TCN: Table Convolutional Network for Web Table Interpretation

Daheng Wang*
University of Notre Dame, Notre Dame, IN 46556, USA
dwang8@nd.edu

Prashant Shiralkar
Amazon.com, Seattle, WA 98109, USA
shiralp@amazon.com

Colin Lockard
Amazon.com, Seattle, WA 98109, USA
clockard@amazon.com

Binxuan Huang
Amazon.com, Seattle, WA 98109, USA
binxuan@amazon.com

Xin Luna Dong
Amazon.com, Seattle, WA 98109, USA
lunadong@amazon.com

Meng Jiang
University of Notre Dame, Notre Dame, IN 46556, USA
mjiang2@nd.edu

ABSTRACT

Information extraction from semi-structured webpages provides valuable long-tailed facts for augmenting knowledge graph. Relational Web tables are a critical component containing additional entities and attributes of rich and diverse knowledge. However, extracting knowledge from relational tables is challenging because of sparse contextual information. Existing work linearize table cells and heavily rely on modifying deep language models such as BERT which only captures related cells information in the same table. In this work, we propose a novel relational table representation learning approach considering both the intra- and inter-table contextual information. On one hand, the proposed Table Convolutional Network model employs the attention mechanism to adaptively focus on the most informative intra-table cells of the same row or column; and, on the other hand, it aggregates inter-table contextual information from various types of implicit connections between cells across different tables. Specifically, we propose three novel aggregation modules for (i) cells of the same value, (ii) cells of the same schema position, and (iii) cells linked to the same page topic. We further devise a supervised multi-task training objective for jointly predicting column type and pairwise column relation, as well as a table cell recovery objective for pre-training. Experiments on real Web table datasets demonstrate our method can outperform competitive baselines by +4.8% of F1 for column type prediction and by +4.1% of F1 for pairwise column relation prediction.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Web table, information extraction, knowledge extraction

ACM Reference Format:
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 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3442381.3450090

*Most of the work was conducted when the author was interning at Amazon

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450090

1 INTRODUCTION

In recent years, there has been a significant thrust both in academia and industry toward the creation of large knowledge bases (KB) that can power intelligent applications such as question answering, personal assistant, and recommendation. These knowledge bases (e.g., DBpedia [20], Wikidata [44], Freebase [2]) contain facts about real-world entities such as people, organizations, etc., from a variety of domains and languages in the form of (subject, predicate, object) triples. The field of Information Extraction (IE) aims at populating these KBs by extracting facts from websites on the Web. The information on the Web can be roughly categorized into four types, namely, unstructured text, semi-structured data, Web tables and semantic annotations [10]. Recently with the advances in natural language processing (NLP), there has been significant progress in the development of effective extraction techniques for the text and semi-structured data [16, 26–28]. However, we have seen limited success to transform the next rich information source, Web tables, into triples that can augment a knowledge base [49].

A Web table is a tabular structure embedded within a webpage displaying information about entities, their attributes and relationships with other entities along rows and columns. It contains high quality relational knowledge and differs from other types of tables such as layout tables, which are primarily meant for formatting purposes, or matrix tables, meant to show numerical summaries in a grid format. Because they contain metadata such as table caption and column headers that mimic tables in a relational database, they are also known as relational Web tables. An example of such table from the detail page of musical artist "Elton John" on discogs.com is shown in Figure 1. This table shows discography information about the artist (the main topic entity of the page) such as albums in which he has performed along the rows, their release date and their publishers, along with other details on the page such as biographical information (e.g., real name, biography, alternate names, etc.) displayed in a key-value format. Although such relational tables are ubiquitous on the Web (a 2016 study estimates a total of 233M tables on the Web [22]), they are particularly prevalent on semi-structured websites such as discogs.com which are known to be very rich sources of information in a variety of domains [25]. These sites contain detail pages of different types, e.g., artist, album, track and publisher pages. Since these websites are created by populating

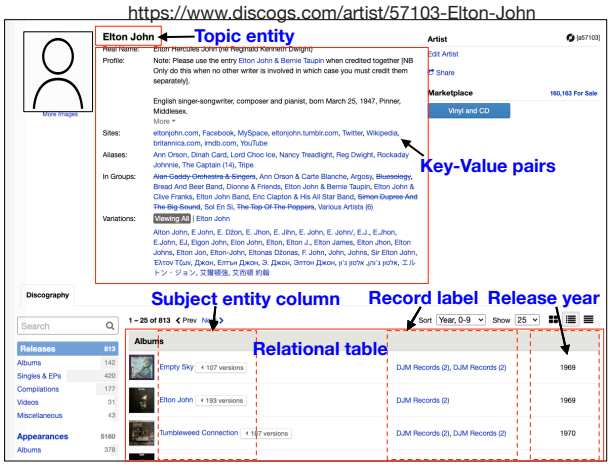


Figure 1: An example page of Elton John from discogs.com showing demographic as key-value pairs and a Web table of discography details. Image is redacted for privacy reasons.

HTML templates from an underlying relational database, there are millions of such pages with embedded Web tables, making them particularly attractive for knowledge base enrichment.

Our goal is to develop effective methods for extracting and interpreting information in Web tables to augment a knowledge base. In this paper, we focus on the task of Web table interpretation, while relying on existing work to perform the task of table extraction, entailing detecting and filtering of tables from the pages [5]. The task is aligning the schema of a given table to the ontology of a knowledge base. It involves determining the type of each column and the relation between columns from a fixed set of types and relations in the ontology so that tabular entries can be extracted as triples to augment the KB. It is also known as the metadata recovery problem in literature [5]. However, such alignment is challenging for two reasons, namely *schema heterogeneity* and *context limitation*. The problem of schema heterogeneity arises because tables from different websites use different terminology for table caption and column headers. For example, one website might use "Name" while another website might use "Label" as the header of a column containing publisher names. Besides, the caption, header and/or tabular cell entry may be missing altogether. The second challenge is that the information in the table cells is often very short, typically consisting of only a few words, and thus lacks adequate context to perform any effective reasoning using off-the-shelf NLP methods, thereby requiring a different approach for table interpretation.

Although the work on Web table interpretation began over a decade ago [3], the success has been limited. Early work employed probabilistic graphical models to capture the joint relationship between rows, columns and table header, but suffered from low precision (~65%) [23]. Most recent work instead ignore the tabular structure and attempt to leverage the power of rich language models such as BERT by applying them to information in the table after transforming it into a long, linear sequence of cell values [47]. However, this approach has the natural downside of resulting in loss of information implicit in the tabular structure, for example, the fact that the values in a column belong to the same type and values along a row belong to the same entity. These implicit connections between cell entries along rows and columns of a table can serve as important context for prediction methods for the two tasks.

Besides this intra-table context, the tables when viewed as a collection can provide two additional useful *inter-table contexts*. They are (a) shared-schema context: information shared between tables following the same schema from different pages of the same web source. For example, tables from other artist pages of discogs.com contain some common values such as publisher names producing albums for various artists, and (b) across-schema context: information shared between tables from multiple sources in the same domain, for example, tables from artist pages on both discogs.com and musicbrainz.org may show the same publisher names for an album. None of the existing approaches consider the opportunity to leverage this inter-tabular context for designing models for the two tasks. In this paper, we answer the question: how can we leverage both the intra-table and inter-table contexts to improve web table interpretation, especially in the presence of shared table schema and overlapping values across webpages and even across websites?

Different from the existing setting, we consider a collection of tables as the input to our problem to utilize the full context available from both intra-table and inter-table implicit connections. We view the collection of tables as a graph in which the nodes represent the tabular cell entries and the edges represent the implicit connections between them. An edge may connect two values from the same row, same column, same cell position in tables following a common schema, or any position in tables having the same value. Given the fact that each node (a value in our case) can have a variable number of links, and inspired by the capabilities of a graph neural network (GNN) to learn effectively from such graph data [45, 46], our goal is to learn a table representation that makes use of the intra-table and inter-table contexts for learning a prediction model for the column type and relation prediction tasks. We propose a novel relational table representation learning framework called Table Convolution Network (TCN) that operates on implicitly connected relational Web tables and aggregates information from the available context.

Our approach gives us two main advantages: (a) it allows us to integrate information from multiple Web tables that provide greater context, and (b) when the inter-table context is unavailable, our model reduces to common case of having only one table as the input, thereby unifying the general case of Web tables. To train the network efficiently, we propose two training schemes: (a) supervised multi-tasking, and (b) unsupervised by way of pre-training for scenarios when the supervision may not be available.

We make the following contributions through this paper:

- We propose a novel representation learning framework called Table Convolution Networks (TCN) for the problem of Web table interpretation involving column type and pairwise column relation prediction. At its core, TCN utilizes the intra-table and inter-table context available from a collection of tables, instead of the limited context from a single table.
- We show two approaches to train the network, namely a classic supervised mode, and an unsupervised mode that employs pre-training through self-supervision for jointly predicting column type and relation between column pairs.
- We perform extensive experiments with several state-of-the-art baselines on two datasets containing 128K tables of 2.5M triples, showing that TCN outperforms all of them with an

F1 score of 93.8%, a relative improvement of 4.8% points on the column type detection task, and an F1 score of 93.3%, a relative improvement of 4.1% on the relation prediction task.

The roadmap of this paper is organized as follows. We review related work in Section 2. In Section 3, we formally define the research problem. Our proposed Table Convolutional Network approach is introduced in Section 4. Section 5 presents experimental results. We conclude the paper and discuss on future work in Section 6.

2 RELATED WORK

We discuss two lines of research related to our work.

Relational Table Interpretation. Relational tables on the Web describe a set of entities with their attributes and have been widely used as vehicle for conveying complex relational information [4, 13]. Since the relationships of table cells are not explicitly expressed, relational table interpretation aims at discovering the semantics of the data contained in relational tables, with the goal of transforming them into knowledge intelligently processable by machines [3, 48]. With the help from existing knowledge bases, this is accomplished by first classifying tables according to some taxonomy [15, 34], then identifying what table columns are about and uncovering the binary relation of table columns [50]. The extracted knowledge can in turn be readily used for augmenting knowledge bases [37].

Column type annotation refers to associating a relational table column with the type of entities it contains. Earlier methods combine the exact match or certain entity search strategies with a majority vote scheme for predicting the column type [31, 43]. Fan et al. [12] proposed a two-stage method which first matches column to candidate concepts and then employ crowdsourcing for refinement on type prediction. T2KMatch by Lehmberg et al. [21] proposed to stitch Web tables of the same schema into a larger one to improve the prediction performance. Sherlock [18] by Hulsebos et al. proposed a set of statistical features describing the character and word level distributions along with some semantic features for feeding into a deep classifier to get high prediction accuracy.

Relation extraction is the task of associating a pair of columns in a table with the relation that holds between their contents. Mulwad et al. [30] proposed a semantic message passing algorithm using knowledge from the linked open data cloud to infer the semantics between table columns. Munoz et al. [32] proposed to use an existing linked data knowledge base to find known pre-existing relations between entities and extend on analogous table columns. Sekhavat et al. [38] proposed a probabilistic model leveraging natural language patterns associated with relations in knowledge base.

Another common task for interpreting relational table is entity linking which is the process of detecting and disambiguating specific entities mentioned in the table [1, 11]. Existing work often couple it with column type annotation and relation extraction together as a prerequisite or joint task [31, 36, 51]. However, this requires expensive preprocessing steps and largely limits the flexibility for interpreting semantics of relational table [6]. In this work, we do not assume the availability of any pre-linked entities, and focus on the tasks of column type annotation and pairwise column relation extraction completely based on the table cell contents.

Representation Learning of Tabular Data. Earlier work utilized probabilistic models to capture dependencies between table cells. Limaye et al. [23] proposed to model the entity, type and relation information of table cells as random variables, and jointly learn their distributions by defining a set of potential functions. A Markov Random Fields model was proposed by Ibrahim et al. [19] for canonicalizing table headers and cells into concepts and entities with a special consideration on numerical cell values of quantities. Meimei by Takeoka et al. [41] proposed to incorporate multi-label classifiers in the probabilistic model to support versatile types of cell and improve predictive preformance. These methods have high complexity due to MCMC sampling and they cannot be directly applied on large-scale relational Web tables.

Some studies made efforts in table representations learning by leveraging the word embedding model word2vec [29]. Table2Vec by Zhang et al. [49] proposed to linearize a cropped portion of the table's grid structure into sequence of cell tokens as the input of word2vec. This treatment was also adopted by Gentile et al. [14] for blocking to reduce the efficiency of entity matching. Record2Vec by Sim et al. [39] transformed structured records into attribute sequence and combined word2vec with a tailored triplet loss. However, shallow neural models like word2vec have relatively limited expressiveness which pose difficulties on effectively capturing the semantics of relational tables.

Recent methods utilize deep neural language model for learning table representations. TURL by Deng et al. [8] proposed a pre-training/finetuning framework for relational Web tables by injecting visibility matrix into the encoder of Transformer [42] to attend on structurally related table components. The authors also proposed a Masked Entity Recovery objective to enhance the learning capability but this requires pre-linked entities of table cells as model input which is not available in most real cases. TaPas by Herzig et al. [17] proposed to jointly learn the embedding of natural language questions over relational tables by extending the BERT [9] model with more table-aware positional embeddings. TaBERT by Yin et al. [47] adopted a similar idea for semantic parsing on database tables combining with Masked Column Prediction and Cell Value Recovery as two additional unsupervised objectives for pre-training. One major limitation of these methods is they only focus on aggregating components of a single table via indirect techniques such as visibility matrix and content snapshot [7, 33, 40]. In contrast, we propose to directly capture intra-table context by attending on the column and rows cells with easy integration of inter-table contexts. And, we fully consider the highly valuable inter-table contextual information by aggregating various types of implicit connections between tables of same cell value or position.

3 PROBLEM DEFINITION

Given a collection of webpages, perhaps from semi-structured websites, with relational Web tables embedded in them, and our goal is to uncover the semantics of the columns by annotating them with their type and determining the relation between pairs of columns. A Web table in such pages can be schematically understood as a grid comprising of rows and columns. Each row contains information about a single real-world entity (e.g., an album name), typically found in one of the first few columns, and its attributes and relationships with other entities in other columns (e.g., release year and publisher), and each column contains attributes or entities of

Table 1: Symbols and their descriptions.

Symbol	Description
T_k	a relational Web table
$t^{m,n}$ ($t_k^{m,n}$)	table cell of T_k locates at the intersection of the m-th row and the n-th column
$t^{m,*}$, $t^{*,n}$	the m-th row, and the n-th column of T_k
S_k^r, S_k^c	T_k's number of rows, and number of columns
ϕ	the table schema mapping function
p_k	T_k's page topic of short text
$\mathcal{D}$	a dataset of relational tables
K, U	number of relational tables, and number of unique table schema in $\mathcal{D}$
$\mathcal{C}$, $\mathcal{R}$	set of target column types, and set of target relations between subject and object columns
$\mathbf{e}_{t^{m,n}}$	initial embedding vector of table cell $t^{m,n}$
$\mathbf{e}^c_{t^{m,n}}$, $\mathbf{e}^r_{t^{m,n}}$	the column-wise, and row-wise aggregated context vectors of target cell $t^{m,n}$
AGG_a	the intra-table aggregation function
$\mathbf{e}^a_{t^{m,n}}$	the intra-table contextual embedding of $t^{m,n}$
$\mathcal{N}_v$, $\mathcal{N}_s$, $\mathcal{N}_p$	set of value cells, set of position cells, and set of and topic cells
$\mathbf{e}^v_{t^{m,n}}$, $\mathbf{e}^s_{t^{m,n}}$ $\mathbf{e}^p_{t^{m,n}}$	aggregated inter-table contextual embeddings of $\mathcal{N}_v(t^{m,n})$, $\mathcal{N}_s(t^{m,n})$, and $\mathcal{N}_p(t^{m,n})$
D, $\mathbf{W}$	dimension of vector, and matrix of parameters

the same type described by an optional column header. Taking the table in Figure 1 as an example, the first column contains the album entities being described, while the rest of the columns indicate its attributes and relationships. We call this column the *subject* column to indicate it is the subject of the rows. Moreover, we can infer "DJM Records" to be the publisher of "Empty Sky" due to the fact that their association is found in the same row, and likewise, we can inter "Empty Sky" to be a "Release" (a technical term for album) by knowing other values in the same column and due to presence of the header "Album". In a set of K relational Web tables, we denote the k-th table T_k ($k = 1, \ldots, K$) as a set of row tuples, i.e., $T_k := \{(t_k^{0,0}, t_k^{0,1}, \ldots, t_k^{0,S_k^c}), \ldots, (t_k^{S_k^r,0}, t_k^{S_k^r,1}, \ldots, t_k^{S_k^r,S_k^c})\}$ where S_k^r and S_k^c are the number of rows and columns of the k-th table. The first row $t^{0,*}$ typically contains the table header (e.g., "Title of Track" and "Name of Composer"). When the context is clear, we omit the subscript and use $t^{m,n}$ to denote the cell at the intersection of m-th row ($0 \le m \le S_k^r$) and n-th column ($0 \le n \le S_k^c$) of table T_k. We use $t^{m,*}$ and $t^{*,n}$ to denote all cells at the m-th row and the n-th column of the table respectively, i.e., $t^{m,*} := (t^{m,0}, t^{m,1}, \ldots, t^{m,S_k^c})$ and $t^{*,n} := (t^{0,n}, t^{1,n}, \ldots, t^{S_k^r,n})$.

A Web table has additional contexts that further describe its semantics and therefore should be leveraged for better table interpretation. They come in two forms: metadata from the page of the table, namely an optional table caption and the `<title>` tag or topic entity of the webpage, and an inter-table context available by considering the collection of tables that either conform to the same underlying schema by virtue of belonging to the same HTML template, or describe similar information from different sites in the same domain. Being present on the detail page of "Elton John" (the topic entity in our example), the table conveys the semantic connection between him and the set of albums in the table, thereby providing additional context for the cell values.

Furthermore, there also exists greater context by considering the implicit connections between tables as a collection. We assume pages of a website have been segregated to form groups that correspond to distinct HTML templates conforming to different page types (i.e., schema) thus each table belongs to a single unknown schema. Accordingly, cell entries in the same position across tables of the same schema provide evidence of belonging to the same type. This can be particularly useful when one of the tables is sparse and difficult to reason about. We can also view the connections between same values from different tables, perhaps from entirely disparate schema, to also provide additional useful information, e.g., one table from site A might describe a publisher's information with artists, while another table from site B might describe its information with bands. Such inter-tabular context can be valuable to discern subtle differences for the task of column type and relation prediction.

We use $\phi : \{1, \ldots, K\} \rightarrow \{1, \ldots, U\}$ to denote a table schema mapping function where U is the number of unique schema. Relational tables of the same schema $\{T_k \mid \phi(k) = u, 1 \le u \le U\}$ have the same header cells and generally U is a large value comparable to K. In practice, when relational tables are from semi-structured websites focusing around certain page topics, the value of U could be much smaller than K, i.e., $U << K$. Besides, table cells of the same column can be characterized by the the common known entity type (e.g., "Release" for tracks and "People" for composers). Assuming the first column $t^{*,0}$ contains all subject entities (e.g., various tracks of an album) we denote it as the *subject* column, and all other columns of the table $t^{*,n}$ ($1 \le n \le S_k^c$) contain attribute information about the first column, we denote them as the *object* columns. Each relational table T_k describes a set of pairwise relations between different pairs of subject column and object column $(t^{*,0}, t^{*,n})$ where $n = 1, \ldots, S_k^c$. In addition, each relational table T_k is also accompanied with a page topic p_k. This is usually a short phrase (e.g., the page title) summarizing the table's main theme (e.g., the name of the artist performing in the music albums).

We now formally define our research problem as below:

Problem: Given a relational table dataset $\mathcal{D} = \{(T_k, p_k)\}_{k=1}^K$, our goal is to perform the following two table interpretation tasks:

Column type detection: we aim to predict the type of a column from among a fixed set of predefined types from an ontology, i.e., $f_c : \{\{t_k^{*,n}\}_{n=0}^{S_k^c}\}_{k=1}^K \rightarrow C$ where C is the set of target entity types;

Pairwise column relation prediction: we aim to predict the pairwise relation between the subject column and object columns, i.e., $f_r : \{\{(t_k^{*,0}, t_k^{*,n})\}_{n=1}^{S_k^c}\}_{k=1}^K \rightarrow \mathcal{R}$ where $\mathcal{R}$ is the set of known relations from the known ontology.

4 THE PROPOSED APPROACH

In this section, we present a novel deep architecture Table Convolutional Network (TCN) for relational table representation learning. TCN first learns the latent embedding of each relational table cell by aggregating both intra- and inter-table contextual information.

These learned cell embeddings are then summarized into column embedding which are used for predicting the column type and pairwise column relation. The overall framework of TCN is shown in Figure 2. We first introduce the intra-table aggregation module for summarizing cells of the same column and row (Section 4.1). Then, for capturing the contextual information across tables, we propose three specific inter-table aggregation methods to fully learn from various types of implicit connections between tables (Section 4.2). At last, we present the model's training procedure in a classic supervised setting as well as for pre-training on large-scale relational Web tables dataset (Section 4.3).

4.1 Intra-table Aggregations

For learning the latent representation of a target table cell $t^{m,n} \in T_k$, besides the information carried by its own cell value, it is natural to assume other cells in T_k of the same column or row are helpful for capturing the intra-table context of $t^{m,n}$. As an example, a single cell that of a common person name "Pete Bellotte" is ambiguous by itself, unless other song composer names appear in the same column, or his composed song names are present in the same row.

4.1.1 Column aggregation. We use $\mathbf{e}_{t^{m,n}} \in \mathbb{R}^{D_d}$ to denote the initial D_d-dim embedding vector of cell $t^{m,n}$. It can be pre-trained word embeddings [35] or simply setting to one-hot identifier vector. A straightforward way to consider other cells of the same column as context of $t^{m,n}$ is applying a pooling operator on their embeddings, e.g., $\sum_{m'=0}^{S_k^r} \mathbf{e}_{t^{m',n}}/(S_k^r - 1)$ where $m' \neq m$. However, different cells in the column have various contributions to the context of the target cell and they should be considered differently. For example, in a trending songs table of a singer, cells of his or her main artist songs should be more important of larger weight values compared with featured artist songs. This can be achieved by setting the target cell embedding $\mathbf{e}_{t^{m,n}}$ as query to attend on other cell embeddings $\{\mathbf{e}_{t^{m',n}}\}_{m'=0}^{S_k^r}$ $(m' \neq m)$ of the same column (see Figure 3(a)):

$$\alpha_{t^{m',n}} = \frac{\exp\left(\mathbf{e}_{t^{m',n}}{}^{\top} \cdot \mathbf{e}_{t^{m,n}}\right)}{\sum_{\tilde{m}=0, \tilde{m}\neq m}^{S_k^r} \exp\left(\mathbf{e}_{t^{\tilde{m},n}}{}^{\top} \cdot \mathbf{e}_{t^{m,n}}\right)}, \tag{1}$$

where $\alpha_{t^{m',n}}$ is the weight of column cell $t^{m',n}$. The column-wise aggregated context embedding $\mathbf{e}^c_{t^{m,n}} \in \mathbb{R}^{D_c}$ can be computed by

$$\mathbf{e}^c_{t^{m,n}} = \sigma\left(\mathbf{W}_c \cdot \sum_{m'=0, m'\neq m}^{S_k^r} \alpha_{t^{m',n}} \mathbf{e}_{t^{m',n}}\right), \tag{2}$$

where σ is nonlinear ReLU, $\mathbf{W}_c \in \mathbb{R}^{D_c \times D_d}$ is parameter matrix.

4.1.2 Row aggregation. Analogous to column aggregation, we can also use the target cell as query to attend and aggregate other row cells. However, different from column cells that are homogeneous of the same entity type, cells of the same row are mostly heterogeneous in type and contain complex relational information conditioning on the page topic p_k [51]. In other words, knowing the page topic can greatly benefit inferring the factual knowledge of other row cells with respect to $t^{m,n}$. For capturing the impact from page topic p_k, we incorporate the topic embedding vector $\mathbf{e}_{p_k} \in \mathbb{R}^{D_p}$ into the target cell query $\mathbf{e}_{t^{m,n}}$ for attending other row cells (see Figure 3(b)):

$$\beta_{t^{m,n'}} = \frac{\exp\left(\mathbf{e}_{t^{m,n'}}{}^{\top} \cdot \mathbf{W}_q \cdot (\mathbf{e}_{t^{m,n}} \| \mathbf{e}_{p_k})\right)}{\sum_{\tilde{n}=0, \tilde{n}\neq n}^{S_k^c} \exp\left(\mathbf{e}_{t^{m,\tilde{n}}}{}^{\top} \cdot \mathbf{W}_q \cdot (\mathbf{e}_{t^{m,n}} \| \mathbf{e}_{p_k})\right)}, \tag{3}$$

where $\mathbf{W}_q \in \mathbb{R}^{D_d \times (D_d + D_p)}$ is a bilinear transformation allowing interactions from row cells to both the target cell and page topic, and $\|$ is the vector concatenation operator. So, comparing with Eqn. (1) the attention weights of row cells are adaptively determined based on the target cell information as well as the page topic semantics.

In addition, we explicitly include the page topic into the row-wise aggregated context vector by concatenating the topic embedding $\mathbf{e}_{p_k}$ with the attended sum of row cell embeddings:

$$\mathbf{e}^r_{t^{m,n}} = \sigma\left(\mathbf{W}_r \cdot \sum_{n'=0, n'\neq n}^{S_k^c} \left(\beta_{t^{m,n'}} \mathbf{e}_{t^{m,n'}}\right) \| \mathbf{e}_{p_k}\right), \tag{4}$$

where $\mathbf{e}^r_{t^{m,n}} \in \mathbb{R}^{D_r}$ denotes the row-wise aggregated D_r-dim context embedding of $t^{m,n}$ and $\mathbf{W}_r \in \mathbb{R}^{D_r \times (D_d + D_p)}$ is parameter matrix. Intuitively, this can be seen as appending the page topic p_k as a pseudo-column of identical topic cells to the last column of T_k.

4.1.3 The intra-table aggregation module. After we have distilled contextual information of $t^{m,n}$ by aggregating from related cells of the same column and row in T_k, we can fuse these column- and row-wise aggregated context embeddings into a holistic intra-table context embedding $\mathbf{e}^a_{t^{m,n}} \in \mathbb{R}^{D_a}$. We use function AGG_a to denote this whole intra-table aggregation process (from Eqn. (2) to (5)):

$$\mathbf{e}^a_{t^{m,n}} = \sigma\left(\mathbf{W}_a \cdot (\mathbf{e}^c_{t^{m,n}} \| \mathbf{e}^r_{t^{m,n}})\right) = \mathrm{AGG}_a(t^{m,n}), \tag{5}$$

where $\mathbf{W}_a \in \mathbb{R}^{D_a \times (D_c + D_r)}$ is the parameter matrix. The output embedding $\mathbf{e}^a_{t^{m,n}}$ encapsulates the intra-table contextual information of target cell $t^{m,n}$ from all informative cells of relational table T_k. Most existing work of table representation learning rely on indirect techniques such as visibility matrix [8] and content snapshot [47] for modeling related cells inside the table and does not consider contexts across tables. In contrast, the proposed intra-table aggregation of TCN directly captures the intra-table context, and can be easily applied for integrating with various inter-table contexts. We use this intra-table aggregation function AGG_a as the underlying operation for summarizing all intra-table contexts of arbitrary cells that are implicitly connected to the target cell $t^{m,n}_k$.

4.2 Inter-table Aggregations

By aggregating from related cells in the same table, we can learn locally context-aware latent representation of a table cell. However, on the Web there are also a lot of *implicit* connections across different tables, and these hidden connections often provide highly valuable context that are complementary to the intra-table context. For example, the song composer's name "Pete Bellotte" could appear in multiple tables where he also serves as a record producer in some of them. These two roles are subtly different yet complementary to each other. Jointly modeling the intra- and inter-table contexts can benefit capturing more accurate relational information.

Such inter-table connections can also be of various types. Besides tables connected by the same cell value, there are often tables sharing the same schema (i.e., the same headers), and the topic

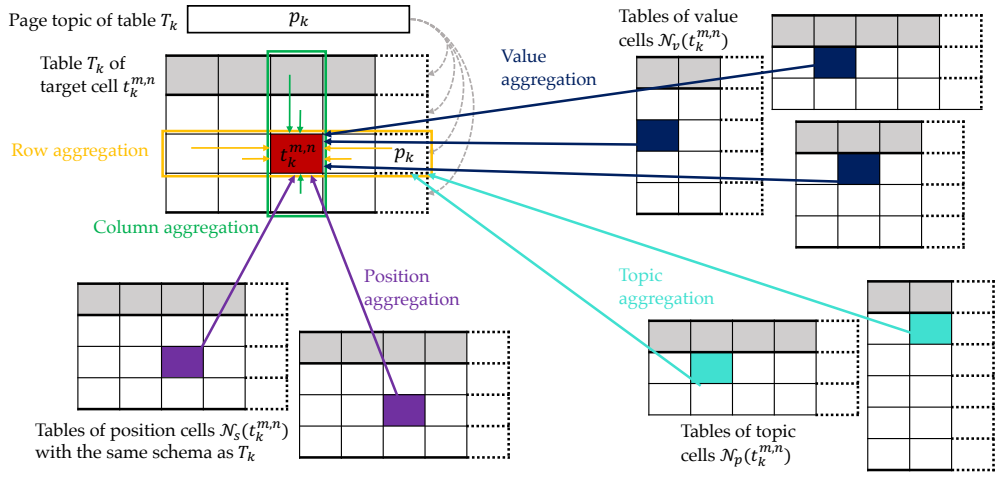
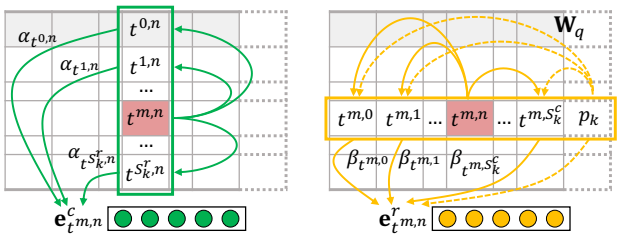


Figure 2: Overall framework of the proposed TCN for learning relational table latent representations by considering both the intra- and inter-table contextual information. Page topic p_k is appended to the right of each table as a pseudo-column (dashed cells). Arrows/cells highlighted in various colors denote different types of connection to the target cell $t_k^{m,n}$ and the corresponding aggregation module. The intra-table context of $t_k^{m,n}$ is aggregated from cells of the same column and row (green and yellow). Morevoer, 3 types of inter-table contexts are aggregated from (i) value cells $\mathcal{N}_b$ of the same value (blue), (ii) position cells $\mathcal{N}_s$ of the same schema position (purple), and (iii) topic cells $\mathcal{N}_p$ of the same value as $t_k^{m,n}$'s topic (turquoise) respectively.



(a) Column aggregation (b) Row aggregation

Figure 3: The intra-table aggregation module for summarizing contexts of target cell $t^{m,n}$ inside table T_k. (a) Column aggregation uses the embedding of $t^{m,n}$ as query to attend on other column cells for generating column-wise aggregated context $\mathbf{e}^c_{t^{m,n}}$. (b) Row aggregation also incorporates the embedding of page topic p_k into the query for attending other row cells. The result is concatenated with topic embedding as the row-wise aggregated context embedding $\mathbf{e}^r_{t^{m,n}}$.

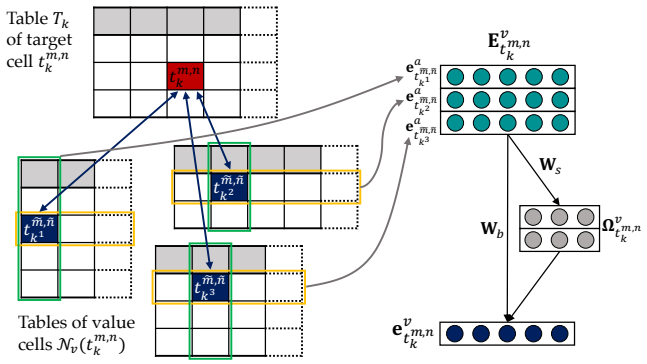


Figure 4: The value aggregation module for summarizing the target cell $t_k^{m,n}$'s inter-table contextual information from its value cells $\mathcal{N}_v(t_k^{m,n})$. Double-sided arrows indicate $\mathcal{N}_v$ share the same cell value as $t_k^{m,n}$. The intra-table contexts of $\mathcal{N}_v$ extracted via AGG_a (Section 4.1.3) are arranged into matrix $\mathbf{E}^v_{t^{m,n}}$. The value cells aggregated context embedding $\mathbf{E}^v_{t^{m,n}}$ is summed by self-attention weights of $\Omega^v_{t^{m,n}}$.

of certain tables might appear in other tables as cell values. For example, Web tables designed for describing music albums will have the identical header cells, and the page topic (i.e., the album name) can also appear in the singer's discography table. To effectively modeling context of heterogeneous connections, we propose three inter-table aggregation modules for distilling inter-table contexts.

4.2.1 Value aggregation. Each relational table describe a set of relations between its cells, and in turn each unique cell could also be expressed by a set of tables where it appears. Intuitively, each table can be seen as a partial view of target cell's context. For capturing the contextual information from different tables, we establish connections between cells of different tables containing the same value. Particularly, we adopt basic normalizing procedures to canonicalize table cells with no additional step of expensive entity linking [31, 36, 51]. In practice, we apply minimal preprocessing on cell values by lowering all cases and removing redundant spaces. Given a target cell $t_k^{m,n}$ in relational table T_k, we use $\mathcal{N}_v$ to denote cells of other tables containing the same value, i.e., $\mathcal{N}_v(t_k^{m,n}) \coloneqq \{t_{k'}^{\tilde{m},\tilde{n}} \mid t_{k'}^{\tilde{m},\tilde{n}} = t_k^{m,n} \wedge 0 \le k' \le K \wedge k' \ne k\}$.

By applying the intra-table aggregation function AGG_a as previously introduced (Section 4.1.3), we can produce the local contexts of all value cells $\{\mathbf{e}^a_{t_{k'}^{\tilde{m},\tilde{n}}} = \text{AGG}_a(t_{k'}^{\tilde{m},\tilde{n}}) \mid t_{k'}^{\tilde{m},\tilde{n}} \in \mathcal{N}_v(t_k^{m,n})\}$ with respect to the corresponding relational table. For effectively focusing on the most useful connections of value cells, we further process

this variant-sized set of value cells' intra-table contexts into a single vector (see Figure 4) by leveraging the self-attention mechanism [42]. Specifically, we arrange all extracted intra-table contexts of $\mathcal{N}_v(t_k^{m,n})$ into a matrix of $\mathbf{E}^v_{t^{m,n}} \in \mathbb{R}^{|\mathcal{N}_v(t_k^{m,n})| \times D_a}$, where each row contains the context aggregated from one value cell of $t_k^{m,n}$. The relative importance for value cells of $t_k^{m,n}$ can be calculated as:

$$\mathbf{\Omega}^v_{t^{m,n}} = \text{softmax}\left(\mathbf{W}_s \cdot \left(\mathbf{E}^v_{t^{m,n}}\right)^\top\right), \tag{6}$$

where $\mathbf{W}_s \in \mathbb{R}^{V \times D_a}$ is a parameter matrix for computing the V-view weight matrix $\mathbf{\Omega}^v_{t^{m,n}} \in \mathbb{R}^{V \times |\mathcal{N}_v(t_k^{m,n})|}$, the softmax is applied row-wisely, and V is the number of attention heads setting to 2 in practice. Each row of $\mathbf{\Omega}^v_{t^{m,n}}$ reflects one view of the value cells importance distribution. Note if $V = 1$, $\mathbf{W}_s$ degenerates into a parameter query vector and the softmax function can be expanded out similarly to Eqn. (1). Then, the value cells aggregated context embedding can be computed as:

$$\mathbf{E}^v_{t^{m,n}} = \text{mean}\left(\mathbf{\Omega}^v_{t^{m,n}} \cdot \mathbf{E}^v_{t^{m,n}} \cdot \mathbf{W}_b\right), \tag{7}$$

where $\mathbf{W}_b \in \mathbb{R}^{D_a \times D_b}$ is the parameter matrix for transforming into D_b-dim value cells aggregated context, and the final output $\mathbf{E}^v_{t^{m,n}} \in \mathbb{R}^{D_b}$ is obtained via a element-wise mean pooling function.

4.2.2 Position aggregation. Besides linking tables based on their cell values, the unique grid-like structure of relational tables also grants us a valuable foundation for establishing connections between tables based on the cell's relative position inside the table. The intuition is that for a subset of relational tables with the same schema, i.e., $\{T_k \mid \phi(k) = u\}$ $(1 \le u \le U)$, cells at the same position in terms of row index m $(1 \le m \le \max(\{S^r_k \mid \phi(k) = u\}))$ and column index n $(0 \le n \le \max(\{S^c_k \mid \phi(k) = u\}))$ could provide useful contextual information to each other. For example, suppose there is a collection of identical schema tables describing various music albums, knowing any cell of a song track name (or composer) would reveal other cells of the same position also instantiate the same "Release" (or "People") type. We use $\mathcal{N}_s$ to denote position cells, i.e., $\mathcal{N}_s(t_k^{m,n}) \coloneqq \{t_{k'}^{m,n} \mid \phi(k) = \phi(k') \wedge 0 \le k' \le K \wedge k' \ne k\}$.

In general domain, the connections between $t_k^{m,n}$ and position cells $\mathcal{N}_s$ maybe sparse because the number of unique table schema U is also large and comparable to total number of tables K. However, an important practical case is relational tables on semi-structured website which consist of a set of detail pages that each contains information about a particular page topic [24, 27]. Typically, the factual information of these tables are automatically populated from an underlying database. When the relational table dataset $\mathcal{D}$ is constructed from such semi-structured websites, there are a large number of helpful inter-table connections to position cells $\mathcal{N}_s$.

Without losing generality, we propose to aggregate from position connections, which potentially is a rich source of the target cell's inter-table contextual information. For generating the position cells aggregated context embedding $\mathbf{e}^s_{t^{m,n}} \in \mathbb{R}^{D_s}$, we adpot the similar strategy proposed for aggregating value cells (see Section 4.2.1). Specifically, we arrange all intra-table contexts of $\mathcal{N}_s(t_k^{m,n})$ into matrix $\mathbf{E}^s_{t^{m,n}} \in \mathbb{R}^{|\mathcal{N}_s(t_k^{m,n})| \times D_a}$ and substitute it into Eqn. (6). The result $\mathbf{\Omega}^s_{t^{m,n}}$ is further substituted into Eqn. (7) for computing $\mathbf{e}^s_{t^{m,n}}$. Note that we truncate the number of position cells when $|\mathcal{N}_s|$ is too large according to a sampling budget b in practice to maintain computational tractability. We will test the effectiveness of our proposed method in both general domain and on a specially constructed dataset from semi-structured websites.

4.2.3 Topic aggregation. Another important type of implicit inter-table connections can be discovered by examining the underlying connectivity between the page topic of target cell and cells of other tables. Relational Web tables are mainly created for conveying knowledge of relations that are relevant to the page topic. The table cells and topic both refer to relevant real world entities [51]. It is common to see the topic of certain tables appear as the cell values of other relational tables. In the example of music album tables, the page topic of album name would appear in other relational tables such as the singer's discography. So, it is also beneficial for the model to extract contextual information from these topic cells.

We use $\mathcal{N}_p$ to denote topic cells containing the same value as the page topic p_k of target cell $t_k^{m,n}$, i.e., $\mathcal{N}_p(t_k^{m,n}) \coloneqq \{t_{k'}^{\tilde{m},\tilde{n}} \mid t_{k'}^{\tilde{m},\tilde{n}} = p_k \wedge 0 \le k' \le K \wedge k' \ne k\}$. Similar to the treatment of value cells $\mathcal{N}_v$ and position cells $\mathcal{N}_s$, we first apply the intra-table aggregation function AGG_a (Section 4.1.3) to extract $\mathcal{N}_p$'s intra-table contexts for constructing $\mathbf{E}^p_{t^{m,n}}$, and then generate the topic cells aggregated context embedding $\mathbf{e}^p_{t^{m,n}} \in \mathbb{R}^{D_p}$ according to Eqn. (6) and (7). Different from the inter-table connections made by $\mathcal{N}_v$ and $\mathcal{N}_s$, which are directly linking to the target cell $t_k^{m,n}$, topic cells $\mathcal{N}_p$ connect to the page topic p_k of $t_k^{m,n}$. Instead of simply using $\mathbf{e}^p_{t^{m,n}}$ as the part of the contextual information of $t_k^{m,n}$, we fuse it with p_k's initial embedding $\mathbf{e}_{p_k}$. This can be seen as bringing inter-table contextual information into the embedding of page topic, and the result $\mathbf{e}^p_{p_k} = \mathbf{e}_{p_k} \| \mathbf{e}^p_{t^{m,n}}$ is substituted into Eqn. (3). In this way, we incorporate the global contexts of topic cells into the intra-table aggregation function AGG_a of the model (Section 4.1.2).

By aggregating from column and row cells, as well as various types of inter-table connections to value, position and topic cells, the proposed TCN fully considers both the intra- and inter-table contextual information during learning. Next, we present the fusion of all contexts and the training procedures of the model.

4.3 Training Objectives

After generating the intra-table contextual embedding $\mathbf{e}^a_{t^{m,n}}$ and different inter-table contextual embeddings $\mathbf{e}^v_{t^{m,n}}$, $\mathbf{e}^s_{t^{m,n}}$, $\mathbf{e}^p_{t^{m,n}}$, the final latent representation of target cell $t^{m,n}$ can be computed as:

$$\mathbf{h}_{t^{m,n}} = \sigma\left(\mathbf{W}_h \cdot \left(\mathbf{e}_{t^{m,n}} \| \mathbf{e}^a_{t^{m,n}} \| \mathbf{e}^v_{t^{m,n}} \| \mathbf{e}^s_{t^{m,n}}\right)\right), \tag{8}$$

where $\mathbf{W}_h$ is the parameter matrix for fusing the initial cell embedding $\mathbf{e}_{t^{m,n}}$ with all intra- and inter-table contextual embeddings into the final D_h-dim representation of $t^{m,n}$. Note that topic cells aggregated context $\mathbf{e}^p_{t^{m,n}}$ is incorporated in $\mathbf{e}^a_{t^{m,n}}$ via AGG_a.

4.3.1 Multi-tasking. The proposed TCN can be trained in a supervised mode by jointly predicting the type of columns and pairwise relation between columns for each relational table. Since both of these two multi-class classification tasks are on table column level, we compute the embedding $\mathbf{h}_{t_k^{*,n}} \in \mathbb{R}^{D_h}$ of table column $t_k^{*,n}$ as the mean of its cell embeddings, i.e., $\mathbf{h}_{t_k^{*,n}} = \text{Avg}\left(\{\mathbf{h}_{t_k^{m,n}}\}_{m=0}^{S^r_k}\right)$. For column type prediction, we use a single dense layer as the final predictive model. The discrepancy between the predicted type

distribution and the ground truth column type is measured by the loss function $\mathcal{J}^C$. Specifically, given $c_{t_k^{*,n}} \in C$ denoted as the true type of $t_k^{*,n}$, we employ the following cross-entropy objective:

$$\mathcal{J}_k^C = -\sum_{n=0}^{S_k^c} \sum_{c \in C} \mathbb{I}_{c_{t_k^{*,n}}=c} \cdot \log \frac{\exp\left(\mathbf{M}_c \cdot \mathbf{h}_{t_k^{*,n}}\right)}{\sum_{c' \in C} \exp\left(\mathbf{M}_{c'} \cdot \mathbf{h}_{t_k^{*,n}}\right)}, \quad (9)$$

where $\mathbf{M}_c$ is the parameter matrix for column type $c \in C$ and $\mathbb{I}$ is an indicator function.

Similarly, we concatenate the embeddings of a pair of subject and object columns $(t_k^{*,0}, t_k^{*,n})$ for feeding into a dense layer to generate the prediction on the true relation $r_{t_k^{*,n}} \in \mathcal{R}$ between them:

$$\mathcal{J}_k^{\mathcal{R}} = -\sum_{n=1}^{S_k^c} \sum_{r \in \mathcal{R}} \mathbb{I}_{r_{t_k^{*,n}}=r} \cdot \log \frac{\exp\left(\mathbf{M}_r \cdot \left(\mathbf{h}_{t_k^{*,0}} \,\|\, \mathbf{h}_{t_k^{*,n}}\right)\right)}{\sum_{r' \in \mathcal{R}} \exp\left(\mathbf{M}_{r'} \cdot \left(\mathbf{h}_{t_k^{*,0}} \,\|\, \mathbf{h}_{t_k^{*,n}}\right)\right)}, \quad (10)$$

where $\mathbf{M}_r$ is parameter matrix for pairwise column relation $r \in \mathcal{R}$.

So, given a mini-batch of relational tables $\mathcal{B} \subseteq \mathcal{D}$, the overall training objective $\mathcal{J}$ of the proposed TCN can be obtained via a convex combination of the above two tasks' loss functions:

$$\mathcal{J} = \sum_{k \in \mathcal{B}} \gamma \mathcal{J}_k^C + (1-\gamma)\mathcal{J}_k^{\mathcal{R}}, \quad (11)$$

where γ is a mixture hyperparameter for balancing the magnitude of two objectives for predicting column type and pairwise relation.

4.3.2 Unsupervised pre-training. Learning relational table representation directly under the supervision of column type and relation labels are not always feasible due to the expensive cost for obtaining high quality annotations. The proposed TCN can also be trained in an unsupervised way without relying on explicit labels. Specifically, we first train TCN according to the pre-training objective to obtain the output cell embeddings. We then use these pre-trained cell embeddings as initialization for the supervised fine-tuning phase aimed at jointly predicting column type and pairwise column relation (Eqn. (11)). Similar to the the Masked Language Model (MLM) objective of BERT [9], we randomly mask 10% of table cells beforehand for recovery. Given a masked cell $\hat{t}_k^{m,n}$ and the global context-aware embedding $\mathbf{h}_{\hat{t}_k^{m,n}}$ learned by TCN, the objective for predicting the original cell value is computed as:

$$\mathcal{J} = -\sum_{k \in \mathcal{B}} \sum_{\hat{t}_k^{m,n} \in \hat{T}_k} \sum_{v \in \mathcal{V}} \mathbb{I}_{t_k^{m,n}=v} \cdot \log \frac{\exp\left(\mathbf{M}_v \cdot \mathbf{h}_{\hat{t}_k^{m,n}}\right)}{\sum_{v' \in \mathcal{V}} \exp\left(\mathbf{M}_{v'} \cdot \mathbf{h}_{\hat{t}_k^{m,n}}\right)}, \quad (12)$$

where $\hat{T}_k$ is the set of all masked cells of the k-th table, $\mathcal{V}$ is the set of all cell values of $\mathcal{D}$, and $\mathbf{M}_v$ is parameter matrix for cell value v which could include one or multiple words after the normalization (Section 4.2.1). The pre-trained cell embeddings can later be used for initializing fine-turning phase. As we show in experiments (Section 5.5), pre-training improves the performance by +2.0% and +2.3% of F1-weighted for predicting column type and relation.

4.4 Complexity

Assuming the attention-based intra-table aggregation function AGG_a takes constant time, the per-batch time complexity of the proposed TCN is $O(|B|\,(b_v + b_s + b_p))$ in principle, where $|B|$ is batch size and b_v (and b_s, b_p) is the sampling budget for inter-table connections of value (and position, topic) cells. In practice, we simply set $b_v = b_s = b_p$ so the time complexity becomes $O(|B|b)$ which is linear to the product of batch size $|B|$ and sampling budget b. In optimized implementation, we can process tables in batch into one large table beforehand by padding and concatenating cells which can further reduce the complexity to $O(b)$. This allows us to scale the model to tens of millions tables while remaining control on the balance between model expressiveness and efficiency.

Table 2: Statistics of two real Web table datasets $\mathcal{D}^m$ and $\mathcal{D}^w$.

	#tables K	Avg. # rows S_k^r	Avg. # cols. S_k^c	#column types $\lvert C\rvert$	#pairwise relations $\lvert\mathcal{R}\rvert$
$\mathcal{D}^m$	128,443	7.0	3.6	8	14
$\mathcal{D}^w$	55,318	16.1	2.4	201	121
	#schemas U	**Avg. #value cells $\lvert\mathcal{N}_v\rvert$**	**Avg. #position cells $\lvert\mathcal{N}_s\rvert$**		**Avg. #topic cells $\lvert\mathcal{N}_p\rvert$**
$\mathcal{D}^m$	11	10.7	5048.1		2.4
$\mathcal{D}^w$	6,538	7.2	0.9		1.9

5 EXPERIMENTS

In this section, we evaluate the performance of the proposed TCN for relational table representation learning against competitive baselines on two real world large-scale relational Web tables datasets. Particularly, we aim at answering the following research questions:

- **RQ1**: How does the proposed method perform compared with the state-of-the-art methods for predicting relational table's column type and pairwise relation between columns?
- **RQ2**: How does each type of the proposed inter-table aggregation module affect the overall performance of the model?
- **RQ3**: Is the proposed method also effective when applied for pre-training on unlabeled corpus of relational tables?
- **RQ4**: What are some concrete examples of the column type and column pairwise relations discovered by the model?
- **RQ5**: What are the recommended hyper-parameter settings for applying the proposed model in practical cases?

5.1 Datasets

We collected a dataset $\mathcal{D}^m$ of 128K relational Web tables from 6 mainstream semi-structured websites in the music domain. Tables of $\mathcal{D}^m$ generally fall into three page topic categories: "Person" (e.g., singer, composer and etc.), "Release" (i.e., music album), and "Recording" (i.e., song track). The number of unique table schemas in $\mathcal{D}^m$ is relatively small ($U = 11$) because tables are automatically populated. We manually annotated each table schema to obtain column type and pairwise column relation information. For relational Web tables in the general domain, we utilize datasets provided by Deng *et al.* [8] containing annotated relational tables from a raw corpus of 570K Web tables on Wikipedia. We build a dataset $\mathcal{D}^w$ by taking a subset of 5.5K tables with annotations on both column type and relation. Specifically, we take the intersection of task-specific

Table 3: Performance of baselines and variants of TCN and on predicting column type C and pairwise column relation $\mathcal{R}$ on dataset $\mathcal{D}^m$. For all metrics, higher values indicate better performance. Bold highlights global highest values. Underline denotes best performance among baselines. Relative improvements over the base variant TCN-intra are shown in parenthesis.

Method	Column type C			Pairwise column relation $\mathcal{R}$		
	Acc.	F1-weighted	Cohen's kappa κ	Acc.	F1-weighted	Cohen's kappa κ
Table2Vec	.832	.820	.763	.822	.810	.772
TaBERT	.908	.861	.834	.877	.870	<u>.846</u>
TURL	.914	.877	<u>.876</u>	<u>.890</u>	<u>.889</u>	.838
HNN	.916	.883	.869	.848	.843	.794
Sherlock	<u>.922</u>	<u>.895</u>	.863	.831	.818	.802
TCN-intra	.911	.881	.873	.893	.894	.869
TCN-$\mathcal{N}_v$	.939 (+3.1%)	.916 (+4.0%)	.897 (+2.8%)	.920 (+3.0%)	.920 (+2.9%)	.898 (+3.3%)
TCN-$\mathcal{N}_s$	.934 (+2.5%)	.908 (+3.1%)	.894 (+2.4%)	.908 (+1.7%)	.912 (+2.0%)	.881 (+1.4%)
TCN-$\mathcal{N}_p$	.923 (+1.3%)	.890 (+1.0%)	.880 (+0.8%)	.906 (+1.4%)	.904 (+1.1%)	.875 (+0.7%)
TCN	**.958 (+5.2%)**	**.938 (+6.5%)**	**.913 (+4.6%)**	**.934 (+4.6%)**	**.925 (+3.5%)**	**.905 (+4.1%)**

datasets for column type annotation and relation extraction described in the paper. For both datasets, we keep tables with at least 2 columns/rows. More descriptive statistics are provided in Table 2.

5.2 Experimental Settings

5.2.1 Baseline methods. We compare the proposed TCN against the state-of-the-art tabular data representation learning methods:

- Table2Vec [49]: This method flattens a cropped portion of relational table and its topic into a sequence of cell tokens and uses word2vec [29] for learning column/cell embeddings.
- TaBERT [47]: It jointly learns embeddings of natural language sentences and relational tables using its content snapshots to carve out relevant rows for feeding into BERT [9].
- TURL [8]: This method uses linearized cells and linked entities as input into its Transformer [42] based encoder enhanced with structure-aware visibility matrix for pre-training.

Besides, we consider methods that are specifically designed for column type prediction or pairwise column relation extraction:

- HNN [6]: This method models the contextual semantics of relational table column using a bidirectional-RNN with an attention layer and learns column embeddings with a CNN.
- Sherlock [18]: It utilizes four categories of statistical features describing the character/word distributions and semantic features to predict the semantic type on column level.

We use open-source implementations provided by the original papers for all baseline methods and follow the recommended setup guidelines when possible. For TaBERT, we use the page topic as its NL utterance. And we set the embedding of linked entity in TURL the same as its cell since they are not provided as input in our case. As HNN and Sherlock are originally designed for predicting semantic type of columns, we concatenate the embeddings of subject and object pair of columns to predict the relation. We also tested with representative methods (e.g., MTab, MantisTable, CVS2KG) from the SemTab 2019 challenges[i]. But we were only able to successfully generate results from MTab due to the high time complexity of others. We observed a relative large performance gap between it and Table2Vec's (probably because of its dependence on pre-linked entities) so we exclude it in following discussions. For all methods, we use the same random split of 80/10/10 percents of tables for training/validation/test at each round and we report the average performance of five runs. For TCN, the sampling budget b for inter-table connections is set to 20 and objective mixture weight γ is set to 0.5. We set the dimension of cell and all context embeddings as 300 and initialize each cell embedding by matching with Glove embeddings [35] (taking mean in case of multiple tokens).

5.2.2 Evaluation metrics. For classifying multi-class column type C and pairwise column relation $\mathcal{R}$, we use metrics of *mean accuracy* (Acc.), F1-weighted score and the *Cohen's kappa* κ coefficient.

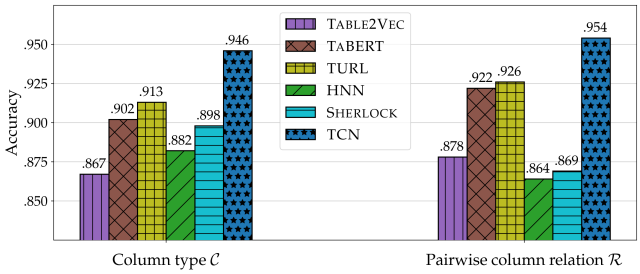
5.3 Overall Performance (RQ1)

Table 3 and Figure 5 presents the experimental results of applying the proposed TCN and baseline methods on predicting relational Web table column type C and pairwise column relation $\mathcal{R}$ on dataset $\mathcal{D}^m$ and $\mathcal{D}^w$ respectively. In the following discussions, we refer to F1-weighted as F1 unless explicitly stated otherwise.

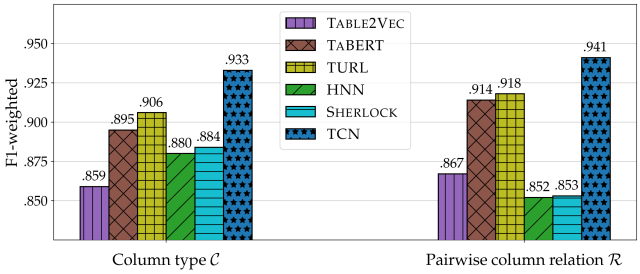
For tabular data representation learning methods, TURL performs better than other baselines on both tasks across datasets. Table2Vec underperforms all other methods because it simply crops and flattens part of the table for feeding into the shallow word2vec. There is quite large performance margin (+5.0% and +7.4% for two tasks on $\mathcal{D}^m$ in terms of F1) from Table2Vec to TaBERT using a deep BERT-based encoder. TURL enhances the deep encoder with visibility mask which can partially capture the grid-like structure of relational table so it has better performance compared with Table2Vec and TaBERT. This means better modeling the intra-table context is beneficial. But these methods all rely on linearizing the table into a long sequence of cell tokens.

For methods specifically designed for column type prediction, Sherlock performs better than HNN for predicting column type C on two datasets. However, it cannot produce competitive results for predicting pairwise column relation $\mathcal{R}$ on two datasets which is

[i]http://www.cs.ox.ac.uk/isg/challenges/sem-tab/2019/



(a) Performance of TCN and baselines on dataset $\mathcal{D}^w$ in terms of accuracy.



(b) Performance of TCN and baselines on dataset $\mathcal{D}^w$ in terms of F1-weighted.

Figure 5: The proposed TCN can consistently outperform baseline methods for column type $\mathcal{C}$ prediction and pairwise column relation $\mathcal{R}$ extraction on open domain dataset $\mathcal{D}^w$.

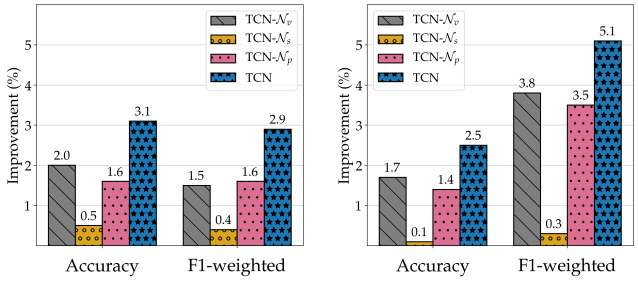
roughly on the same performance level as TABLE2VEC. It is interesting to note SHERLOCK achieves the best performance for predicting column type $\mathcal{C}$ on $\mathcal{D}^m$ among all baselines because of the effectiveness of its statistical features. But none of these baselines is capable of modeling the valuable inter-table contextual information.

The proposed TCN can consistently outperform baseline methods on all metrics across two datasets. For predicting the column type $\mathcal{C}$, TCN scores an F1 of .938 on $\mathcal{D}^m$ which is +4.8% relatively over SHERLOCK (+7.0% relatively over TURL), and scores an F1 of .933 on $\mathcal{D}^w$ which is +5.5% relatively over SHERLOCK (+3.0% relatively over TURL). For predicting the pairwise column relation $\mathcal{R}$, TCN can generate an F1 of .925 on $\mathcal{D}^m$ which is +4.1% relatively over TURL, and score an F1 of .941 on $\mathcal{D}^w$ which is +2.5% relatively over TURL. This justifies the effectiveness TCN's inter-table aggregation modules for capturing the contextual information across various types of implicitly connected tables. These inter-table contexts are complementary to the intra-table context providing additional discriminative power in downstream tasks.

5.4 Ablation Studies (RQ2)

To further validate the effectiveness of each inter-table aggregation module of TCN, we propose 4 model variants by including selected type(s) of inter-table aggregations and compare against them:

- TCN-intra: Base version only considers intra-table contexts, i.e., cells of the same column/row, using the intra-table aggregation AGG_a (Section 4.1.3) on each table independently.
- TCN-$\mathcal{N}_v$: This variant considers the inter-table context of value cells $\mathcal{N}_v$ (Section 4.2.1) besides intra-table context.



(a) Column type $\mathcal{C}$ (b) Pairwise column relation $\mathcal{R}$

Figure 6: Relative improvements of TCN and variants over the base variant TCN-intra for predicting column type $\mathcal{C}$ and pairwise column relation $\mathcal{R}$ on the open domain dataset $\mathcal{D}^w$.

- TCN-$\mathcal{N}_s$: Position cells $\mathcal{N}_s$ of the same schema position (Section 4.2.2) are considered as the inter-table context here.
- TCN-$\mathcal{N}_p$: Topic cells $\mathcal{N}_p$ of the same value as target cell's page topic (Section 4.2.3) are considered in this variant.

The performance of TCN and its variants are presented in Table 3 and Figure 6. The base variant TCN-intra which only considers intra-table contextual information performs roughly on the same level as the best baseline model TURL for tabular data representation learning. This demonstrates the intra-table aggregation function AGG_a can successfully summarize the contextual information inside the target table from cells of the same column and row. We compare other TCN's variant against TCN-intra for evaluating the relative contribution of each inter-table aggregation module.

The TCN-$\mathcal{N}_v$ considering inter-table contexts from value cells provides a significant increase on the performance for both tasks. By only aggregating from value cells, TCN-$\mathcal{N}_v$ can score F1s of .916 and .921 for predicting $\mathcal{C}$ on two datasets which are relatively +2.4% and +3.6% over the baseline SHERLOCK (similar trend also hold for predicting $\mathcal{R}$). This indicates value cells of the same value as target cell consistently provide rich inter-table contextual information complementary to the intra-table contexts, and the value cells aggregation module of TCN is effective in learning from them.

The TCN-$\mathcal{N}_s$ considering position cells besides the intra-table context gives high performance for both tasks on dataset $\mathcal{D}^m$ but is less helpful on dataset $\mathcal{D}^w$. This is because of the difference in two datasets: $\mathcal{D}^m$ contains relational tables from semi-structured websites with rich connections between position cells ($\overline{|\mathcal{N}_s|} > 5 \times 10^3$) while $\mathcal{D}^w$ is constructed from open domain Wikipedia pages with sparse position cells ($\overline{|\mathcal{N}_s|} = 0.9$). Given relatively densely connected relational table schemas, we found aggregating from position cells can provide us useful inter-table contextual information.

The TCN-$\mathcal{N}_p$ considering topic cells besides the intra-table context generally gives good improvements of performance for both tasks on two datasets comparable to the improvements provided by TCN-$\mathcal{N}_v$. This confirms that aggregating from topic cells of other tables can provide TCN-$\mathcal{N}_p$ additional contextual information. And, at last, by considering all three types of inter-table aggregation modules, the full version of TCN can consistently improve upon TCN-intra which focuses only on the intra-table context by +6.5% and +5.1% for two tasks across datasets in terms of F1.

Table 4: Performance of TCN and baselines under different training settings evaluated in terms of F1-weighted.

Method	$\mathcal{D}^m$		$\mathcal{D}^w$	
	Type C	Relation $\mathcal{R}$	Type C	Relation $\mathcal{R}$
TCN (supervised)	.938	.925	.933	.941
TCN + TaBERT for pre-training	.948	.933	.942	.949
TCN + TURL for pre-training	.945	.934	.946	.953
TCN full w/ pre-training & fine-tuning	.957	.946	.951	.960

5.5 Unsupervised Pre-training (RQ3)

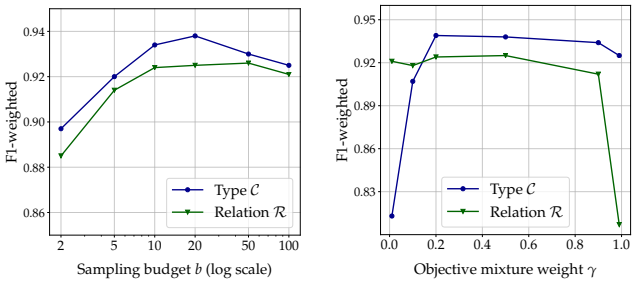
Besides training the proposed TCN under the explicit supervision of column type and relation labels, we can also pre-train TCN on large-scale unlabeled corpus of relational Web tables and fine-tune on downstream tasks (see Section 4.3.2). We also leverage two baseline methods (TaBERT and TURL) capable of pre-training on unlabeled relational tables to obtain initial cell embeddings as input for supervised multi-task training of TCN. We conduct comparative experiments on both datasets and present the results in Table 4.

We can see that incorporating the pre-trained cell embeddings of two baseline methods can improve the final performance of TCN on both tasks of predicting column type C and pairwise column relation $\mathcal{R}$. The performance improvement gained by utilizing the pre-trained embeddings of TURL is similar to TaBERT since they both focus on effectively summarizing intra-table contextual information during the pre-training phase. In contrast to TaBERT and TURL, the proposed TCN can naturally incorporate different types of inter-table context during the pre-training phase without column type or relation labels. So, by combing the intra- and inter-table contexts learned during both the pre-training and fine-tuning phases, the propose TCN can score F1s of .957 and .951 for predicting C on two datasets (+2.0% and +1.9% relatively over TCN without pre-training); and scores F1s of .946 and .960 for predicting $\mathcal{R}$ (+2.3% and +2.0% relatively over TCN without pre-training).

5.6 Qualitative Analysis (RQ4)

Our goal of TCN is to automatically discover column type and pairwise relation between columns given a relational Web table and an external knowledge base. We provide concrete examples on the output of TCN to show its effectiveness and practical utilities.

We used a proprietary music ontology for dataset $\mathcal{D}^m$ in experiments which includes column types such as "Release", "People", "Recording", "RecordLabel", "XMLSchema#string", and etc. We found TCN can achieve almost perfect performance on major types (e.g., column ["Lucky Old Sun", "Life on a Rock", "Cosmic Hallelujah",...] predicted as "Release", and column ["Johnny Cash", "Joan Jett", "John Keawe",...] predicted as type "People"). We also found that column like ["US", "Japan", "UK",...] was classified as the "XMLSchema#string" because there is no corresponding type, i.e., "Nation" in the given knowledge base. This can be optimized by replacing alternative source ontology in the music domain with more complete coverage and finer granularity. For the task of predicting relations between column pairs, the pre-defined column pairwise relations include "hasPerformer", "hasOriginalReleaseDate", "hasDuration", "isRecordingOfTrack", "isTrackOfRelease", and etc. One interesting example of the identified relation between columns of ["Miles Wintner", "Harmony Tividad", "Sariah Mae",...] and ["drums", "bass", "keyboards",...] is "playsInstrument" although the latter column is identified as type "XMLSchema#string". This indicates the potential advantage of jointly modeling the type and relation of columns to capture their complementary information.



(a) Improving the value of sampling budget b is generally beneficial until 20.

(b) A choice of γ in range of [0.2, 0.8] can yield stable model performance.

Figure 7: Sensitivity of TCN's performance on different values of sampling budget b and objective mixture weight γ.

5.7 Sensitivity (RQ5)

We examine the impact of TCN's key hyper-parameters: (1) the sampling budget b for different types of inter-table connections ($|\mathcal{N}_b|$, $|\mathcal{N}_s|$, and $|\mathcal{N}_p|$), and (2) the mixture weight γ of overall objective (Eqn. (11)), on the model's performance using dataset $\mathcal{D}^m$. We analyze these two hyper-parameters adopting the gird-search strategy: b is chosen from $\{2, 5, 10, 20, 50, 100\}$ and γ is chosen from $\{0.01, 0.1, 0.2, 0.5, 0.9, 0.99\}$. Figure 7 presents the results.

In Figure 7(a) we can see that improving the value of b from 2 to 10 can noticeably improve the performance because larger b values allow TCN to learn from more inter-table connections and thus aggregating more contextual information across tables. But further improving b brings in diminishing benefits and large values such as 100 in turn hurts the performance because of additional noise. In Figure 7(b), we can see that TCN's performance is generally stable for a γ values in range of [0.2, 0.8]. For extreme cases of small (or large) values of γ, the supervised multi-task objective of TCN will degenerates into single-task setting because the gradients from one task are being squashed leading to the model only focuses on either the column type or pairwise relation prediction. In practice, we recommend finding the optimal values of b based on the dataset and choosing a balancing value of γ such as 0.5.

6 CONCLUSIONS

In this work, we proposed a novel approach for learning relational table latent representations. Our proposed method aggregates cells of the same column and row into the intra-table context. In addition, three types of inter-table contexts are aggregated from value cells of the same value, position cells of the same position, and topic cells of the same value as target cell's page topic. Extensive experiments

on two real relational table datasets from open domain and semi-structured websites demonstrated the effectiveness of our model. We focus on relational Web tables of horizontal format in this work but a huge number of tables with various types and structures can be found on the Web. We consider handling complex table formats such as different orientations, composite header cells, changing subject column index and nested cells as an interesting future direction. Also, applying the leaned cell embeddings for other downstream tasks such as cell entity linking, table type prediction, and table relation detection would be promising to explore in the future

ACKNOWLEDGMENTS

We thank all anonymous reviewers for valuable comments. This research was supported in part by NSF Grants IIS-1849816.

REFERENCES

[1] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: entity linking in web tables. In *International Semantic Web Conference*. Springer, 425–441.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[3] Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten years of webtables. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2140–2149.

[4] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1, 1 (2008), 538–549.

[5] Michael J. Cafarella, A. Halevy, Y. Zhang, D. Wang, and E. Wu. 2008. Uncovering the Relational Web. In *WebDB*.

[6] J Chen, I Horrocks, E Jimenez-Ruiz, and C Sutton. 2019. Learning Semantic Annotations for Tabular Data. *IJCAI 2019* (2019), 2088–2094.

[7] Marco Cremaschi, Roberto Avogadro, and David Chieregato. 2019. MantisTable: an Automatic Approach for the Semantic Table Interpretation. *SemTab ISWC* 2019 (2019), 15–24.

[8] Xiang Deng, Huan Sun, A. Lees, Yingfang Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *ArXiv* abs/2006.14806 (2020).

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

[10] Xin Luna Dong, Hannaneh Hajishirzi, Colin Lockard, and Prashant Shiralkar. 2020. Multi-Modal Information Extraction from Text, Semi-Structured, and Tabular Data on the Web. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3543–3544. https://doi.org/10.1145/3394486.3406468

[11] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. 2017. Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *International Semantic Web Conference*. Springer, 260–277.

[12] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. 2014. A hybrid machine-crowdsourcing system for matching web tables. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 976–987.

[13] Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference*. 2736–2742.

[14] Anna Lisa Gentile, Petar Ristoski, Steffen Eckel, Dominique Ritze, and Heiko Paulheim. 2017. Entity Matching on Web Tables: a Table Embeddings approach for Blocking.. In *EDBT*. 510–513.

[15] Majid Ghasemi-Gol and Pedro A. Szekely. 2018. TabVec: Table Vectors for Classification of Web Tables. *ArXiv* abs/1802.06290 (2018).

[16] Ralph Grishman. 2015. Information extraction. *IEEE Intelligent Systems* 30, 5 (2015), 8–15.

[17] Jonathan Herzig, P. Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TAPAS: Weakly Supervised Table Parsing via Pre-training. In *ACL*.

[18] 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*. 1500–1508.

[19] Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. 2016. Making sense of entities and quantities in web tables. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 1703–1712.

[20] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic web* 6, 2 (2015), 167–195.

[21] Oliver Lehmberg and Christian Bizer. 2017. Stitching web tables for improving matching quality. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1502–1513.

[22] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on WWW*. 75–76.

[23] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1338–1347.

[24] Colin Lockard, Xin Luna Dong, Arash Einolghozati, and Prashant Shiralkar. 2018. CERES: Distantly Supervised Relation Extraction from the Semi-Structured Web. *Proceedings of the VLDB Endowment* 11, 10 (June 2018), 1084–1096. https://doi.org/10.14778/3231751.3231758

[25] Colin Lockard, Prashant Shiralkar, Xin Dong, and Hannaneh Hajishirzi. 2020. Web-scale Knowledge Collection. *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020).

[26] Colin Lockard, Prashant Shiralkar, X. Dong, and Hannaneh Hajishirzi. 2020. ZeroShotCeres: Zero-Shot Relation Extraction from Semi-Structured Webpages. In *ACL*.

[27] Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. 3047–3056.

[28] Mausam Mausam. 2016. Open Information Extraction Systems and Downstream Applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA) *(IJCAI'16)*. AAAI Press, 4074–4077.

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[30] Varish Mulwad, Tim Finin, and Anupam Joshi. 2013. Semantic message passing for generating linked data from tables. In *International Semantic Web Conference*. Springer, 363–378.

[31] Varish Mulwad, Tim Finin, Zareen Syed, Anupam Joshi, et al. 2010. Using linked data to interpret tables. In *Proceedings of the the First International Workshop on Consuming Linked Data*.

[32] 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*. 533–542.

[33] Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. 2019. MTab: matching tabular data to knowledge graph using probability models. (2019). arXiv:1910.00246

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

[35] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[36] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*. 1–6.

[37] Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. 2016. Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *Proceedings of the 25th International Conference on WWW*. 251–261.

[38] Yoones A Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. 2014. Knowledge Base Augmentation using Tabular Data.. In *LDOW*.

[39] Adelene YL Sim and Andrew Borthwick. 2018. Record2Vec: unsupervised representation learning for structured records. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1236–1241.

[40] Bram Steenwinckel, Gilles Vandewiele, Filip De Turck, and Femke Ongenae. 2019. Csv2kg: Transforming tabular data into semantic knowledge. *SemTab, ISWC Challenge* (2019).

[41] Kunihiro Takeoka, Masafumi Oyamada, Shinji Nakadai, and Takeshi Okadome. 2019. Meimei: An efficient probabilistic approach for semantically annotating tables. In *Proceedings of the AAAI Conference*, Vol. 33. 281–288.

[42] 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 *Advances in neural information processing systems*. 5998–6008.

[43] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering Semantics of Tables on the Web. *Proceedings of the VLDB Endowment* 4, 9 (2011).

[44] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.
[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
[46] Keyulu Xu, Weihua Hu, J. Leskovec, and S. Jegelka. 2019. How Powerful are Graph Neural Networks? *ArXiv* abs/1810.00826 (2019).
[47] Pengcheng Yin, G. Neubig, W. Yih, and S. Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *ACL*.
[48] Wenhao Yu, Zongze Li, Qingkai Zeng, and Meng Jiang. 2019. Tablepedia: Automating pdf table reading in an experimental evidence exploration and analytic system. In *The World Wide Web Conference*. 3615–3619.
[49] Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2Vec: neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1029–1032.
[50] Shuo Zhang and Krisztian Balog. 2020. Web Table Extraction, Retrieval, and Augmentation: A Survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 2 (2020), 1–35.
[51] Ziqi Zhang. 2017. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web* 8, 6 (2017), 921–957.