Watchog: **A Light-weight Contrastive Learning based Framework for Column Annotation**

ZHENGJIE MIAO, Megagon Labs, United States
JIN WANG*, Megagon Labs, United States

Relational Web tables provide valuable resources for numerous downstream applications, making table understanding, especially column annotation that identifies semantic types and relations of columns, a hot topic in the field of data management. Despite recent efforts to improve different tasks in table understanding by using the power of large pre-trained language models, existing methods heavily rely on large-scale and high-quality labeled instances, while they still suffer from the data sparsity problem due to the imbalanced data distribution among different classes. In this paper, we propose the Watchog framework, which employs contrastive learning techniques to learn robust representations for tables by leveraging a large-scale unlabeled table corpus with minimal overhead. Our approach enables the learned table representations to enhance fine-tuning with much fewer additional labeled instances than in prior studies for downstream column annotation tasks. Besides, we further proposed optimization techniques for semi-supervised settings. Experimental results on popular benchmarking datasets illustrate the superiority of our proposed techniques in two column annotation tasks under different settings. In particular, our Watchog framework effectively alleviates the class imbalance issue caused by a long-tailed label distribution. In the semi-supervised setting, Watchog outperforms the best-known method by up to 26% and 41% in Micro and Macro F_1 scores, respectively, on the task of semantic type detection.

CCS Concepts: • **Information systems → Mediators and data integration**.

Additional Key Words and Phrases: table understanding, contrastive learning, semi-supervised learning, web tables

ACM Reference Format:
Zhengjie Miao and Jin Wang. 2023. Watchog: A Light-weight Contrastive Learning based Framework for Column Annotation. *Proc. ACM Netw.* 1, N4 (SIGMOD), Article 272 (December 2023), 24 pages. https://doi.org/10.1145/3626766

1 INTRODUCTION

The importance of Relational Web tables has been widely recognized in the data management community, owing to the vast amount of knowledge they contain. Many large web table collections have been constructed in previous studies [5, 9, 28]. Table understanding plays a significant role in analyzing such table corpora [2, 3, 12], which can be leveraged for various data management applications, e.g. schema matching, dataset discovery, and data cleaning.

As illustrated in [9], there are two major categories of table understanding applications, namely table augmentation and table interpretation. Earlier studies [60, 61] provided feature-based methods as a solution. For example, Zhang et al. explored featured engineering efforts in entity linking [60]

*corresponding author

Authors' addresses: Zhengjie Miao, Megagon Labs, United States, zhengjie@megagon.ai; Jin Wang, Megagon Labs, United States, jin@megagon.ai.

2834-5509/2023/12-ART272 $15.00
https://doi.org/10.1145/3626766

and cell filling [61]. Hulsebos et al. [22] developed a feature-based framework for semantic type annotation, and Zhang et al. [59] further improved it with topic modeling. However, these methods were limited in their generality. Recent years have witnessed the burgeoning of pre-trained Language Models (LM) for NLP applications due to their ability to provide rich semantic and contextual information. Pre-trained LMs like BERT [10] have been widely applied in data management applications and achieved promising results [30, 43]. Deng et al. [9] proposed the first work that pre-trains an LM for multiple table understanding tasks, and several efforts have been made in the NLP field [19, 23, 54, 55] on pre-training LMs specifically for tabular data. Additionally, Wang et al. [45] and Suhara et al. [42] aimed at improving column annotation tasks, which is an essential subset of the table interpretation application, via fine-tuning.

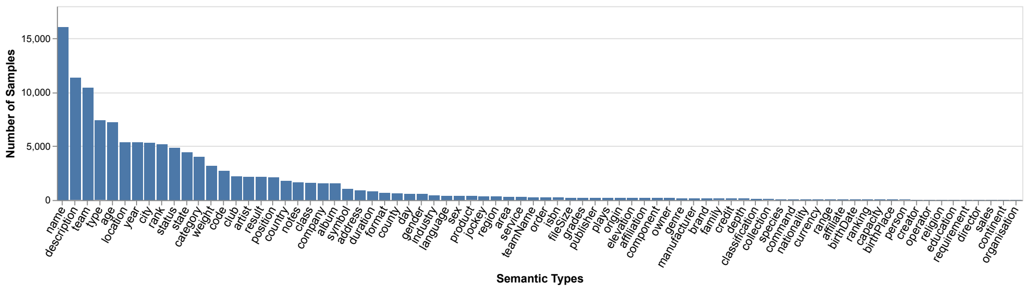

Fig. 1. The counts of 78 class labels in the VizNet that is a typical long-tail distribution shown in [59]. The data sparsity problem happens for most classes in the tail, even under a supervised setting.

Despite the impressive performance of previous fine-tuning based methods in table understanding tasks, their effectiveness largely hinges on the availability of annotated training instances. For example, the semantic type detection task in the TURL benchmarks [9] requires 628k labeled columns from 397k tables. While the ground truth of some table understanding tasks can be automatically generated by annotating the table corpus with a Knowledge Base (KB) [6] without significant human efforts, existing solutions may still suffer from data sparsity issues in both KBs and table corpus. For example, in Figure 1, the 78 semantic types in the VizNet dataset exhibit a typical long-tail distribution. We can see that it is rather imbalanced, with a frequent type *description* comprising 11,348 instances, while the minority types such as *depth*, *ranking*, and *education* only have 136, 55, and 36 instances, respectively. As a result, it is difficult for models to capture sufficient signals for types in the minority classes, even under supervised settings. Additionally, users need to create specialized training sets for different tasks in the fine-tuning process. Since those tasks have some shared knowledge in the unlabeled table corpus, such as [9], there will be certain wastes in proposing separate training sets and models for each task.

In this paper, we introduce Watchog, a unified framework that addresses the challenges of data sparsity and class imbalance in column annotation [32] tasks. Firstly, we employ the contrastive learning [7, 8, 24] techniques to learn table representations from a vast collection of unlabeled table corpus without any labeled instances. Then the learned table representations can be utilized in a variety of downstream applications in different ways. For instance, they can be used to determine the relatedness between two table units, such as columns from different tables, by computing the cosine similarity between the embedding vectors; or can be employed in fine-tuning with only a few additional labeled instances. Unlike other fine-tuning-based approaches that require a considerable number of labeled instances for every single task, the embeddings generated by Watchog can be used in various tasks with far fewer labeled instances. Meanwhile, Watchog is rather lightweight compared with previous studies [23, 55] that pre-train an LM for tabular data. In other words, with the help of contrastive learning, we can obtain a similar level of versatility while avoiding the expensive overhead of pre-training. The second major technique in Watchog is to utilize unlabeled data in a semi-supervised manner. We further come up with (i) a pseudo-labeling technique that can assign labels to unlabeled instances to obtain more training signals by leveraging the target model in the current epoch and (ii) a balance-aware labeling method to include more instances from minority and difficulty classes to resolve the data sparsity problem. With the help of such techniques, our proposed framework can achieve comparable results with state-of-the-art methods with less than half the number of labeled instances.

The contributions of this paper are summarized as follows:

- We propose a unified framework Watchog for column annotation based on contrastive learning. It can train a column representation model from a table corpus without any labeled instances. Then the pre-trained embeddings can be applied to various downstream tasks.
- We develop several optimizations for the semi-supervised scenario where the available labeled instances are much fewer. To the best of our knowledge, it is the first work that studied the problems of column annotation under semi-supervised settings.
- We conduct empirical studies on several popular benchmarking datasets. Experimental results illustrate that Watchog achieved the best performance under all settings. Specifically, Watchog can outperform the state-of-the-art method under the semi-supervised setting by up to 26% and 41% in Micro F1 and Macro F1, respectively.
- Besides, we also showcase that Watchog can clearly alleviate the issues caused by long-tail label distribution. Watchog can improve the average F1 score for the bottom 50 classes in the tail by up to 28% on the WikiTable dataset for semantic type detection.

The rest of this paper is organized as follows: Section 2 introduces necessary background knowledge and the overall framework. Section 3 describes the contrastive-learning-based framework for learning table representations. Section 4 proposes several semi-supervised learning techniques under low resource settings for column annotation. Section 5 presents the experimental results. Section 6 surveys the related works. Finally, Section 7 concludes the whole paper.

2 PRELIMINARY

In this section, we first introduce the terminology related to pre-trained language models in Section 2.1. Then we formally define the table understanding tasks in Section 2.2. Finally, we give an overview of our proposed framework in Section 2.3.

2.1 Pre-trained Language models

Recent years have witnessed a rapid advance in the application of large-scale, pre-trained language models in almost all NLP tasks. The idea of pre-trained LMs originated from Elmo [36] that aimed at learning contextual word embeddings by pre-training Bi-directional LSTM models. Then BERT [10] replaced LSTM in Elmo with Transformer [44] based architecture, which is a stack of self-attention layers that calculates distributed representations based on the similarity against all tokens. The stack of multiple self-attention layers produced contextual embedding of each input token. There are two steps for utilizing pre-trained LMs: pre-training and fine-tuning. In the *pre-training* step, the language model is trained on a large unlabeled corpus such as Wikipedia to gain deep language understanding via the designed pre-training tasks. For example, BERT defined two novel pre-training tasks: masked language model and next sentence prediction. In the *fine-tuning* step, the pre-trained model is fine-tuned with labeled training instances for the targeted task. This way, all downstream tasks can benefit from the common knowledge acquired in the pre-training

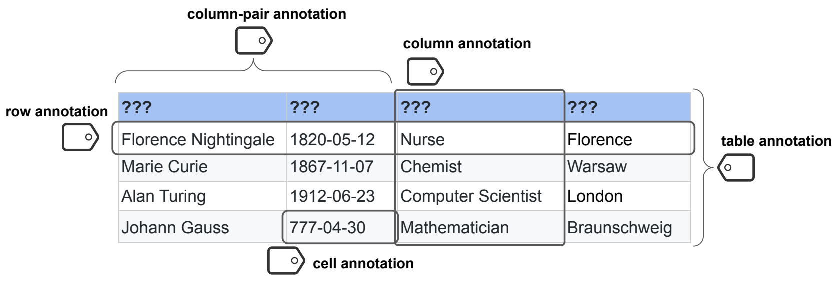

Fig. 2. Illustration of table understanding tasks. In this paper, we focus on two column annotation tasks, semantic type detection (Definition 2.1) and relation extraction (Definition 2.2).

step. Generally speaking, the pre-training step would be costly in the aspect of both hardware resources and training time. Meanwhile, fine-tuning is much cheaper, and the inference time is trivial compared with the fine-tuning time. And different kinds of pre-trained LM with similar model sizes tend to have a similar amount of fine-tuning and inference time for the same task.

Our proposed Watchog framework can be considered an intermediate between pre-training and fine-tuning. On the one hand, the contrastive learning process of Watchog is as lightweight as the fine-tuning step. On the other hand, it has the advantage that the outcome of contrastive learning, i.e., the trained table representations of Watchog, can provide shared knowledge for multiple downstream tasks in the process of fine-tuning. In the rest of this paper, we use the checkpoint of the pre-trained BERT model as the cornerstone to apply our proposed techniques.

2.2 Problem Definition

Table understanding is essential in many real scenarios. As shown in Figure 2, it is widely adopted in different variants of table annotation applications. A web table T is associated with two components: (i) the metadata m that includes information like the header, caption, or topic entities; (ii) the cell values. The header of T, denoted as H, consists of n columns $\langle C_1, C_2, \cdots, C_n \rangle$. We use H_i to denote the name of the i-th column and C_i to denote its cell values. Without generality, we assume that the cell values can be regarded as plain texts that are a sequence of tokens. Next, we will show the formal definitions of the two tasks, semantic type detection and relation extraction that are formally defined in previous studies [9, 42].

Definition 2.1 (Semantic Type Detection). Given a table T and a set of semantic types $\mathcal{L}$, semantic type detection aims at identifying a type label $l \in \mathcal{L}$ for each column $C \in T$ so that each cell in C has the same semantic types.

Definition 2.2 (Relation Extraction). Given a table T, a set of relations $\mathcal{R}$ and a pair of columns $C_i, C_j \in T$ $(i \neq j)$, relation extraction aims at identifying a relation label $r \in \mathcal{R}$ so that r describes the relation between all pairs of cells in the two columns.

Figure 2 shows a table about professionals to illustrate common table understanding tasks. In this paper, we primarily focus on predicting semantic column types and relations between column pairs. For example, the semantic type of the third row in the table is occupation, which can be inferred from the cell values “Nurse”, “Chemist”, and so on. Moreover, for the first and second rows, one could infer that their pairwise relation can be “is_born_on.” There are also other table understanding tasks, such as row population, cell filling, schema augmentation, etc., for which our framework can be extended to support by fine-tuning based on table representations learned from the contrastive learning process (in Section 3). Due to the space limitation, we leave it as future work.

2.3 Overall Framework

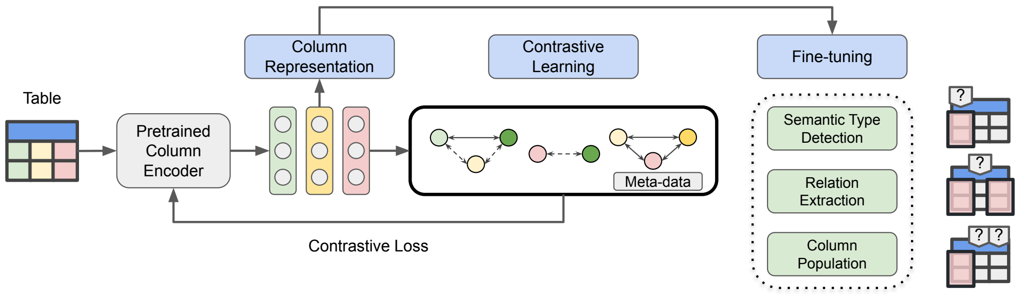

Fig. 3. The Overall Architecture of Watchog

Figure 3 shows the overall architecture of Watchog. The goal of Watchog is to first train an encoder for table representation, i.e., column encoder in this paper, without any labeled data and then fine-tune the model based on column embeddings for different downstream applications with only a few labeled instances. In the first step, the encoder is trained in a self-supervised manner which is similar to but much cheaper than the pre-training step of language models. To reach this goal, we need to make full use of available signals from the unlabeled table corpus to obtain high-quality representations. We address this issue with contrastive learning techniques (Section 3). Besides, improving the performance with much fewer labeled instances in the fine-tuning step is also essential. To this end, we propose two novel techniques in Section 4 based on a semi-supervised learning paradigm.

3 CONTRASTIVE LEARNING BASED FRAMEWORK

In this section, we describe how to employ contrastive learning techniques for column annotation tasks. We first introduce the general learning process in Section 3.1. Then we define the data augmentation operations for tabular data and use them in contrastive learning in Section 3.2. Finally, we discuss the way to utilize meta-data to improve contrastive data creation in Section 3.3.

3.1 Learning Algorithms

Contrastive learning is a popular paradigm for self-supervised learning. It aims at learning a data representation where similar instances are close to each other while dissimilar instances are far away from each other in the embedding space. We adopt SimCLR [7], one of the most popular contrastive learning algorithms, as the cornerstone of our Watchog framework. The goal is to learn an encoder $\mathcal{M}$ that encodes (a part of) a table into a high-dimensional vector representation. In this paper, we treat $\mathcal{M}$ as a column encoder for ease of presentation. The proposed techniques can be extended to train row or cell encoders similarly to support a wider scope of applications beyond column annotation.

The high-level idea of SimCLR is shown in Figure 4. To train a column encoder $\mathcal{M}$, we need to address two issues: (i) to create contrastive data for the training process without labeled instances. (ii) to define the contrastive loss function to minimize the distance between pairs of similar columns and maximize that between distinct ones at the same time.

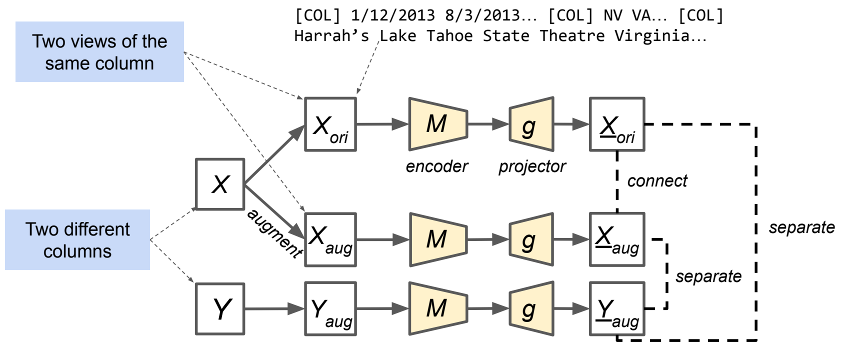
[COL] 1/12/2013 8/3/2013... [COL] NV VA... [COL]
Harrah's Lake Tahoe State Theatre Virginia...
Two views of the same column
Two different columns
X
Y
augment
X ori
X aug
Y aug
M
g
encoder
projector
connect
separate
separate

Fig. 4. Contrastive learning for table understanding with SimCLR

The creation of contrastive data includes the positive and negative instances for each original instance. The approach for positive data creation could vary for different applications. For tabular data, we will treat the different semantic-preserving views of the original instance as positive instances, e.g., in Figure 4 X_{aug} is considered as a positive instance for the original instance X_{ori}. In contrast, column Y, a different column, will be considered a negative instance. In order to generate different views of a column, we devise a set of *data augmentation operations for tables*, which will be detailed in Section 3.2 later. Meanwhile, creating negative instances is relatively straightforward: we can directly sample over the set of columns different from the original instance.

To reach the goal of connecting relevant columns while separating irrelevant ones, we employ contrastive loss over a batch of columns and their generated views. Given a batch of columns $\mathcal{X}$ with n original instances X_{ori}, we first use data augmentation operations to generate a different view X_{aug} that preserves the original semantic types for each X. We denote the embedding of X_{ori} and X_{aug} generated from the encoder $\mathcal{M}$ as $\boldsymbol{V}_{ori}$ and $\boldsymbol{V}_{aug}$, respectively. Then we merge $\boldsymbol{V}_{aug}$ with $\boldsymbol{V}_{ori}$ and formulate a batch $\boldsymbol{V}$ with $2n$ instances. Given an instance $\boldsymbol{v}_i \in \boldsymbol{V}$ where $\boldsymbol{v}_i$ is an original instance, suppose its augmented version is $\boldsymbol{v}_j \in \boldsymbol{V}$, then the contrastive loss between the single pair $\langle i, j \rangle$ could be calculated as in Equation (1):

$$\ell(i, j) = -\log \frac{\exp\left(sim\left(\boldsymbol{v}_i, \boldsymbol{v}_j\right)/T\right)}{\sum_{k=1}^{2n} \mathbb{1}_{[k \neq i, k \neq j]} \exp\left(sim\left(\boldsymbol{v}_i, \boldsymbol{v}_k\right)/T\right)} \tag{1}$$

where sim is the similarity between two vectors, and here we use cosine similarity; T is a hyper-parameter that controls the portion that the similarity contributes to the loss calculation. By minimizing such a loss between representations from the same column, we can maximize the score $sim(\boldsymbol{v}_i, \boldsymbol{v}_j)$ while minimizing the similarity between $\boldsymbol{v}_i$ and the representations of all other columns.

Following this route, we can further obtain the contrastive loss of the batch by averaging all pairs of positive instances shown in Equation (2):

$$\mathcal{L}_c = \frac{1}{2n} \sum_{k=1}^{n} [\ell(k, k+n) + \ell(k+n, k)] \tag{2}$$

where each term $\ell(k, k+n)$ and $\ell(k+n, k)$ refers to pairs of views generated from the same column.

Based on the above discussion, we then describe the whole contrastive learning process in Algorithm 1. It first initializes the column encoder $\mathcal{M}$ with the checkpoint of a pre-trained LM, where in this paper, we use BERT [1] by default (line: 1). Next we perform randomly sampling in the

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

Algorithm 1: Contrastive learning with SimCLR

```
Input: A collection $D$ of table columns
Variables :Number of training epochs n_epoch;
           Data augmentation operator op; Learning rate $\eta$
Output: A column encoder $\mathcal{M}$
1 Initialize $\mathcal{M}$ using a pre-trained LM;
2 for ep = 1 to n_epoch do
3     Randomly split $D$ into batches $\{B_1, \dots B_n\}$;
4     for $B \in \{B_1, \dots B_n\}$ do
          /* augment and encode every item */
5         $B_{\text{ori}}, B_{\text{aug}} \leftarrow$ augment($B$, op);
6         $V_{\text{ori}}, V_{\text{aug}} \leftarrow \mathcal{M}(B_{\text{ori}}), \mathcal{M}(B_{\text{aug}})$;
          /* Equation (1) and (2) */
7         $\mathcal{L} \leftarrow \mathcal{L}_c(V_{\text{ori}}, V_{\text{aug}})$;
          /* Back-prop to update $\mathcal{M}$ */
8         $\mathcal{M} \leftarrow$ back-propagate($\mathcal{M}, \eta, \partial\mathcal{L}/\partial\mathcal{M}$);
9 return $\mathcal{M}$;
```

table collection D and generate n batches B_i, $i \in [1, n]$ (line: 3). Then we apply data augmentation for each batch of columns to generate a different view of each column to provide the positive instances for contrastive learning (line: 5). We encode each column with $\mathcal{M}$ and calculate the contrastive loss using Equation (1) and (2) (line: 6 and 7). And the encoder $\mathcal{M}$ is updated with the contrastive loss via back-propagation (line: 8).

We can make further improvements based on Algorithm 1. Firstly, we find it essential to involve each column's context in the learning process. When learning the representation of one column, we should also take other columns in the same table into consideration. This can be realized by feeding a set of columns into the LM-based column encoder $\mathcal{M}$ simultaneously in the process of contrastive learning following the practice in [42]. To this end, the first step is to serialize the set of columns into a sequence by concatenating their cell values. Then for each column, we can insert a special token at the beginning of its values and use its output embedding as that of the column as previous studies did [30, 42]. The pre-trained LM first converts the input sequence into a sequence of token embedding independent of their context and then applies 12 Transformer layers, whose self-attention mechanism then converts the token embedding into a sequence of *contextualized embeddings*. In this way, each column representation depends not only on the tokens within itself but also on the context of those in other columns. The augmented instances consist of a subset of rows and columns in the original table. We will only take the embedding of the targeted column when computing contrastive loss, although the sampling and augmentation operations are done on the whole table.

Another finding is that in the process of generating positive pairs for contrastive learning, we perform two data augmentations on the same original instance X_{ori} and generate two different views X_{aug} and X'_{aug}. Then instead of making pairs of X_{ori} and X'_{aug} as the positive instance, we will treat the pair X_{aug} and X'_{aug} as a positive instance. The reason is that applying two data augmentation operators increases the difficulty of the contrastive learning objective: if we make positive pairs as the original instance and one augmented view of the instance, the overlap between them should be much higher than the overlap between two different augmented views, which makes learning the invariant properties easier. In other words, learning that the representations of two different augmented views are close should make the model "learn harder" than using only one augmented view. The original SimCLR paper [8] has the same finding that performing data augmentation only for one view works worse than using two augmented views.

To apply such improvements in Algorithm 1, we make the following changes: Firstly, when sampling the batches of tables(line: 3), we keep the whole tables to be fed into the pre-trained LM and later use each individual column embedding for computing the contrastive loss (line: 7). When applying the data augmentation operators (line: 5), we generate two views with different operators as positive instances.

3.2 Data Augmentation for Tables

Next, we introduce the data augmentation operations for creating different views of a column. Data Augmentation (DA) is a popular technique to generate labeled instances without human labors automatically [52]. It has been widely applied in many applications in the NLP and data management fields [31]. The typical data augmentation operator for textual data includes the replacement, insertion, deletion and swap of different granularities of text, i.e. token and span. However, here we need to perform DA operators over tables; thus, such existing approaches cannot be directly applied.

To solve this problem, we develop a series of DA operators for tables. Similar to DA operators for textual data, we categorize the DA operators based on the levels of granularity they are applied on, namely table, column, and row-level ones. The cell-level operators are general transformations also used in related NLP tasks. The row and column-level operators cover different ways of creating samples of rows/columns. The details of DA operators are summarized in Table 1. To use a DA operator to create contrastive data, we first need to perform uniform sampling to obtain a set of tables. Then the batches of columns mentioned in line 5 of Algorithm 1 are obtained from such tables after applying the DA operator. One can also perform more complex transformations by applying multiple operators simultaneously.

Table 1. DA operators for tables at different levels.

Level	Operators	Description
Cell	drop_cell, drop_token, swap_token, repl_token	Dropping a random cell; Dropping/swapping tokens within cells
Row	sample_row, shuffle_row	Sampling x% (e.g., 50) of rows; Shuffling the row order
Col	drop_col, drop_num_col, shuffle_col	Dropping$X$ (numeric) columns; Shuffling column order

In this work, we select the DA operators empirically based on the guidelines provided in [31]. We find that the combination of operators drop_col and shuffle_row provides the best overall performance for all tasks. It is also possible to support automatically finding the best set of DA operators with machine learning techniques following previous efforts like [34]. We will leave it as a future work to explore.

3.3 Incorporation of Meta-data

Finally, we explore the method of utilizing the meta-data, i.e., the headers, captions, and topics, to enhance the overall performance when such kinds of meta-data are available in the table corpus for contrastive learning. It is also common to use table meta-data when pre-training LMs on tabular data [9, 55]. Since such information is essential in many tasks, it would be beneficial to include them in the contrastive learning process. For example, column annotation tasks like semantic type detection sometimes relate to the column headers. If we can include the headers in the contrastive learning process, the learned column embedding with the same header will tend to be more similar. As a result, it could save certain efforts in the fine-tuning process for the model to recognize such a rule.

First, we introduce how to utilize header information. Recall that in the process of contrastive data creation, we assume that a column is a positive instance for a given column if and only if it is an augmented view of it. Otherwise, it will be regarded as a negative instance. With the help of the header information, we can alleviate this constraint to that *if a column has the same header name as the original column, it could also be considered a positive instance.* In this way, we can broaden the scope of positive contrastive data using the signals provided by the headers. To apply this optimization into Algorithm 1, we just need to replace Equation (2) with the following Equation (3)

$$\mathcal{L}_c' = \frac{1}{2|H|} \sum_{(i,j)\in H} [\ell(i,j) + \ell(j,i)]. \tag{3}$$

where H is the set of column pairs $\langle i, j \rangle$ s.t. (i) columns i and j are different views of the same column or (ii) i and j share the same header name.

Next, we discuss how to utilize the caption and topic entities. They could be considered as the context of a table. There are two ways to utilize them: Firstly, we can follow the idea in [45] to learn a contextual representation $\boldsymbol{V}_{ctx}$ of them. Then given a column X, its embedding will be the element-wise product between $\mathcal{M}(X)$ and $\boldsymbol{V}_{ctx}$. Secondly, we can calculate the cosine similarity between the contextual representation vectors of the two tables. Then if two columns from these two tables have the same header, we will consider there is a probability that they can form a positive pair. And the probability could be decided by (or directly equal to) the cosine similarity between the contextual representation vectors. Such techniques might have certain limitations if the data quality of meta-data is not good. In such cases, more efforts in denoising are required before using meta-data.

4 IMPROVING SEMI-SUPERVISED SETTINGS

4.1 Challenges

After training the column encoder $\mathcal{M}$ with contrastive learning, we can then conduct fine-tuning over it in different downstream tasks. In previous studies based on pre-trained LM [9, 42, 45], the fine-tuning process still requires hundreds of thousands of labeled instances. In this section, we explore how to improve the performance in the process of fine-tuning when there are very few labeled instances. To this end, we rely on semi-supervised learning techniques where models learn from a small amount of *labeled data* and a large amount of *unlabeled data.* And the research challenges to be resolved are as follows: Firstly, it is essential to make full use of the unlabeled data to obtain rich signals for training. Secondly, due to the unbalanced distribution of labels in table understanding tasks, some minority classes would suffer from the data sparsity problem. Thus, how to collect enough training instances for such classes is also crucial to train a good model.

4.2 Consistency Regularization

First we introduce the technique of consistency regularization, which has been recently shown to be an important component in semi-supervised learning [1, 4, 41]. Intuitively, consistency regularization encourages the target model to make similar predictions on the perturbed variants of the same input example, which is crucial in assigning proper labels to unlabeled instances. This is realized by data augmentation operators discussed in Section 3.2. Similar to self-supervised contrastive learning, where the encoder learns to minimize the distance between different views of the input column representation, consistency regularization lets the target model output close probability distributions on different views of the input column for table understanding tasks.

For example, given an unlabeled column of person names, there could be many possible candidates of its semantic type. In the VizNet dataset, it could be either person, director, artist, and so on, depending on the context. Although we do not have its ground truth label, the target model should return similar probability distributions for the augmented view of this column, like a shuffled version or a subset of the original column. This idea could be formalized as below:

For a multi-class classification problem, suppose we have a batch of b labeled instances W and a batch of $\mu \cdot b$ unlabeled ones U, where b is the batch size, and μ is a hyper-parameter determining the ratio of unlabeled to labeled instances. Each item in W is a pair of instance and the ground truth label (w_i, y_i); And those in U only have the instance u_i. For each u_i we create 2 new instances $\{u_i^1, u_i^2\}$ via DA operators introduced in Section 3.2.

Let $\mathcal{M}(w)$ be the predicted class distribution for an input example w with the encoder in the current epoch; the idea of consistency regularization is to minimize the loss function on the unlabeled data defined in Equation (4):

$$\frac{1}{\mu b} \sum_{i=1}^{\mu b} || \mathcal{M}(u_i^1) - \mathcal{M}(u_i^2) ||_2 \tag{4}$$

4.3 Pseudo Labeling

Based on the idea of consistency regularization, we then propose a *pseudo-labeling* technique to create training signals from unlabeled data. The basic idea of pseudo-labeling is to leverage the prediction of the target model $\mathcal{M}$ to obtain training signals [35]. The model's prediction is a probability distribution among different classes that can be further post-processed for classification. A straightforward solution for pseudo-labeling is to assign a "hard" label to each unlabeled instance, which adopts the class with the highest probability as the pseudo-label. Now the loss function on the unlabeled data is

$$\frac{1}{\mu b} \sum_{i=1}^{\mu b} \mathbb{1}(\max(\mathcal{M}(u_i^1)) \geq \tau)\, H(\mathrm{pl}(u_i^1), \mathcal{M}(u_i^2)) \tag{5}$$

where $\mathbb{1}()$ is the indicator function to select unlabeled examples with high confidence (probability higher than the scalar hyper-parameter τ) as training targets, $\mathrm{pl}(u_i^1) = arg\max(\mathcal{M}(u_i^1))$ is the "hard" one-hot pseudo-label for an augmented view of the unlabeled example u_i, H is the cross-entropy loss. This loss function requires the target model's output on the second augmented view of the unlabeled example to match the pseudo-label.

4.4 Balance-aware Optimizations

With the help of pseudo-labeling, the performance of the target model can be obviously improved since there are more training instances. However, it still cannot fully address the long-tail problem in label distribution. The reason is that pseudo-labeled instances are selected according to the fixed confidence threshold τ. Since the number of instances in minority classes is much smaller than that of majority ones, the same imbalance distribution also tends to happen in the newly generated instances. As shown in [17, 58], the issue of imbalanced distribution can even be aggravated under semi-supervised settings because the pseudo-labels are still highly class-imbalanced and it will lead to biased classification toward major classes. To solve this problem, we need to balance the unlabeled data fed to the target model via re-weighting or re-sampling the unlabeled data by a factor inversely proportional to the number of examples of each class.

In this work, we resolve this problem by adjusting the selection criterion $\mathbb{1}(\max(\mathcal{M}(u_i^1)) \geq \tau)$ in the loss function for unlabeled data (Equation 5) in the following ways: (1) we use the probability class distribution by the target model as a "soft" label instead of using the one-hot "hard" label, and (2) we adapt the idea of curriculum pseudo-labeling in [58] to adjust the confidence threshold τ dynamically. Intuitively, we want to feed the model with more instances in the classes with lower accuracy by lowering the confidence threshold for those classes. This new unsupervised loss is illustrated in Equation (6):

$$\frac{1}{\mu b}\sum_{i=1}^{\mu b}\mathbb{1}(\max(\mathcal{M}(u_i^1)) \geq \tau \cdot prop_t(\mathsf{pl}(u_i^1)))\; H(\mathcal{M}(u_i^1), \mathcal{M}(u_i^2)) \tag{6}$$

where $prop_t(c)$ is the proportion of unlabeled examples hard-labeled to class c by the model at epoch t to the number of examples in the largest predicted class:

$$prop_t(c) = \frac{\sum_{i=1}^{\mu b}\mathbb{1}(arg\max(\mathcal{M}(u_i)) == c)}{\max_c \sum_{i=1}^{\mu b}\mathbb{1}(arg\max(\mathcal{M}(u_i)) == c)} \tag{7}$$

With the standard classification loss on the labeled data:

$$\mathsf{L}_l = H(y_i, \mathcal{M}(w_i)) \tag{8}$$

the loss function for the semi-supervised setting is

$$\mathsf{L} = \mathsf{L}_l + \lambda_u \mathsf{L}_u \tag{9}$$

where λ_u is a hyper-parameter denoting the ratio of unlabeled loss to labeled loss.

Finally we summarize the semi-supervised learning process in Algorithm 2.

Algorithm 2: Semi-supervised learning with balance-ware optimizations

```
Input: $W$: set of labeled examples; $U$: set of unlabeled examples
Variables :Number of training examples $b$; number of training epochs $E$; number of classes $C$;
            the ratio of unlabeled examples $\mu$;
            the confidence threshold $\tau$;
Output: the classification model $\mathcal{M}$
1 Initialize $\mathcal{M}$ using the LM-based column encoder;
2 for $t = 1$ to $E$ do
3     for $i = 1$ to $\mu b$ do
4         Calculate $\mathcal{M}(u_i^1)$ and record the predicted class $\mathsf{pl}(u_i^1)$
5     for $c = 1$ to $C$ do
6         Calculate $prop_e(c)$ with Equation (7) ;
7     Calculate the loss $\mathsf{L}_l$, $\mathsf{L}_u$, and $\mathsf{L}$ with Equation (8), (6), (9) and back-propagate;
8 return $\mathcal{M}$;
```

5 EXPERIMENTS

5.1 Experiment Setup

5.1.1 Datasets. For the two tasks of semantic type detection and relation extraction, we evaluate the proposed methods on VizNet [20] and WikiTable [9] datasets that are widely used in previous studies. WikiTable is a collection of web tables from Wikipedia [2] proposed in [9]. It provides benchmarking datasets for both semantic type detection (ST) and relation extraction (RE) tasks. The ground truth of semantic type detection is obtained by aligning Freebase with the web tables. There are 255 semantic types and 570,171 tables in total. The training set consists of 628,254 columns from 397,098 tables, while the validation and test set consists of 13,391 columns from 4,844 tables and 13,025 columns from 4,764 tables, respectively. The ground truth of relation extraction is constructed in a similar way. There are 121 relation types and 78,733 tables in total. The training set consists of 62,954 column pairs from 52,943 tables, while the validation and test set consists of 2,175 column pairs from 1,560 tables and 2,072 column pairs from 1,467 tables, respectively. We strictly follow the previous study for the split of training, validation, and test sets.

The VizNet dataset is processed in the previous study [59] on the basis of the VizNet corpus [20] [3]. It only provides the benchmarking dataset for the semantic type detection task. There are 78 column types and 119,360 columns from 78,733 tables in total. We used the same splits for the 5-fold cross-validation following that in [59].

To evaluate over the semi-supervised setting, we randomly sampled a subset of training instances from the above datasets. Specifically, we try two sampling strategies: uniform and balanced ones.

- Uniform: We make sure that every class appears at least once. Based on that, we perform uniform random sampling from the whole dataset to obtain a certain portion of columns. The number of columns with each class label in the sampled dataset has a similar distribution to that in the original dataset. In our experiments, we try to sample 2%, 5% and 10% of the original dataset.
- Balanced: We randomly sample X columns for each class label to make the sampled dataset balanced. If the original dataset contains less than X columns, we will include all the columns. We try to set X as 20, 50, and 100 in our experiments.

5.1.2 *Baseline Methods.* We choose the following existing solutions as baseline methods:

SATO [59] is a multi-column prediction model, which extracts features with different granularities from the texts in a table to form the representation vector following the practice of Sherlock [22]. Besides, it also makes an extension by adding LDA features to capture table context and a CRF layer to incorporate column type dependency into prediction.

TURL [9] is a pre-trained Transformer-based LM for table understanding. It pre-trains an LM considering the structure of tables, so the model becomes more suitable for tabular data. To perform column type/relation annotation, we fine-tuned the pre-trained TURL model on the same training sets as for other baselines. We do not use entity and meta-data in fine-tuning for a fair comparison with other methods.

Doduo [42] is a recent approach for column annotation based on pre-trained LMs. It employed multi-task learning to train a model for different tasks jointly. And it is the state-of-the-art method for the two tasks evaluated in this paper.

Starmie [13] employs contrastive learning to train a column encoder to support table search related applications. We directly use the outcome of its contrastive learning approach as the starting point for fine-tuning.

5.1.3 *Evaluation Metrics.* We follow previous studies and use F_1 score as the primary evaluation metric. Since the two tasks belong to the multi-class classification problem and the label distribution is very imbalanced, we report the results of both Micro F_1 and Marco F_1. Specifically, the Macro F_1

[2]https://github.com/sunlab-osu/TURL#data
[3]https://github.com/megagonlabs/sato/tree/master/table_data

is calculated by averaging the F_1 score from all classes as Equation (10):

$$F_{macro} = \frac{1}{L} \sum_{i=1}^{L} F_1(i) \tag{10}$$

where L is the number of classed and $F_1(i)$ is the F_1 score of the i-th class. For the multi-class classification task, the Macro F_1 is more important in many situations, especially when the classes have long-tail distribution.

5.1.4 Environment. We implement Watchog in Python using Pytorch and the Hugging Face Transformers library. We use BERT [10] as the base language model for contrastive learning. We use Adam [25] as the optimizer in the training process. All experiments are run on a server with configurations similar to those of a g5.12xlarge AWS EC2 machine with 4 Nvidia A10g GPUs. The server has 1 AMD EPYC 7R32 48-Core processor and 192GB RAM. The hyper-parameter for fine-tuning might differ in different tasks, which will be detailed in Section 5.2. We ran all experiments 5 times and reported the average result.

5.1.5 Settings for Contrastive Learning. For the unsupervised contrastive learning process, we use the same Wikitable corpus consisting of 570,171 tables as in TURL [9] and train our models for 10 epochs. In addition, we set the batch size to 32, the learning rate to 5e-5, the max sequence length to 256, and the temperature hyper-parameter T in Equations 1 to 0.05.

5.2 Main Results

Table 2. Main results under Supervised settings. ST and RE denote semantic type detection task and relation extraction task, respectively.

Task	Method	Micro F_1	Macro F_1
WikiTable ST	TURL	88.86	69.95
	Doduo	92.45	77.60
	Starmie	92.36	77.00
	Watchog w/o meta	92.53	76.06
	Watchog	**93.33**	**78.72**
WikiTable RE	TURL	90.94	81.52
	Doduo	91.90	84.73
	Starmie	92.33	86.41
	Watchog w/o meta	92.38	86.72
	Watchog	**92.98**	**88.45**
VizNet ST	Sherlock	86.7	69.2
	SATO	88.4	75.6
	Doduo	94.3	84.6
	Starmie	93.97	83.57
	Watchog w/o meta	94.07	83.79
	Watchog	**95.02**	**85.63**

5.2.1 Fully-supervised setting. We first look at the results in the supervised setting. For each downstream task, we added an output layer on top of the language model for prediction. Following

Table 3. The mean imbalance rate of training data under semi-supervised setting for two sampling strategies.

Balanced	$X = 20$	$X = 50$	$X = 100$
WikiTable ST	58.76	57.57	57.53
WikiTable RE	7.99	8.05	7.94
VizNet ST	7.61	18.58	33.74
Uniform	2%	5%	10%
WikiTable ST	3,698.00	9186.80	6584.89
WikiTable RE	251.40	335.87	250.01
VizNet ST	205.00	504.00	1018.20

Table 4. The cardinality of training data and the corresponding portion to the original dataset with the balanced sampling strategy.

Task	$X = 20$	$X = 50$	$X = 100$
WikiTable ST	5,105 (0.8%)	12,552 (2%)	24,963 (3.9%)
WikiTable RE	3,606 (5.7%)	8,949 (14%)	17,800 (28.3%)
VizNet ST	1,934 (1.6%)	4,420 (3.7%)	7,765 (6.5%)

the settings in Doduo [42], we use cross-entropy loss for the VizNet dataset and binary cross-entropy loss for the WikiTable dataset. For the fine-tuning phase, we choose the hyper-parameters based on the performance on the validation set. For all tasks, we set the batch size to 32, the learning rate to 5e-5, and the max sequence length to 256. We train all models for 30 epochs for VizNet ST and WikiTable RE and 20 epochs for WikiTable ST. We report the performance of the epoch with the highest Macro/Micro F_1 score on the validation set.

The results are shown in Table 2. The choice of baseline methods for each task or dataset is aligned with those in previous studies [9, 42, 59]. The method Watchog w/o meta is the one in that we do not include optimizations with metadata in the contrastive learning process, i.e., techniques in Section 3.3. Since the original studies do not report results of Macro F_1 score on WikiTable dataset, we run experiments using the code provided by the authors and obtain the results by ourselves. We can see that Watchog achieves the best performance under all settings. The reason is that even under a fully supervised setting, the contrastive learning process can still bring some benefits to the encoder to learn a better column representation, which also improves the fine-tuning of different downstream applications. Specifically, compared with the Doduo method, which relies on fine-tuning the pre-trained LMs, Watchog shows more significant improvement in Macro F_1. The reason could be that the proposed techniques can help provide more signals to the minority classes to alleviate the data sparsity problem in those classes.

5.2.2 Semi-supervised setting. Next, we show the results under semi-supervised settings. We set $\lambda_u = 0.005$ for VizNet dataset and $\lambda_u = 0.01$ for WikiTable dataset. We use the same set of rest hyper-parameters ($\tau = 0.95$, $\mu = 7$, b=16) for all semi-supervised settings, except for WikiTable RE $X = 100$ we set $\mu = 3$ because there are not that many examples in the training set. In the experiments for semi-supervised settings, the number of labeled instances is crucial for the overall performance. To ensure a fair comparison, we do not use the multi-task learning version of Doduo, which might roughly double the number of labeled instances. We will still use the notion Doduo in the following part if there is no ambiguity. We report the results of both sampling strategies. To show the extent that imbalanced distribution happened under each sampling strategy, we provide

Table 5. Main results under semi-supervised settings with uniform sampling. ST and RE denote the application of semantic type detection and relation extraction, respectively.

Task	Method	2%		5%		10%	
		Micro F_1	Macro F_1	Micro F_1	Macro F_1	Micro F_1	Macro F_1
WikiTable ST	TURL	1.16	0.03	79.45	22.24	84.46	43.67
	Doduo	76.81	11.08	83.3	33.13	86.35	53.17
	Starmie	77.57	11.98	83.59	33.00	86.33	54.07
	Watchog	**82.04**	**32.61**	**84.84**	**49.04**	**86.53**	**57.49**
WikiTable RE	TURL	0	0	0	0	5.74	0.61
	Doduo	3.33	0.47	49.63	12.58	75.34	46.88
	Starmie	3.04	0.56	58.46	16.36	78.40	51.96
	Watchog	**64.91**	**26.66**	**80.33**	**60.11**	**84.32**	**71.98**
VizNet ST	SATO	68.15	33.58	73.98	43.97	79.29	51.97
	Doduo	74.19	38.39	81.86	52.32	85.73	62.03
	Starmie	75.85	40.86	83.11	54.15	86.76	63.50
	Watchog	**80.72**	**48.89**	**85.74**	**58.31**	**88.52**	**67.01**

the information on the imbalance rate for each dataset in Table 3. The definition of imbalance rate is calculated by dividing the cardinality of the most frequent class and that of the least frequent one. We can observe that the balanced sampling strategy can make the distribution relatively even while the uniform one will keep the original distribution.

For the balanced sampling that requires the number of instances for each class instead of the total number, we provide the cardinality of datasets as well as their portion of the original datasets in Table 4 for clarification. The cardinality of datasets varies based on that of the original ones. For example, on ST datasets, the cardinality from a balanced sampling strategy is smaller than that of a uniform one because (i) the cardinality of original datasets is larger and (ii) there are a larger number of classes, especially minority classes.

First, we look at the performance with the uniform strategy in Table 5. We can see that Watchog outperforms baseline methods by a significant margin. For example, on the semantic type detection task over WikiTable dataset when the sampling rate is 2% and 5%, Watchog outperforms state-of-the-art method Doduo in Macro F_1 score by 21.53 and 15.91, respectively. Moreover, we can see that the TURL method cannot produce reasonable outcomes for both tasks on the WikiTable dataset under most settings. This illustrates that it is not feasible to directly apply fine-tuning methods based on supervised settings, such as TURL and Doduo, to semi-supervised settings. Thus it is essential to provide new techniques to address this problem. And our work fills this gap with the help of proposed techniques such as pseudo labeling and balance aware optimization.

Then we report results of the balanced sampling strategy in Table 6. We can see that in this situation, the advantage of Watchog is more remarkable over Doduo than that with uniform sampling. The reason could be that Doduo is underfitting due to insufficient training instances for the majority classes. Thus it is more difficult for Doduo to distinguish between different classes in the test set. Meanwhile, TURL achieves some good results on the semantic type detection task on Micro F_1 when X becomes larger. However, the overall results are still much worse than Watchog. The reason could be that TURL tends to predict the majority classes while the performance on minority classes is somewhat limited.

Table 6. Main results under semi-supervised settings with balanced sampling. ST and RE denote the application of semantic type detection and relation extraction, respectively.

Task	Method	$X = 20$		$X = 50$		$X = 100$	
		Micro F_1	Macro F_1	Micro F_1	Macro F_1	Micro F_1	Macro F_1
WikiTable ST	TURL	0	0	0.23	0.01	76.14	45.86
	Doduo	50.04	1.84	75.68	36.48	79.74	56.82
	Starmie	50.87	1.77	75.85	38.12	79.85	57.11
	Watchog	**76.31**	**43.05**	**79.02**	**56.41**	**80.50**	**57.97**
WikiTable RE	TURL	0	0	0	0	78.17	69.28
	Doduo	1.32	0.09	67.49	56.45	81.79	76.91
	Starmie	2.96	0.27	75.39	66.50	83.17	79.29
	Watchog	**76.72**	**68.97**	**82.38**	**77.10**	**83.88**	**79.82**
VizNet ST	SATO	47.59	33.86	53.81	37.59	59.11	41.62
	Doduo	51.42	40.57	55.94	45.05	61.57	48.86
	Starmie	50.17	40.27	56.18	45.72	60.50	48.32
	Watchog	**57.01**	**45.63**	**59.86**	**49.38**	**63.69**	**53.86**

Table 7. Ablation study: semi-supervised settings with uniform sampling

Task	Method	2%		5%		10%	
		Micro F_1	Macro F_1	Micro F_1	Macro F_1	Micro F_1	Macro F_1
WikiTable ST	Watchog	82.04	32.61	84.84	49.04	86.53	57.49
	w/o BL	79.72	18.70	84.02	38.33	86.49	54.50
	w/o BL+PS	78.52	12.75	84.47	35.75	86.07	55.46
	w/o BL+PS+MT	77.44	11.75	83.71	33.58	86.69	54.32
WikiTable RE	Watchog	64.91	26.66	80.33	60.11	84.32	71.98
	w/o BL	28.51	4.55	69.37	33.82	82.27	63.70
	w/o BL+PS	2.57	0.39	57.43	15.82	80.50	57.06
	w/o BL+PS+MT	3.54	0.47	60.38	18.51	78.93	53.42
VizNet ST	Watchog	80.72	48.89	85.74	58.31	88.52	67.01
	w/o BL	80.21	47.45	85.58	58.77	88.09	66.63
	w/o BL+PS	79.67	46.96	84.98	57.33	87.05	66.25
	w/o BL+PS+MT	76.67	43.34	83.54	55.20	85.53	64.05

Finally, we want to discuss the efficiency of our proposed framework. As illustrated in Section 2.1 before, the bottleneck of the whole process is the contrastive learning process, while the fine-tuning and inference time is similar for all methods. The contrastive learning process takes around 6.2 hours on average when running on 4 A10g GPUs in parallel using Distributed-Data-Parallel in PyTorch. The peak memory usage is 63.9 GB in total on 4 GPUs, where each GPU has 24GB of memory. The fine-tuning time for the three tasks on the full dataset using a single GPU is around 5.7, 23.8, and 5.0 hours, respectively. Meanwhile, the pre-trained LMs for tabular [23, 55] required days or even weeks of training time with much more hardware resources. Thus our proposed techniques can benefit multiple downstream tasks with much less overhead.

Table 8. Ablation study: semi-supervised settings with balanced sampling

Task	Method	$X = 20$		$X = 50$		$X = 100$	
		Micro F_1	Macro F_1	Micro F_1	Macro F_1	Micro F_1	Macro F_1
WikiTable ST	Watchog	76.31	43.05	79.02	56.41	80.50	57.97
	w/o BL	76.45	42.90	78.16	40.52	79.78	57.82
	w/o BL+PS	49.46	1.87	78.27	42.65	79.50	57.97
	w/o BL+PS+MT	44.09	1.34	76.32	37.87	79.48	56.75
WikiTable RE	Watchog	76.72	68.97	82.38	77.10	83.88	79.82
	w/o BL	33.48	10.28	80.57	74.38	83.79	79.50
	w/o BL+PS	4.06	0.37	78.15	70.31	83.72	80.43
	w/o BL+PS+MT	1.24	0.05	74.05	63.78	83.54	79.89
VizNet ST	Watchog	57.01	45.63	59.86	49.38	63.69	53.86
	w/o BL	56.33	45.16	60.07	49.31	63.66	51.93
	w/o BL+PS	55.89	43.84	59.09	47.13	63.58	51.10
	w/o BL+PS+MT	51.92	41.02	56.30	45.59	60.94	50.35

5.3 Ablation Study

Next, we conduct some ablation studies to show the effect of each proposed technique under semi-supervised settings. To this end, we evaluate 3 methods by removing the components one by one. BL is the optimization of balancing the accuracy across classes when doing pseudo labeling (Section 4.4). PS is the use of unlabeled data along with consistency regularization and pseudo labeling techniques (Section 4.2 and 4.3). MT is the technique of utilizing meta-data in the contrastive learning process (Section 3.3). The results on uniformly sampled data are shown in Table 7. We can see that the proposed techniques can help improve the overall performance in most cases. For example, on the task of semantic type detection over WikiTable when the sampling rate is 2%, the introduction of MT, PS, and BL can bring 1, 5.95, and 13.51 improvement in Macro F_1 score, respectively. Another finding is that the PS technique plays a significant role in improving the overall performance under low-resource settings. For instance, on relation extraction where data sparsity is the most serious, PS could bring 25.94 and 11.94 improvement in Micro F_1 score when the sampling rate is 2% and 5%, respectively. The reason might be it can help produce more labeled instances by leveraging the unlabeled instances to provide richer signals for training. Besides, it is also essential to introduce the BL in this case so as to feed the model with more labeled instances from minority classes. Take the relation extraction task with sampling rate 2% as an example again, when the BL can help improve the Micro and Macro F_1 score by 31.86 and 16.18, respectively. Finally, the improvement is not so significant on the task of semantic type detection over the VizNet dataset. The reason could be that the data sparsity problem is relatively moderate, and thus, the performance of all methods increases stably along with the sampling rate.

We also report the results on the balanced sampled dataset in Table 8. We can see that the results show similar trends with those in Table 7. The performance is generally worse because the number of involved training instances is smaller. One observation is that the benefit brought by the BL technique is relatively smaller. The reason is that the balanced sampling strategy tends to include more instances from the minority classes. Under the setting of $X = 100$, the instances from some minority classes will be almost fully covered. This could also be observed from the result of Macro F_1 score: when X equals 20 and 50, the cardinality of training data is smaller than that when the sampling rate is 2% and 5% in most cases (Table 4). The Micro F_1 is generally lower in this case, but

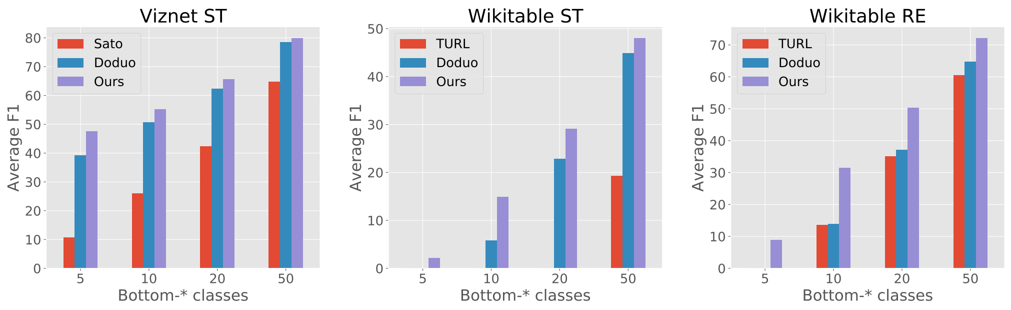

Fig. 5. Average class F1 values of bottom classes for each method. Left: column type annotation on VizNet (full), middle: column type annotation on WikiTable (full), right: relation extraction on WikiTable (full).

the Macro F_1 is higher. When X reaches 100, the performance is closer to that when the sampling rate is 10%.

5.4 Further Exploration

5.4.1 Pre-trained LM for Tabular Data. Firstly, we make an initial exploration of applying the contrastive learning techniques on pre-trained LM for tabular data instead of BERT variants. Specifically, we choose TaBERT [55] as the cornerstone and perform contrastive learning techniques proposed in Section 3. Then we perform fine-tuning for the semantic type detection and relation extraction tasks in the same way as the above experiments. The experimental results are shown in Table 9. The performance gain brought by contrastive learning for TaBERT is somewhat limited compared with that of BERT variants. The reason could be that, unlike the original BERT, TaBERT has obtained enough knowledge about table structure from pre-training; meanwhile, the corpus for pre-training TaBERT is much larger than for contrastive learning (26M vs. 571K). TaBERT is also optimized for learning the representation of both natural language utterances and the table, not only for column representations. Thus, the benefit of our current contrastive learning strategies would be limited for TaBERT. However, this comparison showcases the efficacy of our lightweight framework as Watchog achieves better performance than TaBERT while avoiding the heavy pre-training. How to devise contrastive learning techniques based on pre-trained LM for tabular data could be an interesting problem for future work.

Table 9. Results of Contrastive Learning over TaBERT.

Task	Method	Micro F1	Macro F1
WikiTable ST	TaBERT_Base (K=3)	91.23	70.35
	TaBERT_Base (K=3) + CL	91.34	70.65
WikiTable RE	TaBERT_Base (K=3)	90.84	79.96
	TaBERT_Base (K=3) + CL	91.95	82.32
VizNet ST	TaBERT_Base (K=3)	93.24	82.95
	TaBERT_Base (K=3) + CL	93.13	83.16

5.4.2 Results on GitTables Corpus. Next, we show more results of the semantic type detection task on another large table corpus GitTables [21] of relational tables. Since the corpus itself does not provide tasks with ground truth, we use the benchmark datasets in SemTab 2021 challenge [4] and SemTab 2022 challenge [5], which are generated from the GitTables corpus. For SemTab 2021, we report the results of 5-fold cross-validation since the original dataset does not provide a split of train/validation/test data. Each challenge involves different sub-tasks where the labels were extracted from different ontologies (DBpedia and Schema.org).

The results on the SemTab 2021 and SemTab 2022 benchmark are shown in Table 10 and Table 11. We can see that Watchog shows clear advantages over Doduo on both tasks. Moreover, the best method for these tasks from SemTab 2022 is KGCODE-Tab [29], which achieved 59, 69 and 62 in Micro F_1 on the DBpedia (SemTab 2022), Schema (SemTab 2022 class) and Schema (SemTab 2022 property) tasks, respectively. This performance is obviously worse than Watchog. We also tried to conduct contrastive learning on Gittable corpus (denoted by (GT)) instead of wiki table one (denoted by (WT)) shown in Table 11. The performance is slightly better due to the rich knowledge obtained from the large corpus. This result further shows the potential benefits of our proposed techniques.

Table 10. Results on GitTables SemTab 21 Semantic Type Annotation Tasks

Task	DBpedia		Schema	
Method	Micro-f1	Marco-f1	Micro-f1	Marco-f1
Doduo	66.83	29.22	78.92	45.10
Watchog (WT)	87.76	47.78	87.11	56.73
Watchog (GT)	87.84	49.70	87.38	60.06

Table 11. Results on GitTables SemTab 22 Semantic Type Annotation Tasks

Task	DBpedia		Schema (class)		Schema (property)	
method	Micro-f1	Marco-f1	Micro-f1	Marco-f1	Micro-f1	Marco-f1
KGCODE-Tab [29]	59	-	69	-	62	-
Doduo	55.05	34.51	62.44	50.75	65.22	41.91
Watchog (WT)	59.44	39.67	69.56	58.09	70.66	49.44
Watchog (GT)	61.01	39.52	70.90	58.85	70.98	48.35

5.5 Case Study

In this section, we propose two case studies to show the potential value of our proposed framework in practice.

5.5.1 Tail analysis. The first case study shows the breakdown performance of Watchog on minority classes belonging to the "long tail" under the supervised setting where the whole training set is used in the fine-tuning process. For example, Table 12 presents the F1 values of the bottom-5 classes by the baseline SATO [59]. Empowered by pre-trained LMs, both Doduo and Watchog significantly outperform SATO. Meanwhile, Watchog further outperforms Doduo by a large margin. Specifically, Watchog performs better on 4 out of 5 classes; the average improvement in F_1 score is more than 10.3. For the WikiTable dataset, as there are more classes (255 for ST and 121 for RE), the long-tail

[4]https://zenodo.org/record/5706316
[5]https://sem-tab-challenge.github.io/2022/

problem is more obvious. Table 13 shows that TURL performs poorly on many classes: 20 with 0 F_1 score and 74 with F_1 score lower than 0.6 for ST; 16 classes with F_1 score lower than 0.6 for RE. At the same time, Watchog has the fewest number of classes with such low F_1 scores.

To clearly illustrate the improvement of Watchog, we further provide the average F_1 score for each method's bottom classes in Figure 5. That is, we sort the F_1 score of each class for every compared method and take the bottom ones, where the set of bottom classes for each method might differ. For all tasks, Watchog consistently outperforms other methods. Specifically, the advantage of Watchog over TURL remains significant when the number of classes increases for WikiTable ST, and that over Doduo is still tangible even for bottom-50 classes. Therefore, Watchog can improve the long-tail performance for all table understanding tasks with the help of contrastive learning. The improvement in Macro F_1 under semi-supervised settings of Watchog also validates this conclusion.

Table 12. Class F1 values of the bottom-5 classes for column type annotation on the VizNet (full) dataset.

type	ranking	director	person	affiliate	command
SATO	3.08	3.33	13.99	32.73	46.41
Doduo	25.65	41.06	33.64	45.71	57.30
Watchog	34.42	40.50	44.63	63.33	61.08

Table 13. Number of classes with low F1 value on the WikiTable dataset.

Task	Column type annotation			Relation extraction		
Method	TURL	Doduo	Watchog	TURL	Doduo	Watchog
f1 = 0	20	7	4	7	6	5
f1 < 0.6	74	36	33	16	14	10

Table 14. Case study on column population. Our model only takes the seed header and seed column content as input. The comments are shown here for better understanding and are not visible to the model.

Seed header	Seed column content	Target	Predicted	AP	Comment
name	[Smooth FM 89.9, Killer Bee Cebu, i FM Sigu-radong Enjoy Ka!, Energy FM Cebu, ...]	company, format, call sign, covered location	covered location, call sign, format, company, genre	1	table of list of radio stations in the Philippines
building	[Beijing Station, Ching Fu Shipbuilding, Cocoon, Alto Vetro Tower, ...]	location, architect, nationality	architect, location, city, nationality, country	0.92	table of RIBA International award winners in 2009

5.5.2 Column population. Besides table understanding tasks, we demonstrate that our method can also be applied to table augmentation tasks. Specifically, we showcase the column population task, which aims to discover new columns from one seed column to widen the table. We use the WikiTable dataset and follow the processing procedure in TURL [9] (with 316,858 training tables and 4,646 test tables) while formulating the task as a multi-label classification task with 5,407 possible column headers. Note that we only rely on column headers and exclude other meta-data information, such as table captions. Instead, our model takes a subset of the contents (cells in the first column) as input. Specifically, we got the results of standard metrics as follows: mean average precision (MAP) as 79.88, mean reciprocal rank (MRR) as 83.63, and normalized discounted cumulative gain (NDCG@10) as 82.94, which illustrates that Watchog can also support this task.

Table 14 shows two examples of how Watchog works for this task. Here we show the average precision (AP) for each example. In the first example, Watchog correctly predicts all column headers related to the header of radio station names. In the second example, although the precision is not 100%, Watchog retrieves all 3 ground truth headers in the top-4, and the other returned headers (city, country) are close to the ground truth (location, nationality).

6 RELATED WORK

6.1 Table Understanding

The table understanding tasks originated from the table annotation [32] application that aims at annotating cells with entities (from an ontology) that the cell mentions, columns with ontology types, and pairs of columns with ontology relationships. Earlier studies [37, 39] relied on hand-crafted features to capture the semantics in tabular data and employed techniques like similarity join based on syntactic similarity [48, 51] to identify the results. Sherlock [22] extracted features from different granularities and then employed machine learning based approaches to predict the semantic types for columns in web tables. SATO [59] made a further improvement on the basis of it by incorporating topic models. Recently deep learning techniques have been widely applied in table understanding applications. TCN [45] and Leva [62] utilized graph-based models to learn the inter-table information to enhance the table representation learning. TURL [9] first employed pre-trained LM to solve the problem of table understanding and provided benchmarking datasets for 6 different tasks. Doduo [42] improved the performance of fine-tuning pre-trained LMs by introducing multi-task learning.

6.2 Pre-trained Language Models

Pre-trained LMs have been a hot topic in natural language processing since the BERT model became popular. Some efforts are made to improve the BERT model, such as reducing the model size [27, 40], handling longer input sequence [26, 38] and improving the pre-training task [33]. Some recent studies utilized Pre-trained LMs for the application of entity matching [30, 46, 47, 57], which requires to learn the embedding of rows instead of columns in tables. Many previous studies aimed at pre-training a language model for tabular data that can be applied to several downstream tasks such as semantic type detection, row/column population, and question answering over web tables. TaBERT [55] defined novel pre-training tasks for tabular data by slicing the table via user queries. Tabbie [23] and RCI [16] made further improvement based on it by incorporating both row and column wise information. Tapas [19] and TableFormer [54] improved the pre-training step by introducing additional encoding mechanisms. TabularNet [11] and TUTA [50] focused on tables with complicated structures. All these works focus on the heavy weighted pre-training step and are orthogonal to our work.

6.3 Contrastive Learning

Contrastive learning is a popular machine learning paradigm in self-supervised representation learning for many applications. It has two essential components: contrastive data creation and contrastive objective optimization [8]. Recently the machine learning community has made a huge amount of efforts to improve either or both aspects of contrastive learning [7, 18, 24, 56]. Contrastive learning has also been widely applied in NLP applications to train sentence representations in an unsupervised manner to benefit one or several downstream applications [14, 15, 53]. Starmie [13] also utilized contrastive learning for training a column encoder. However, Starmie focused on problems that have totally different settings from those in our paper: It aims at table search tasks that are fully unsupervised, while our work focuses on column annotation tasks under supervised and semi-supervised settings. Sudowoodo [49] employed contrastive learning for entity matching. Its proposed techniques mainly targeted on the entity matching application and thus cannot be direclty applied to our problem. To the best of our knowledge, our work is the first to employ contrastive learning techniques for column annotation.

7 CONCLUSION

In this paper, we proposed to employ contrastive learning techniques to facilitate column annotation tasks. To this end, we developed a novel and unified framework, namely Watchog, that enables learning high-quality table representation models. These models could be applied to various downstream applications with significantly fewer labeled instances. Unlike language models pre-trained for tabular data, Watchog introduced much less overhead in training time while remaining highly versatile. Moreover, we presented the first optimization technique for table understanding under semi-supervised settings, which builds on the representation model learned from contrastive learning. As a result, Watchog could deliver promising results in the fine-tuning process with limited labeled instances. Experimental results on several tasks demonstrated that Watchog achieves state-of-the-art performance, particularly in semi-supervised settings. Furthermore, we provided case studies that showcase how Watchog can mitigate the long-tailed issue and enhance the performance of various real-world applications. For future work, we plan to explore how to support other table understanding tasks, such as row population, cell filling, and schema augmentation, with similar methodology. Besides, we will also aim at devising specified contrastive learning techniques for pre-trained LM for tabular data. Finally, we plan to create new benchmarking datasets for the relation extraction task to comprehensively evaluate different solutions.

ACKNOWLEDGMENT

We would like to thank Renée J. Miller, Estevam Hruschka and Dan Zhang for their valuable comments on this work.

REFERENCES

[1] Philip Bachman, Ouais Alsharif, and Doina Precup. 2014. Learning with Pseudo-Ensembles. In *Advances in Neural Information Processing*. 3365–3373.

[2] Gilbert Badaro and Paolo Papotti. 2022. Transformers for Tabular Data Representation: A Tutorial on Models and Applications. *Proc. VLDB Endow.* 15, 12 (2022), 3746–3749.

[3] Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Transformers for Tabular Data Representation: A Survey of Models and Applications. *Transactions of the Association for Computational Linguistics* 11 (2023), 227–249.

[4] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *NeurIPS*. 5050–5060.

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

[6] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019. Learning Semantic Annotations for Tabular Data. In *IJCAI*. 2088–2094.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, Vol. 119. 1597–1607.

[8] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. Big Self-Supervised Models are Strong Semi-Supervised Learners. In *NeurIPS*.

[9] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *Proc. VLDB Endow.* 14, 3 (2020), 307–319.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[11] Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei Zhang. 2021. TabularNet: A Neural Network Architecture for Understanding Semantic Structures of Tabular Data. In *ACM SIGKDD*. 322–331.

[12] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *SIGMOD Companion*. 69–75.

[13] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.* 16, 7 (2023), 1726–1739.

[14] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP*. 6894–6910.

[15] John M. Giorgi, Osvald Nitski, Bo Wang, and Gary D. Bader. 2021. DeCLUTR: Deep Contrastive Learning for Unsupervised Textual Representations. In *ACL/IJCNLP*. 879–895.

[16] Michael R. Glass, Mustafa Canim, Alfio Gliozzo, Saneem A. Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing Row and Column Semantics in Transformer Based Question Answering over Tables. In *NAACL-HLT*. 1212–1224.

[17] Lan-Zhe Guo and Yu-Feng Li. 2022. Class-Imbalanced Semi-Supervised Learning with Adaptive Thresholding. In *ICML*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.), Vol. 162. PMLR, 8082–8094.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[19] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *ACL*. 4320–4333.

[20] 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 *CHI*. ACM, 662.

[21] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. GitTables: A Large-Scale Corpus of Relational Tables. *Proc. ACM Manag. Data* 1, 1 (2023), 30:1–30:17.

[22] Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César A. Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *SIGKDD*. 1500–1508.

[23] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained Representations of Tabular Data. In *NAACL-HLT*. 3446–3456.

[24] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. In *NeurIPS*.

[25] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.).

[26] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *ICLR*.

[27] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[28] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A Large Public Corpus of Web Tables containing Time and Context Metadata. In *WWW*. ACM, 75–76.

[29] Xinhe Li, Shuxin Wang, Wei Zhou, Gongrui Zhang, Chenghuan Jiang, Tianyu Hong, and Peng Wang. 2022. KGCODE-Tab Results for SemTab 2022. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), CEUR-WS.org*, Vol. 3320. 37–44.

[30] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, Jin Wang, Wataru Hirota, and Wang-Chiew Tan. 2021. Deep Entity Matching: Challenges and Opportunities. *ACM J. Data Inf. Qual.* 13, 1 (2021), 1:1–1:17.

[31] Yuliang Li, Xiaolan Wang, Zhengjie Miao, and Wang-Chiew Tan. 2021. Data Augmentation for ML-driven Data Preparation and Integration. *Proc. VLDB Endow.* 14, 12 (2021), 3182–3185.

[32] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow.* 3, 1 (2010), 1338–1347.

[33] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).

[34] Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A Meta-Learned Data Augmentation Framework for Entity Matching, Data Cleaning, Text Classification, and Beyond. In *SIGMOD*. 1303–1316.

[35] Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. Snippext: Semi-supervised Opinion Mining with Augmented Data. In *WWW*. 617–628.

[36] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*. 2227–2237.

[37] Minh Pham, Suresh Alse, Craig A. Knoblock, and Pedro A. Szekely. 2016. Semantic Labeling: A Domain-Independent Approach. In *ISWC*, Vol. 9981. 446–462.

[38] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive Transformers for Long-Range Sequence Modelling. In *ICLR*.

[39] S. K. Ramnandan, Amol Mittal, Craig A. Knoblock, and Pedro A. Szekely. 2015. Assigning Semantic Labels to Data Sources. In *ESWC*, Vol. 9088. 403–417.

[40] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019).

[41] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[42] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çagatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating Columns with Pre-trained Language Models. In *SIGMOD*. 1493–1503.

[43] Saravanan Thirumuruganathan, Nan Tang, Mourad Ouzzani, and AnHai Doan. 2020. Data Curation with Deep Learning. In *EDBT*. 277–286.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

[45] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. TCN: Table Convolutional Network for Web Table Interpretation. In *WWW*. 4020–4032.

[46] Jin Wang and Yuliang Li. 2022. Minun: evaluating counterfactual explanations for entity matching. In *DEEM '22: Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*. 7:1–7:11.

[47] Jin Wang, Yuliang Li, Wataru Hirota, and Eser Kandogan. 2022. Machop: an end-to-end generalized entity matching framework. In *aiDM '22: Proceedings of the Fifth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 2:1–2:10.

[48] Jin Wang, Chunbin Lin, Mingda Li, and Carlo Zaniolo. 2020. Boosting approximate dictionary-based entity extraction with synonyms. *Inf. Sci.* 530 (2020), 1–21.

[49] Runhui Wang, Yuliang Li, and Jin Wang. 2023. Sudowoodo: Contrastive Self-supervised Learning for Multi-purpose Data Integration and Preparation. In *ICDE*. 1502–1515.

[50] 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 *ACM SIGKDD*. 1780–1790.

[51] Jiacheng Wu, Yong Zhang, Jin Wang, Chunbin Lin, Yingjia Fu, and Chunxiao Xing. 2019. Scalable Metric Similarity Join Using MapReduce. In *ICDE*. 1662–1665.

[52] Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised Data Augmentation for Consistency Training. In *NeurIPS*.

[53] Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer. In *ACL/IJCNLP*. 5065–5075.

[54] Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust Transformer Modeling for Table-Text Encoding. In *ACL*. 528–537.

[55] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *ACL*. 8413–8426.

[56] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *ICML*, Vol. 139. 12310–12320.

[57] Alexandros Zeakis, George Papadakis, Dimitrios Skoutas, and Manolis Koubarakis. 2023. Pre-trained Embeddings for Entity Resolution: An Experimental Analysis. *Proc. VLDB Endow.* 16, 9 (2023), 2225–2238.

[58] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. 2021. FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling. In *Advances in Neural Information Processing Systems (NeurIPS)*. 18408–18419.

[59] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çagatay Demiralp, and Wang-Chiew Tan. 2020. Sato: Contextual Semantic Type Detection in Tables. *Proc. VLDB Endow.* 13, 11 (2020), 1835–1848.

[60] Shuo Zhang and Krisztian Balog. 2017. EntiTables: Smart Assistance for Entity-Focused Tables. In *SIGIR*. 255–264.

[61] Shuo Zhang and Krisztian Balog. 2019. Auto-completion for Data Cells in Relational Tables. In *CIKM*. ACM, 761–770.

[62] Zixuan Zhao and Raul Castro Fernandez. 2022. Leva: Boosting Machine Learning Performance with Relational Embedding Data Augmentation. In *SIGMOD*. 1504–1517.

Received April 2023; revised July 2023; accepted September 2023