12] is a Transformer-based pre-training framework for table understanding tasks. Contextualized representations for tables are learned in an unsupervised way during pre-training and later applied to 6 different tasks in the fine-tuning phase. SeLaB [47] leverages pre-trained LMs for column annotation while incorporating table context. Their approach uses fine-tuned BERT models in a two-stage manner. TaPaS[19] conducts weakly supervised parsing via pre-training, and TaBERT[58] pre-trains for a joint understanding of textual and tabular data for the text-to-SQL task. TUTA [53] makes use of different pre-training objectives to obtain representations at token, cell, and table levels and propose a tree-based structure to describe spatial and hierarchical information in tables. TCN [52] makes use of both information within the table and across external tables from similar domains to predict column type and pairwise column relations.

Table 1: Notations.

Symbol	Description
$T = (c_1, c_2, \ldots, c_n)$	Columns in a table.
$c_i = (v_1^i, v_2^i, \ldots, v_m^i)$	Column values.
$v_j^i = (w_{j,1}^i, w_{j,2}^i, \ldots, w_{j,K}^i)$	A single column value.
$D_{\text{train}} = \left\{T^{(n)}, L_{\text{type}}^{(n)}, L_{\text{rel}}^{(n)}\right\}_{n=1}^{N}$	Training data
$L_{\text{type}} = (l_1, l_2, \ldots, l_n), l_* \in C_{\text{type}}$	Column type labels.
$L_{\text{rel}} = (l_{1,2}, l_{1,3}, \ldots, l_{1,n}), l_{*,*} \in C_{\text{rel}}$	Column relation labels.

In this paper, we empirically compare Doduo with Sherlock [21], Sato [60], and TURL [12] as baseline methods. Sherlock is a single-column model while Doduo is multi-column by leveraging table context to predict column types and relations more accurately. Sato leverages topic model (LDA) features as table context while Doduo can additionally take into account fine-grained, token-level interactions among columns via its built-in self-attention mechanism. TURL is also a Transformer-based model like Doduo but it requires additional meta table information such as table headers for pre-training. Doduo is more generic as it predicts column types and relations only relying on cell values in the table. See Section 5 for a more detailed comparison.

3 BACKGROUND

In this section, we formally define the two column annotation tasks: column type prediction and column relation annotation. We also provide a brief background on pre-trained language models (LMs) and how to fine-tune them for performing column annotations.

3.1 Problem Formulation

The goal of the column type prediction task is to classify each column to its *semantic type*, such as "country name", "population", and "birthday" instead of the standard column types such as `string`, `int`, or `Datetime`. See also Figure 2 for more examples. For column relation annotation, our goal is to classify the relation of each pair of columns. In Figure 2, the relation between the "person" column and the "location" column can be "`place_of_birth`".

As summarized in Table 1, more formally, we consider a standard relational data model where a relation T (i.e., table) consists of a set of attributes $T = (c_1, \ldots c_n)$ (i.e., columns.) We denote by $\text{val}(T.c_i)$ the sequence of data values stored at the column c_i. For each value $v \in \text{val}(T.c_i)$, we assume v to be of the `string` type and can be split into a sequence of input tokens $v = [w_1, \ldots, w_k]$ to pre-trained LMs. This approach of casting cell values into text might seem restricted since tables columns can be of numeric types such as `float` or `date`. There has been extensions of the Transformer models to support numeric data [55] and providing such direct support of numeric data is important future work. We also provide a brief analysis on Doduo's performance on numeric column types in Section 5.4.

Problem 1 (Column type prediction). *Given a table T and a column c_i in T, a column type prediction model $\mathcal{M}$ with type vocabulary C_{type} predicts a column type $\mathcal{M}(T, c_i) \in C_{\text{type}}$ that best describes the semantics of c_i.*

Problem 2 (Column relation prediction). *Given a table T and a pair of columns (c_i, c_j) in T, a column relation prediction model $\mathcal{M}$ with relation vocabulary C_{rel} predicts a relation $\mathcal{M}(T, c_i, c_j) \in C_{\text{rel}}$ that best describes the semantics of the relation between c_i and c_j.*

In Doduo, we consider the supervised setting of multi-class classification. This means that we assume a training data set D_{train} of tables annotated with columns types and relations from two fixed vocabularies $(C_{\text{type}}, C_{\text{rel}})$. Note that Doduo does not restrict itself to specific choices of vocabularies $(C_{\text{type}}, C_{\text{rel}})$ which are customizable by switching the training set D_{train}. In practice, the choice of $(C_{\text{type}}, C_{\text{rel}})$ is ideally application-dependent. For example, if the downstream task requires integration with a Knowledge Base (KB), it is ideal to have $(C_{\text{type}}, C_{\text{rel}})$ aligned with the KB's type/relation vocabulary. In our experiment, we evaluated Doduo on datasets annotated with (1) KB types [2] and (2) DBPedia types [32].

The size and quality of the training set are also important for training high-quality column annotation models. While manually creating such datasets can be quite expensive, the datasets used in our experiments rely on heuristics that map table meta-data (e.g., header names, entity links) to type names to create training sets of large scale. See Section 5.1 for more details.

While KB can work as a training example provider, Doduo does not require the training examples to be from a single source but can combine labels from any resources such as human annotations, labeling rules, and meta-data that can be transformed into the column type/relation label format.

We also note that the learning goal of Doduo is to train column annotation models with high accuracy while being generalizable to unannotated tables (e.g., as measured by an unseen test set D_{test}). The column type/relation prediction models of Doduo *only* considers the table content (i.e., cell values) as input. This setting allows Doduo to be more flexible to practical applications without replying on auxiliary information such as column names, table titles/captions, or adjacent tables typically required by existing works (See Section 2 for a comprehensive overview).

3.2 Pre-trained Language Models

Pre-trained Language Models (LMs) emerge as general-purpose solutions to tackle various natural language processing (NLP) tasks. Representative LMs such as BERT [13] and ERNIE [43] have shown leading performance among all solutions in NLP benchmarks such as GLUE [16, 51]. These models are pre-trained on large text corpora such as Wikipedia pages and typically employ multi-layer Transformer blocks [48] to assign more weights to informative words and less weight to stop words for processing raw texts. During pre-training, a model is trained on self-supervised language prediction tasks such as missing token prediction and next-sentence prediction. The purpose is to learn the semantic correlation of word tokens (e.g., synonyms), such that correlated tokens can be projected to similar vector representations. After pre-training, the model is able to learn the lexical meaning of the input sequence in the shallow layers and the syntactic and semantic meanings in the deeper layers [10, 46].

A special component of pre-trained LMs is the attention mechanism, which embeds a word into a vector based on its context (i.e., surrounding words). The same word has different vectors if it appears in different sentences, and this is very different from other embedding mechanisms such as word2vec [28], GloVe [35], and fastText [1], which always generate the same vector for the same word in any context. Pre-trained LMs' embeddings are context-dependent and thus offer two strengths. First, it can discern polysemy. For example, the person name `George Miller` referring to a producer is different from the same name that refers to a director. Pre-trained LMs discern the difference and generate different vectors. Second, the embedding deals with synonyms well. For example, the words `Derrick Henry` and `Derrick Lamar Henry Jr` (respectively, (`USA`, `US`), (`Oregon`, `OR`)) are likely the same given their respective contexts. Pre-trained LMs will generate similar word vectors accordingly. Due to the two favorable strengths, pre-trained models should enable the best performance to column annotation tasks, where each cell value is succinct, and its meaning highly depends on its surrounding cells.

The pre-trained model does not know what to predict for specific tasks unless the task is exactly the same the pre-training task. Thus, a pre-trained LM needs to be *fine-tuned* with task-specific training data, so the model can be tailored for the task. A task-specific output layer is attached to the final layer of the pre-trained LM, and the loss value (e.g., cross-entropy loss) is back-propagated from the output layer to the pre-trained LM for a minor adjustment.

In this paper, we fine-tune the popular BERT Base model [13]. However, Doduo is flexible with the choice of pre-trained LMs and can potentially perform even better with larger pre-trained LMs.

3.3 Multi-task learning

Multi-task learning [6] is a type of supervised machine learning framework, where the objective function is calculated based on more than one task. Generally, different types of labels are used, and the labels may be or may not be annotated on the same example. The intuition and an assumption behind multi-task learning is that the tasks intrinsically share knowledge, and thus, training with the same base model benefits each other.

The major benefit of multi-task learning is that it can help improve the generalization performance of the model, especially when the training data is not sufficient. Multi-task learning can be easily applied to Deep Learning models [39] by attaching different output layers to the main model, which is considered a "learned" representation encoder that converts input data to dense representations.

There are a variety of approaches for multi-task learning [39], depending on how to model and optimize shared parameters. Multi-task learning models can be split into two categories based on how parameters are shared. With *hard parameter sharing* [5], models for multiple tasks share the same parameters, whereas *soft parameter sharing* [56] adds constraints on distinct models for different tasks. In this paper, we consider hard parameter sharing as it is a more cost-effective approach. Among hard parameter sharing models, we choose a joint multi-task learning framework [18] that uses the same base model with different output layers for different tasks.

4 MODEL

In this section, we first introduce a baseline single-column model that fine-tunes a pre-trained LM on individual columns. Then, we describe Doduo's model architecture and training procedure.

4.1 Single-column Model

Since LMs take token sequences (i.e., text) as input, one first has to convert a table into token sequences so that they can be meaningfully processed by LMs. A straightforward serialization strategy is to simply concatenate column values to make a sequence of tokens and feed that sequence as input to the model. That is, suppose a column C has column values $v_1, \dots v_m$, the serialized sequence is

$$\text{serialize}_{\text{single}}(C) ::= \text{[CLS]}\ v_1\ \dots\ v_m\text{[SEP]},$$

where [CLS] and [SEP] are special tokens used to mark the beginning and end of a sequence[4]. For example, the first column of the first table in Figure 2 is serialized as: [CLS] Happy Feet Cars Flushed Away [SEP]. This serialization converts the problem into a sequence classification task. Thus, it is straightforward to fine-tune a BERT model using training data.

The column relation prediction task can be also formulated as a sequence classification task by converting a pair of columns (instead of a single column) into a token sequence in a similar manner. For this case, we also insert additional [SEP] between values of two columns to help the pre-trained LM distinguish the two columns. Namely, given two columns $C = v_1, \dots, v_m$ and $C' = v'_1, \dots, v'_m$, the single-column model serializes the pair as:

$$\text{serialize}_{\text{single}}(C, C') ::= \text{[CLS]}\ v_1\ \dots\ v_m\text{[SEP]}\ v'_1\ \dots\ v'_m\text{[SEP]}.$$

Using the above serialization scheme, we can cast the column type and relation prediction tasks as sequence classification and sequence-pair classification tasks, which can be solved by LM fine-tuning. However, such sequence classifications predict column types independently, even if they are in the same table. We refer to this method as the *single-column* model. Although the single-column model can leverage the language understanding capability learned by the pre-trained LM, it has an obvious drawback of treating columns in the same table as independent sequences. As a result, the single-column model fails to capture the table context, which is known to be essential for column annotation tasks [9, 23, 60].

4.2 Table Serialization

In contrast to the single-column model described above, Doduo is a *multi-column* (or table-wise) model that takes an entire table as input. Doduo serializes data entries as follows: for each table that has n columns $T = (c_i)_{i=1}^{n}$, where each column has N_m column values $c_i = (v_j^i)_{j=1}^{m}$. We let

$$\text{serialize}(T) ::= \text{[CLS]}\ v_1^1\ \dots \text{[CLS]}\ v_1^n\ \dots v_m^n\text{[SEP]}.$$

For example, the first table in Figure 2 is serialized as:

[CLS] Happy Feet, …[CLS] George Miller, …[CLS] USA, …, France [SEP].

Different from the single-column model, which always has a single [CLS] token in the input, Doduo's serialization method inserts as many [CLS] tokens as the number of columns in the input table. This difference makes a change in the classification formulation. While the single-column model classifies a single sequence (i.e., a single column) by predicting a single label, Doduo predicts as many labels as the number of [CLS] tokens in the input sequence. The learning procedure of Doduo starts by serializing and tokenizing all tables in the datasets (Line 3-4 of Algorithm 1.)

[4]Note that [CLS] and [SEP] are the special tokens for BERT and other LMs may have other special tokens, which are usually implemented as part of their tokenizers.

Algorithm 1: Training procedure of Doduo

Input: Number of training epochs N_{Epoch}; For each task $i \in [1, T]$, training set D_i, loss function $\mathcal{L}_i$, and optimizer O_i
Output: Annotation model $\mathcal{M}$ = (LM, $\{g_1, \dots, g_T\}$) with a language model LM and T heads g_1 to g_T

```
// Initialize model weights
1  Initialize LM using its pre-trained weights;
2  Initialize {g_1, ..., g_T} randomly;
   // Serialize all tables in each training set
3  for i = 1 to T do
4      D_i ← {serialize(T) for T ∈ D_i};
5  for ep = 1 to N_Epoch do
6      for i = 1 to T do
7          Randomly split D_i into mini-batches {B_1, ... B_k};
8          for B in {B_1, ... B_k} do
               // Evaluate the L_i loss func on the g_i head
9              L ← L_i(B, LM ⊕ g_i);
               // Back-propagate to update model weights
10             M ← back-propagate(M, O_i, ∂L/∂(LM ⊕ g_i));
11 return M;
```

4.3 Contextualized Column Representations

We describe how Doduo obtains table context through contextualized column embeddings using the Transformer architecture. Figure 3 depicts how each Transformer block aggregates contextual information from all column values (including dummy [CLS] symbols and themselves) in the same table. This example illustrates the first Transformer layer calculates the *attention vector* by aggregating embeddings of other tokens based on the similarity against the second column's [CLS] token. Thus, an attention vector for the same symbol (e.g., George) can be different when it appears in a different context. This resolves the ambiguity issue of conventional word embedding techniques such as word2vec or GloVe.

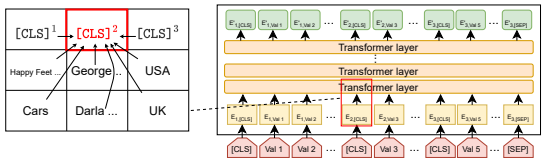


Figure 3: How Doduo computes contextualized column embeddings using the Transformer layers. Each Transformer block calculates an embedding vector for every token based on surrounding tokens.

After encoding tokens into token embeddings, a Transformer layer converts a token embedding into key (K), query (Q), and value (V) embeddings. A contextualized embedding for a token is calculated by the weighted average of value embeddings of all token embeddings, where the weights are calculated by the similarity between the query embedding and key embeddings. By having key embeddings and query embeddings separately, the model is able to calculate contextualized embeddings in an asymmetric manner. That is, the importance of Happy Feet for George Miller, which should be a key signal to disambiguate the person name, may not be necessarily equal to that of George Miller for Happy Feet. Furthermore, a Transformer-based model usually has multiple *attention heads* (e.g., 12 attention heads for the BERT base model.)

Different attention heads have different parameters for K, Q, V calculation so that they can capture different characteristics of input data holistically. Finally, the output of a Transformer block is converted into the same dimension size as that of the input (e.g, 768 for BERT) so that the output of the previous Transformer block can be directly used as the input to the next Transformer block. The same procedure is carried out as many as the number of Transformer blocks (i.e., 12 blocks for the BERT Base model.)

Column representations. Since Doduo inserts dummy `[CLS]` symbols for each column, we can consider the output embeddings of the pre-trained LM for those symbols as *contextualized column representations*. Note that Doduo is a *table-wise* model, which takes the entire table as input and thus contextualized column representations take into account table context in a holistic manner. For column type prediction, Doduo attaches an additional dense layer followed by output layer with the size of $|C_{\text{type}}|$.

Column-pair representations. For column relation prediction, Doduo concatenates a corresponding pair of contextualized column representations as a contextualized column-pair representation. The additional dense layer should capture combinatorial information between two column-level representations. Same as the column representations, column-pair representations are also table-wise representations. In the experiment, we also tested a variant of Doduo that only takes a single column (a single column pair) as input for the column type (column relation) prediction task.

More formally, given a table T, the language model LM takes the serialized sequence $\text{serialize}(T) = \{t_1, \ldots, t_p\}$ of p tokens as input and outputs a sequence $\text{LM}(T)$ where each element $\text{LM}(T)_i$ is a d-dimensional context-aware embedding of the token t_i. Let $\{i_1, \ldots, i_n\}$ be the indices of the inserted special [CLS] tokens. Let g_{type} be the column type prediction dense layer of dimension $d \times |C_{\text{type}}|$. The column type model computes:

$$\text{softmax}(g_{\text{type}}(\text{LM}(T)_{i_j})) \tag{1}$$

as the predicted column type of the j-th column. Similarly, for column relation prediction with dense layer g_{rel} of dimension $2d \times |C_{\text{rel}}|$, the column relation model computes:

$$\text{softmax}(g_{\text{rel}}(\text{LM}(T)_{i_j} \oplus \text{LM}(T)_{i_k})) \tag{2}$$

as the predicted relation between the j-th and the k-th column of table T. The $\oplus$ symbol denotes concatenation of two vectors. Doduo then feeds the predictions and the groundtruth labels into a cross entropy loss function to update the model parameters (Line 9-10, Algorithm 1).

4.4 Learning from Multiple Tasks

In the training phase, Doduo fine-tunes a pre-trained LM using two different training data and two different objectives. As shown in Algorithm 1, Doduo switches the task every epoch and updates the parameters for different objectives using different optimization schedulers. This design choice enables Doduo to naturally handle imbalanced training data for different tasks. Furthermore, with a single objective function and a single optimizer, we need to carefully choose hyper-parameter(s) that balance different objective terms to create a single objective function (e.g., $\ell = \lambda\ell_1 + (1-\lambda)\ell_2$ like TCN [52].) With our strategy, we can avoid adjusting the hyper-parameter. Also, in Section 6, we will show that Doduo can be robustly trained with imbalanced training data.

Table 2: Dataset description.

Name	# tables	# col	# col types	# col rels
WikiTable	580,171	3,230,757	255	121
VizNet	78,733	119,360	78	–

Note that Doduo is not limited to training with just two tasks. By adding more output layers and corresponding loss functions, Doduo can be used for more than two tasks. Finding more relevant tasks and testing Doduo on them are part of our future work.

5 EVALUATION

5.1 Dataset

We used two benchmark datasets for evaluation. The WikiTable dataset [12] is a collection of tables collected from Wikipedia, which consists of 580,171 tables in total. The dataset provides both annotated column types and relations for training and evaluation. For column type prediction, the dataset provides 628,254 columns from 397,098 tables annotated by 255 column types. For column relations, the dataset provides 62,954 column pairs annotated with 121 relation types from 52,943 tables for training. According to [12], the type and relation labels are from FreeBase [2] and are obtained by aggregating entity links attached to the original tables. For both tasks, we used the same train/valid/test splits as TURL [12]. Each column/column-pair allows to have more than one annotation, and thus, the task is a multi-label classification task.

The VizNet dataset [60] is a collection of WebTables, which is a subset of the original VizNet corpus [20]. The dataset is for the column type prediction task. The dataset has 78,733 tables, and 119,360 columns are annotated with 78 column types. The dataset constructed the columns types by mapping column headers to DBpedia types [32] by a set of mapping rules. We used the same splits for the cross-validation to make the evaluation results directly comparable to [60]. Each column has only one label, and thus, the task is a multi-class classification task.

5.2 Baselines

TURL [12] is a recently developed pre-trained Transformer-based LM for tables. TURL further pre-trains a pre-trained LM using table data, so the model becomes more suitable for tabular data. Since TURL relies on entity-linking and meta information such as table headers and table captions, which are not available in our scenario, we used a variant of TURL pre-trained on table values for a fair comparison. Note that to perform column type/relation annotation, we fine-tuned the pre-trained TURL model on the same training sets as for Doduo and other baselines.

Sherlock [21] is a single-column prediction model that uses multiple feature sets, including character embeddings, word embeddings, paragraph embeddings, and column statistics (e.g., mean, std of numerical values.) A multi-layer "sub" neural network is applied to each column-wise feature set to calculate compact dense vectors except for the column statistics feature set, which are already continuous values. The output of the subnetworks and the column statistics features are fed into the "primary" neural network that consists of two fully connected layers.

Sato [60] is a multi-column prediction model, which extends Sherlock by adding LDA features to capture table context and a CRF

Table 3: Performance on the WikiTable dataset.

Method	Col type			Col rel		
	P	R	F1	P	R	F1
Sherlock	88.40	70.55	78.47	–	–	–
TURL	90.54	87.23	88.86	91.18	90.69	90.94
Doduo	**92.69**	**92.21**	**92.45**	**91.97**	**91.47**	**91.72**
TURL+metadata	92.75	92.63	92.69	92.90	93.80	93.35
Doduo+metadata	93.25	92.34	92.79	91.20	94.50	92.82

layer to incorporate column type dependency into prediction. Sato is the state-of-the-art column type prediction on the VizNet dataset.

5.3 Experimental Settings

We used Adam optimizer with an ϵ of 10^{-8}. The initial learning rate was set to be 5×10^{-5} with a linear decay scheduler with no warm-up. We trained Doduo for 30 epochs and chose the checkpoint with the highest F1 score on the validation set.

Since the WikiTable dataset can have multiple labels on each column/column pair, we used Binary Cross Entropy loss to formulate as a multi-label prediction task. For the VizNet dataset, which only has a single annotation on each column, we used Cross Entropy loss to formulate as a multi-class prediction task. Models and experiments were implemented with PyTorch [34] and the Transformers library [54]. All experiments were conducted on an AWS `p3.8xlarge` instance (V100 (16GB)).

Following the previous studies [12, 60], we use micro F1 for the WikiTable dataset, and micro F1 and macro F1 for the VizNet dataset, as evaluation metrics. The micro F1 score is the weighted average of F1 values based on the sample size of each class, while the macro F1 score is the simple average of F1 values for all classes. Additional results and analysis can be found in the appendix of the full version [42].

5.4 Main Results

WikiTable Table 3 shows the micro F1 performance for the column type prediction and column relation prediction tasks on the WikiTable dataset. Doduo significantly outperforms the state-of-the-art method TURL on both of the tasks with improvements of 4.0% and 0.9%, respectively.

A significant difference in the model architecture between Doduo and TURL is whether the model uses full self-attention. In TURL, the model uses the self-attention mechanism with the "cross-column" edges removed, which they referred to as visibility matrix [12]. Let us use the example in Figure 3, which depicts how the contextualized embedding for the second column is calculated. TURL's visibility matrix removes the connections to `[CLS]`[2] from the cells "`Happy Feet`", "`Cars`", "`USA`", and "`UK`", whereas our Doduo uses the full set of connections.

Since TURL is designed for tables with meta information (e.g., table captions or column headers), we consider the major benefit of this design (i.e., the visibility matrix) to effectively incorporate descriptions in the meta information into table values. From the results, Doduo with the full self-attention performs better than TURL, which indicates that some direct intersections between tokens in different columns and different rows are useful for the column annotation problem.

Table 4: Performance on the VizNet dataset.

Method	Full		Multi-column only	
	Macro F1	Micro F1	Macro F1	Micro F1
Sherlock	69.2	86.7	64.2	87.9
Sato	75.6	88.4	73.5	92.5
Doduo	**84.6**	**94.3**	**83.8**	**96.4**

We also tested Doduo with metadata, which appends the column name to column values for each column. As shown in Table 3, by using column names, Doduo slightly improves the performance and performs competitively against TURL with metadata. This indicates that TURL relies on metadata and Doduo performs better and more robustly than TURL when metadata is not available.

VizNet Table 4 shows the results on the VizNet dataset. Note that Doduo is trained only using the column prediction task for the VizNet dataset, as column relation labels are not available for the dataset. The results show that Doduo outperforms Sherlock and Sato, the SoTA method for the dataset, by a large margin and establishes new state-of-the-art performance with micro F1 (macro F1) improvements of 11.9% (6.7%.)

As described in Section 2, Sato is a multi-column model that incorporates table context by using LDA features. Different from the LDA features that provide multi-dimensional vector representations for the entire table, the Transformer-based architecture enables Doduo to capture more fine-grained inter-token relationships through the self-attention mechanism. Furthermore, Doduo's table-wise design naturally helps incorporate inter-column information into the model.

Performance on numeric columns. As mentioned in Section 3, Doduo casts all cell values as strings thus it may be weak in handling numeric columns. Table 5 shows Doduo performance on the top-15 most numeric column types from the VizNet dataset. Doduo did have low performance on some numeric types such as ranking (33.21%) and capacity (62.55%). However, on these 15 types, Doduo achieves an average F1 score of 86.9% which is comparable to the overall macro F1 (84.6%) and even slightly better. This can be due to the Transformer model being able to recognize digit patterns in the numeric values to predict the correct types. This results aligns with the findings from the NLP literature [15, 50] that Transformer models can partially handle numeric data.

Table 5: Doduo's column type prediction performance on the 15 most numeric types of the VizNet dataset. We use %num to measure how many cell values of a type can be cast as a numeric type (e.g., `int`, `float`, `date`).

type	%num	F1	type	%num	F1	type	%num	F1
plays	100.00	88.55	fileSize	87.84	88.23	grades	67.18	97.68
rank	93.01	94.52	elevation	87.39	92.14	weight	60.41	97.59
depth	92.86	88.45	ranking	86.88	33.21	isbn	43.77	96.51
sales	92.05	75.13	age	81.04	98.53	capacity	42.06	62.55
year	91.47	98.94	birthDate	67.85	95.64	code	35.93	95.43

6 ANALYSIS

6.1 Ablation Analysis

To verify the effectiveness of multi-task learning and multi-column architecture, we tested variants of Doduo. Dosolo is a Doduo

Table 6: Ablation study on the WikiTable dataset.

Method	Type prediction	Relation prediction
Doduo	**92.50**	**91.90**
w/ shuffled rows	91.94	91.61
w/ shuffled cols	92.68	91.98
Dosolo	91.37 (1.23% ↓)	91.24 (0.7% ↓)
Dosolo$_{SCol}$	82.45 (21.9% ↓)	83.08 (9.6% ↓)

Table 7: Ablation study on the VizNet dataset (Full.)

	Macro F1	Micro F1
Doduo	**84.6**	**94.3**
Dosolo$_{SCol}$	77.4 (8.5% ↓)	90.2 (4.3% ↓)

model without multi-task learning. Thus, we trained Doduo models only using training data for the target task (i.e., column type prediction or column relation prediction.) Dosolo$_{SCol}$ is a single column model that only uses column values of the target column (or target column pair for column relation prediction.) Dosolo$_{SCol}$ is also trained without multi-task learning.

Table 6 shows the results of the ablation study. For both of the tasks, Dosolo degraded the performance compared to the multi-task learning of Doduo. As Dosolo$_{SCol}$ shows significantly lower performance than the others, the results also confirm that the multi-column architecture of Doduo successfully captures table context to improve the performance on the column type prediction and column relation prediction tasks.

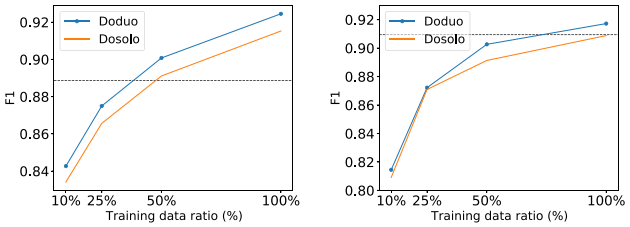
The same analysis with Dosolo$_{SCol}$ on the VizNet dataset is shown in Table 7. As expected, the multi-column model (Doduo) performs significantly better than the single-column model (Dosolo$_{SCol}$.) The results further confirm the strengths of the multi-column model. We would like to emphasize that Dosolo$_{SCol}$ outperforms Sato, which incorporates table context as LDA features.

The pre-trained LM (e.g., BERT) is sensitive to the input sequence order, which may not reflect the property of tables that rows/columns are order-invariant. To verify if Doduo has this limitation, we trained and evaluated Doduo on two versions of the WikiTable dataset, where the input table's rows (columns) were randomly shuffled. As shown in Table 6, somewhat surprisingly, Doduo shows only subtle degradation for shuffled rows and no substantial difference for shuffled columns. We conjecture that Doduo successfully tailors the original position embeddings to be aligned with the table structure during the fine-tuning step.

6.2 Data Efficiency

Learning Efficiency Pre-trained LMs are known for its capability of training high-quality models using a relatively small number of labeled examples. Furthermore, multi-task learning (i.e., training with multiple tasks simultaneously) should further stabilize the performance with fewer training data for each task. To verify the effectiveness of Doduo and Dosolo with respect to label efficiency, we compared the Doduo models trained with different training data sizes (10%, 25%, 50%, and 100%) and evaluated the performance on the column type prediction and column relation prediction tasks.

As shown in Figure 4, Doduo consistently outperforms Dosolo and achieves higher than 0.9 F1 scores on both tasks even when trained with half of the training data. In particular, with only 50% or fewer labeled examples, Doduo outperforms the SoTA method



(a) Column type prediction **(b) Column relation prediction**

Figure 4: Performance improvements over increasing the training data size on the WikiTable dataset. The dashed lines in the plots denote the state-of-the-art methods (TURL.)

Table 8: Comparisons of Doduo with different input token size on the WikiTable dataset.

MaxToken/col	Col type (F1)	Col rel (F1)	Max. # of cols
8	89.8	88.9	56
16	91.4	90.7	30
32	**92.4**	**91.7**	15

TURL on column-type prediction and achieves comparable performance on column relation prediction.

Input Data Efficiency A major challenge of applying pre-trained LMs to data management tasks is their limits of the maximum input sequence length. For example, LMs like BERT can only take at most 512 tokens so ingesting a full wide table may not be feasible for the LMs. Doduo (or the multi-column model in general) has the advantage that it is *input data efficient*, meaning that it can make table-wise predictions accurately by only taking a small number of samples of each column. This makes Doduo more attractive in practice as it can handle large tables with many columns.

Thus, we evaluated different variants of Doduo with shorter input token length to discuss the input data efficiency. Since any of the recent studies applying pre-trained Transformer-based LMs to data management tasks (e.g., [12, 25, 47, 52]) did not conduct this analysis, it is still not clear how many tokens should we feed to the model to obtain reasonable task performance.

Table 8 shows the results of Doduo with different max token sizes on the WikiTable dataset. We simply truncated column values if the number of tokens exceeded the threshold. As shown in the table, the more tokens used, the better performance Doduocan achieve, as expected. However, somewhat surprisingly, Doduo already outperforms TURL using just 8 tokens per column for column type prediction (TURL has micro F1 of 88.86 on WikiTable). For the column relation prediction task, Doduo needs to use more tokens to outperform TURL (i.e., 32 tokens to beat TURL's score of 90.94.) This is understandable, as column relation prediction is more contextual than column type prediction, and thus it requires more signals to further narrow down to the correct prediction. For the VizNet dataset, we confirm that Doduo with 8 max tokens per column with Doduo (92.5 F1) outperforms the state-of-the-art method (i.e., Sato) (88.4 F1) on the task. Table 8 reports how many columns each variant can support under the maximum token configuration. The average numbers of columns in Web Tables, Enterprise Data, and Open Data are reported to be 4, 12, and 16, respectively [11, 29]. Thus, we confirm that Doduo has a nice input data efficiency property, so it can be also used for "wide" tables.

We note that 16 columns may not be sufficient for tables outside of WebTables. For such cases, one option is to first split the wide table into clusters of relevant columns, then apply Doduo on each cluster. In this case, Doduo still has the advantage of leveraging partial context of the input table to improve prediction quality.

7 CASE STUDY: CLUSTERING COLUMNS

We apply Doduo to a real data science application scenario of clustering relevant columns. On a daily basis, data scientists collect information from multiple sources and incorporate data from various tables. Therefore, as a first step of data exploration and data integration [33], it is essential to know which columns are semantically similar. However, this is not always straightforward as different column names can be assigned to semantically similar columns. In this section, we demonstrate the usefulness of Doduo's contextualized column embeddings with a case study of clustering semantically similar columns on an enterprise database.

For this case study, we use an in-production enterprise database from the HR domain. Different teams create and update tables about job seekers and companies that are hiring, etc. Despite some company-wide naming conventions, we observe that semantically similar columns are sometimes given different column names across tables. Some tables have additional meta information describing the meaning of each column, which helps to understand the similarity of columns. However, the meta information is missing for many columns and is not always reliable as they are worded differently by different teams. Next, we simulate a workflow of a data scientist performing analysis related to job search and review of companies.

Scenario: Our data scientist Sofia starts by filtering tables with keywords "jobsearch" and "review" and gets **10** tables with **50** columns in total (**29** columns of type "string" and **21** columns of type "integer".) To group similar columns together, she can simply create contextualized column embedding using the Doduo toolbox for all the columns and then apply her favorite clustering algorithm to form the final groups.

To evaluate Sofia's column groups, we generate an initial cluster using both the column names and descriptions and manually refine the results to form the ground-truth[5] as shown below:

date, IP address, job title, timestamp (unixtime), timestamp (hhmm), counts, status, file path, browser, location, search term, rating, company ID, review ID, user ID

Sofia uses the Doduo model trained on the WikiTable dataset to obtain contextualized column embeddings for each column (Doduo+ column value emb.)

We compared the method with three baselines and two traditional schema matching approaches. We directly used Doduo's column type predictions as the clustering criteria where columns with the same predicted types got assigned into the same cluster (Doduo+predicted type). We tested fastText [1] to verify how *non*-contextualized column embeddings perform for the task. We use column value embeddings (fastText+column value emb) and column name embeddings (fastText+column name emb) with fastText. We choose fastText as a baseline as it offers a widely used off-the-shelf toolbox, which is a "go-to" option for data scientists. To achieve a fair evaluation of the embedding quality, we use the same k-means clustering algorithm for all models. In addition, we compared with more traditional schema matching approaches tested in the experiment suite Valentine [24]. We picked the two most effective approaches from Valentine's empirical study: COMA [14] and DistributionBased [61]. To generate the clustering label for comparison, we went over all possible pairs of tables and connected matched columns to assign the same cluster labels. Since those schema matching methods take two tables as input and return pairs of matched columns, we regard returned pairs as connected nodes in a graph and merge them into connected components to obtain column clusters. We use Homogeneity (Precision), Completeness (Recall), V-Measure (F1) to evaluate the quality of clusters using the ground-truth cluster assignment.

[5]We used column name and description for ground-truth creation purpose only as it is not available for all tables in practice. The initial clustering uses a combination of TF-IDF vectors and fastText embeddings.

Table 9: Case study results.

Method	Prec.	Recall	F1
Doduo+column value emb	**68.19**	70.40	**69.28**
Doduo+predicted type	44.87	61.32	51.82
fastText+column value emb	35.90	**76.61**	48.89
fastText+column name emb	56.62	74.68	64.40
COMA (with column name)	58.47	66.06	62.03
DistributionBased (with column name)	23.87	69.51	35.53

As shown in Table 9, Doduo's column embeddings show the best clustering performance with respect to Precision and F1. The results confirm that contextualized column embeddings are more useful than predicted column types and can be more accurate representations than column name/value embeddings created by fastText. Compared to Doduo, fastText tends to generate similar embeddings even for semantically different columns, which leads to creating unnecessarily large clusters that contain many irrelevant columns. This significantly increases (decreases) fastText methods' Recall (Precision) values in Table 9. COMA shows reasonably good performance and DistributionBased falls short on Precision. Doduo outperforms both matching-based approaches using the contextualized embedding. Note that both the column names and column descriptions are not given to the Doduo model as input, and the model was trained on the WikiTable dataset (i.e., different domain.) Thus, this case study also indicates the transferability of a Doduo model trained on one domain (i.e., Wikipedia tables) to another domain (i.e., enterprise database.) so that data scientists can apply the Doduo model in the toolbox for their own needs.

8 CONCLUSION

In this paper, we present Doduo, a unified column annotation framework based on pre-trained Transformer language models and Multi-task learning. Experiments on two benchmark datasets show that Doduo achieves new state-of-the-performance. With a series of analyses, we confirm that the improvements are benefited from the multi-task learning framework. Through the analysis, we also confirm that Doduo is data-efficient, as it can achieve competitive performance as the previous state-of-the-art methods only using 8 tokens per column or about 50% of training data. We conduct a case study and verify the effectiveness of Doduo on a real-world data science problem. We believe our toolbox will further help researcher/data scientists easily apply the state-of-the-art column annotation model to a wide variety of data science and data management problems.

REFERENCES

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://doi.org/10.1162/tacl_a_00051

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. 1247–1250.

[3] Matteo Cannaviccio, Denilson Barbosa, and Paolo Merialdo. 2018. Towards Annotating Relational Data on the Web with Language Models. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1307–1316. https://doi.org/10.1145/3178876.3186029

[4] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1335–1349. https://doi.org/10.1145/3318464.3389742

[5] Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning* (Amherst, MA, USA) *(ICML'93)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 41–48.

[6] R. Caruana. 2004. Multitask Learning. *Machine Learning* 28 (2004), 41–75.

[7] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *VLDB J.* 29, 1 (2020), 251–272.

[8] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019. ColNet: Embedding the Semantics of Web Tables for Column Type Prediction. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 29–36. https://doi.org/10.1609/aaai.v33i01.330129

[9] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019. Learning Semantic Annotations for Tabular Data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 2088–2094. https://doi.org/10.24963/ijcai.2019/289

[10] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proc. BlackBoxNLP '19*. 276–286.

[11] Dong Deng, Raul Castro Fernandez, Ziawasch Abedjan, Sibo Wang, Michael Stonebraker, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, and Nan Tang. 2017. The Data Civilizer System. In *8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2017/papers/p44-deng-cidr17.pdf

[12] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *Proc. VLDB Endow.* 14, 3 (nov 2020), 307–319. https://doi.org/10.14778/3430915.3430921

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[14] Hong-Hai Do and Erhard Rahm. 2002. COMA—a system for flexible combination of schema matching approaches. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 610–621.

[15] Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting Numerical Reasoning Skills into Language Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 946–958. https://doi.org/10.18653/v1/2020.acl-main.89

[16] GLUE. 2021. GLUE Leaderboard. *https://gluebenchmark.com/leaderboard* (2021).

[17] Google. [n.d.]. Google Data Studio. https://datastudio.google.com/

[18] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 1923–1933. https://doi.org/10.18653/v1/D17-1206

[19] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[20] Kevin Hu, Snehalkumar 'Neil' S. Gaikwad, Madelon Hulsebos, Michiel A. Bakker, Emanuel Zgraggen, César Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çağatay 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* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300892

[21] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1500–1508. https://doi.org/10.1145/3292500.3330993

[22] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics* 8 (2020), 423–438. https://doi.org/10.1162/tacl_a_00324

[23] Udayan Khurana and Sainyam Galhotra. 2021. Semantic Concept Annotation for Tabular Data. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 844–853. https://doi.org/10.1145/3459637.3482295

[24] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 468–479.

[25] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment* 14, 1 (Sep 2020), 50–60. https://doi.org/10.14778/3421424.3421431

[26] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow.* 3, 1–2 (Sept. 2010), 1338–1347. https://doi.org/10.14778/1920841.1921005

[27] Erin Macdonald and Denilson Barbosa. 2020. *Neural Relation Extraction on Wikipedia Tables for Augmenting Knowledge Graphs*. Association for Computing Machinery, New York, NY, USA, 2133–2136. https://doi.org/10.1145/3340531.3412164

[28] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1301.3781

[29] Renée J. Miller. 2018. Open Data Integration. *Proc. VLDB Endow.* 11, 12 (Aug. 2018), 2130–2139. https://doi.org/10.14778/3229863.3240491

[30] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. 2014. Using Linked Data to Mine RDF from Wikipedia's Tables. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining* (New York, New York, USA) *(WSDM '14)*. Association for Computing Machinery, New York, NY, USA, 533–542. https://doi.org/10.1145/2556195.2556266

[31] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, 1135–1145. https://www.aclweb.org/anthology/D12-1104

[32] Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. [n.d.]. MTab4DBpedia: Semantic Annotation for Tabular Data with DBpedia. ([n. d.]).

[33] Masayo Ota, Heiko Müller, Juliana Freire, and Divesh Srivastava. 2020. Data-Driven Domain Discovery for Structured Datasets. *Proc. VLDB Endow.* 13, 7 (March 2020), 953–967. https://doi.org/10.14778/3384345.3384346

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[35] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[36] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2463–2473. https://doi.org/10.18653/v1/D19-1250

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

[38] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model?. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 5418–5426. https://doi.org/10.18653/v1/2020.emnlp-main.437

[39] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. arXiv:1706.05098 [cs.LG]

[40] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Bießmann, and Andreas Grafberger. 2018. Automating Large-Scale Data Quality Verification. *Proc. VLDB Endow.* 11, 12 (2018), 1781–1794.

[41] Tableau Software. [n.d.]. Tableau. https://www.tableau.com/

[42] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2021. Annotating Columns with Pre-trained Language Models. arXiv:2104.01785 [cs.DB]

[43] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding. In *AAAI*. 8968–8975.

[44] Kunihiro Takeoka, Masafumi Oyamada, Shinji Nakadai, and Takeshi Okadome. 2019. Meimei: An Efficient Probabilistic Approach for Semantically Annotating Tables. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 281–288. https://doi.org/10.1609/aaai.v33i01.3301281

[45] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. 2021. RPT: Relational Pre-Trained Transformer is Almost All You Need towards Democratizing Data Preparation. *Proc. VLDB Endow.* 14, 8 (apr 2021), 1254–1261. https://doi.org/10.14778/3457390.3457391

[46] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *ACL*. 4593–4601.

[47] Mohamed Trabelsi, Jin Cao, and Jeff Heflin. 2020. Semantic Labeling Using a Deep Contextualized Language Model. arXiv:2010.16037 [cs.LG]

[48] 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 *Proc. NIPS '17*. 5998–6008.

[49] 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 (June 2011), 528–538. https://doi.org/10.14778/2002938.2002939

[50] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP Models Know Numbers? Probing Numeracy in Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 5307–5315. https://doi.org/10.18653/v1/D19-1534

[51] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *ICLR*.

[52] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. TCN: Table Convolutional Network for Web Table Interpretation. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 4020–4032. https://doi.org/10.1145/3442381.3450090

[53] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. TUTA: Tree-Based Transformers for Generally Structured Table Pre-Training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1780–1790. https://doi.org/10.1145/3447548.3467434

[54] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[55] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. 2020. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317* (2020).

[56] Yongxin Yang and Timothy M. Hospedales. 2017. Trace Norm Regularised Deep Multi-Task Learning. In *ICLR '17 Workshop Track*.

[57] Alexander Yates, Michele Banko, Matthew Broadhead, Michael Cafarella, Oren Etzioni, and Stephen Soderland. 2007. TextRunner: Open Information Extraction on the Web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. Association for Computational Linguistics, Rochester, New York, USA, 25–26. https://www.aclweb.org/anthology/N07-4013

[58] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 8413–8426. https://doi.org/10.18653/v1/2020.acl-main.745

[59] ChengXiang Zhai. 2008. Statistical Language Models for Information Retrieval: A Critical Review. *Found. Trends Inf. Retr.* 2, 3 (2008), 137–213. https://doi.org/10.1561/1500000008

[60] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. 2020. Sato: Contextual Semantic Type Detection in Tables. *Proc. VLDB Endow.* 13, 12 (July 2020), 1835–1848. https://doi.org/10.14778/3407790.3407793

[61] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M Procopiuc, and Divesh Srivastava. 2011. Automatic discovery of attributes in relational databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 109–120.