Structure-aware Pre-training for Table Understanding with Tree-based Transformers

Zhiruo Wang*
Carnegie Mellon University
zhiruow@andrew.cmu.edu

Haoyu Dong* †
Microsoft Research
hadong@microsoft.com

Ran Jia
Microsoft Research
jia.ran@microsoft.com

Jia Li
Peking university
lijiaa@pku.edu.cn

Zhiyi Fu
Peking University
ypfzy@pku.edu.cn

Shi Han
Microsoft Research
shihan@microsoft.com

Dongmei Zhang
Microsoft Research
dongmeiz@microsoft.com

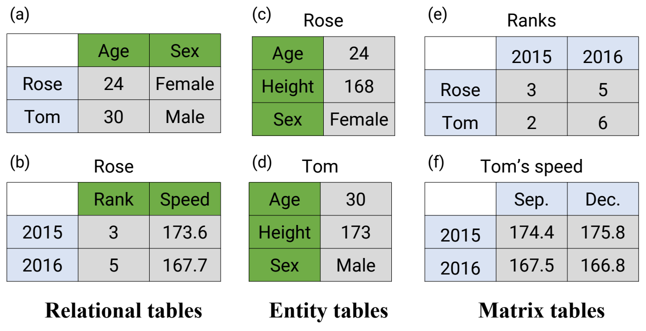
ABSTRACT

Tables are widely used with various structures to organize and present data. Recent attempts on table understanding mainly focus on relational tables, yet overlook to other common table structures. In this paper, we propose TUTA, a unified pre-training architecture for understanding generally structured tables. Since understanding a table needs to leverage both spatial, hierarchical, and semantic information, we adapt the self-attention strategy with several key structure-aware mechanisms. First, we propose a novel tree-based structure called a bi-dimensional coordinate tree, to describe both the spatial and hierarchical information in tables. Upon this, we extend the pre-training architecture with two core mechanisms, namely the tree-based attention and tree-based position embedding. Moreover, to capture table information in a progressive manner, we devise three pre-training objectives to enable representations at the token, cell, and table levels. TUTA pre-trains on a wide range of unlabeled tables and fine-tunes on a critical task in the field of table structure understanding, i.e. cell type classification. Experiment results show that TUTA is highly effective, achieving state-of-the-art on four well-annotated cell type classification datasets.

ACM Reference Format:
Zhiruo Wang*, Haoyu Dong* †, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. . Structure-aware Pre-training for Table Understanding with Tree-based Transformers. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages.

1 INTRODUCTION

Table, as a key structure to organize and present data, is widely used for documentations in webpages, spreadsheets, PDFs, etc. Unlike sequenced NL text, tables usually lay cells in a two-dimensional matrix and contain additional information like format and formula besides pure text. As Figure 1 shows, **tables are flexible with various structures**, including vertical and horizontal relational tables, vertical and horizontal entity tables, matrix tables, etc [8, 28]. To present data more effectively in the two-dimensional space, real-world tables often have hierarchical structures and expandable data groups [2, 3, 13]. For example, Figure 2 (b) shows a matrix PDF table identified by a hierarchical left header with different indentation levels, and Figure 2 (c) shows a matrix spreadsheet table that has both hierarchical top and left headers ("B1:E2" and "A3:A22"), shaped by merged cells and indents respectively, to organize data in a compact and hierarchical way for easy look-up or side-by-side comparison.


(a)

	Age	Sex
Rose	24	Female
Tom	30	Male

(b) Rose

	Rank	Speed
2015	3	173.6
2016	5	167.7

Relational tables

(c) Rose

Age	24
Height	168
Sex	Female

(d) Tom

Age	30
Height	173
Sex	Male

Entity tables

(e) Ranks

	2015	2016
Rose	3	5
Tom	2	6

(f) Tom's speed

	Sep.	Dec.
2015	174.4	175.8
2016	167.5	166.8

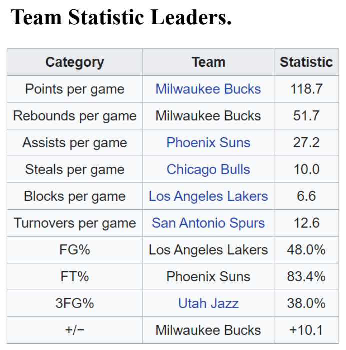
Matrix tables

Figure 1: Examples of various structured tables. Left: (a,b) vertical relational, Middle: (c,d) horizontal entity, and Right: (e,f) matrix tables. Different colors denote different cell types.

Tables include a huge amount of high-quality data and recently gain increasing attention. There is a flurry of research on table understanding tasks, including entity linking in tables [1, 9, 33, 44], column type identification in tables [5, 18], answering natural language questions over tables [23, 31, 35], and generating data analysis for tables [46]. **However, the vast majority of these works only focuses on relational tables** that only account for 0.9% of

* Equal contribution. Work done while Zhiruo Wang was a full-time intern at Microsoft Research. † Contact person.

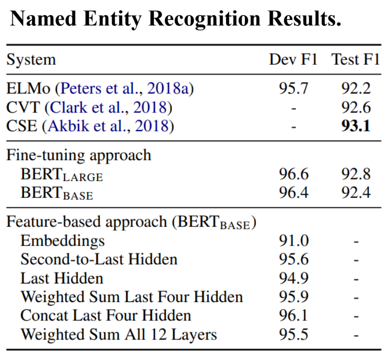
Conference'17, July 2017, Washington, DC, USA
© Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00



Team Statistic Leaders.

Category	Team	Statistic
Points per game	Milwaukee Bucks	118.7
Rebounds per game	Milwaukee Bucks	51.7
Assists per game	Phoenix Suns	27.2
Steals per game	Chicago Bulls	10.0
Blocks per game	Los Angeles Lakers	6.6
Turnovers per game	San Antonio Spurs	12.6
FG%	Los Angeles Lakers	48.0%
FT%	Phoenix Suns	83.4%
3FG%	Utah Jazz	38.0%
+/−	Milwaukee Bucks	+10.1

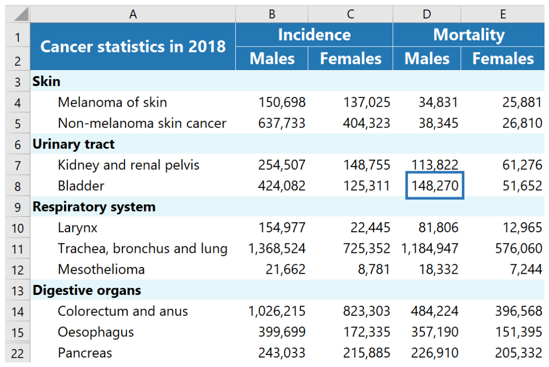
(a) A relational web table



Named Entity Recognition Results.

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

(b) A matrix PDF table



	A	B	C	D	E
1	Cancer statistics in 2018	Incidence		Mortality	
2		Males	Females	Males	Females
3	**Skin**				
4	Melanoma of skin	150,698	137,025	34,831	25,881
5	Non-melanoma skin cancer	637,733	404,323	38,345	26,810
6	**Urinary tract**				
7	Kidney and renal pelvis	254,507	148,755	113,822	61,276
8	Bladder	424,082	125,311	148,270	51,652
9	**Respiratory system**				
10	Larynx	154,977	22,445	81,806	12,965
11	Trachea, bronchus and lung	1,368,524	725,352	1,184,947	576,060
12	Mesothelioma	21,662	8,781	18,332	7,244
13	**Digestive organs**				
14	Colorectum and anus	1,026,215	823,303	484,224	396,568
15	Oesophagus	399,699	172,335	357,190	151,395
22	Pancreas	243,033	215,885	226,910	205,332

(c) A matrix spreadsheet table

Figure 2: Examples of real-world tables. (a) shows a relational web table with a flat top header. (b) shows a matrix PDF table with a flat top header and a hierarchical left header. (c) shows a spreadsheet table that has hierarchical top and left headers.

web tables[1] and 22.0% of spreadsheet tables [3]. Also, other widely-used table types such as matrix tables and entity tables are largely overlooked. Both problems lead to a huge gap between cutting-edge table understanding techniques and real-world tables in varied structures. Therefore, it is significant to enable table semantic understanding in variously structured tables and make a critical step to mitigate this step. There are several attempts [2, 3, 11, 16, 22] in identifying table hierarchies and cell types to extract relational data from variously structured tables, but **labeling such structural information is very time-consuming and labor-intensive**, thus greatly challenges machine learning methods which are rather data-hungry.

Motivated by the success of large-scale pre-trained language models (LMs) [10, 32] in a variety of NL tasks, one potential way to mitigate the label-shortage challenge is leveraging weakly supervised pre-training using large volumes of unlabeled tables on the web. [19, 42] target question answering over relational tables via joint pre-training of tables and their textual descriptions. [9] attempts to pre-train embeddings on relational tables to enhance table knowledge matching and table augmentation. However, **these pre-training methods still only target relational web tables** because the structure is simple and clear. In relational tables, each column is homogeneous and described by its column name, so [42] augments each data cell with its corresponding column name, and [9] enforces each cell only aggregate information from its row and column. But these methods are not suitable for other commonly used table types. For example, Figure 2 (c) shows a hierarchical matrix table, where cell "D8" (148,270) is not only described by its top header, but also described by its left header. Furthermore, indicated by merged cells in the top header and indentation levels in the left header, cell "D8" is jointly described by multiple cells including "D1" ("Mortality"), "D2" ("Males"), "A6" ("Urinary tract") and "A8" ("Bladder"), and constructs a relational mapping ("Mortality", "Males", "Urinary tract", "Bladder", 148,270) [3]. Such structural information greatly helps human readers to understand this table, but simply treating it as a relational table will lose the valuable structural information.

Hence in this paper, we aim to propose a structure-aware method for general table pre-training. Fortunately, previous studies show strong structural commonalities in real-world tables: **(1)** tables are arranged in a two-dimensional format with vertical orientation, horizontal orientation, or both orientations [28]; **(2)** tables usually contain headers on the top or left side to describe other cells [40, 43]; **(3)** headers are often structured using a hierarchical tree [2, 3, 26], especially in professional finance and government tables[2]. Motivated by previous studies, we propose a bi-dimensional coordinate tree, a unified structure to describe both cell locations and hierarchies in variously structured tables. **Based on this novel tree-based structure, we propose TUTA, a structure-aware architecture for Table Understanding with Tree-based Attention. TUTA consists of two key enhancements to capture structural information in tables. (1)** To encode spatial and hierarchical information in TUTA, we devise tree-based positional encodings. Different from explicit tree-based positional encodings using a uni-dimension tree structure [34], TUTA compares both explicit and implicit positional encodings with the bi-dimensional tree structure. Also, to encode spatial and hierarchical information jointly, TUTA combines tree-based coordinates with traditional rectangular Cartesian coordinates. **(2)** Self-attention [37] is considered to be more parallelizable and efficient than dedicated structure-aware models like tree-LSTMs and RNNs [27, 34], however, is challenged by numerous "distraction"s in large bi-dimensional tables. Structural information is crucial for local cells in both aggregating structural relevant contexts and ignoring noisy contexts. To enable greater efficiency in both processes, we, therefore, uniquely devise a structure-aware attention mechanism. Different from existing practices that use bottom-up tree-based attention [27] and constituent tree attention [39] in the NL domain, TUTA adapts the general idea of graph attention networks [38] to tree structures. Through the bi-dimensional tree structure, data readily flows in top-down,

[1]http://webdatacommons.org/webtables/index.html

[2]SAUS (Statistical Abstract of US from the Census Bureau) is a widely studied public spreadsheet dataset, in which most tables contain hierarchical headers.

Table 1: Dataset comparison between pre-training methods.

	Document types	Table types	Total amount
TUTA	WikiTable	General	2.62 million
	WDC		50.8 million
	Spreadsheet		4.49 million
TAPAS	WikiTable	Relational	Not published
TaBERT	WikiTable	Relational	1.3 million
	WDC		25.3 million
TURL	WikiTable	Relational	0.57 million

bottom-up, and peer-to-peer manners. Key contributions of this paper summarize as follows:

- For generally structured tables, we propose in this paper a novel bi-dimensional coordinate tree to describe both the spatial and hierarchical information. Based on this bi-tree, we propose TUTA, a structure-aware pre-training method following a self-attention strategy. To better incorporate spatial and structural information in tables, we devise two crucial techniques, called tree positional encodings and tree-based attention, that prove to be highly effective throughout experiments.
- We leverage three novel pre-training tasks for TUTA, including token-level masked language modeling, cell-level cloze, and table-level context retrieval. During pre-training, TUTA progressively learns token/cell/table representations on a large volume of tables in an unsupervised manner.
- To demonstrate the effectiveness of TUTA, we fine-tune our pre-trained model on a critical task for table semantic structure understanding, cell type classification. TUTA is the first transformer-based method applied to this task and achieves state-of-the-art performance on four well-annotated datasets.

2 PRELIMINARIES

2.1 Dataset construction

In this paper, we extend the amount and diversity of tables in two ways. (1) In addition to web tables, spreadsheet tables are also widely used[3], especially for professional usage in government, finance, and medical domains. Thus in this paper, we additionally build a large-scale table corpus of web-crawled spreadsheets. (2) In addition to relational tables, other diversely structured tables, such as matrix tables and entity tables [28] are also commonly used to store high-quality tabular data. Hence in this paper, we include various structured tables into our pre-training corpus.

We collect web tables from Wikipedia (WikiTable) and the WDC WebTable Corpus [25], meanwhile, build a large-scale web-crawled spreadsheet corpus including 115 million tables. The data collection and pre-processing details can be found in **Appendix A**. We further unify the feature schema for web tables and spreadsheet tables as shown in Table 7 in **Appendix A**. After pre-processing, the total amount of tables in the combined corpus is 57.9 million. To capture diverse table structures and data characteristics from our table corpus, we iterate three datasets in parallel during the pre-training process.

2.2 Bi-dimensional coordinate tree

As introduced in Section 1, tables usually employ top and left headers to describe other cells [40, 43]. For example, vertical/horizontal tables often contain top/left headers, while matrix tables usually contain both [28]. A table is recognized as **hierarchical** if its header on the top or left exhibits a hierarchical tree structure of at least two layers [3, 26]. In contrast, **flat** tables refer to those without any hierarchical structure. Motivated by existing definitions, we propose a bi-dimensional coordinate tree to jointly describe the cell locations and hierarchies in tables.

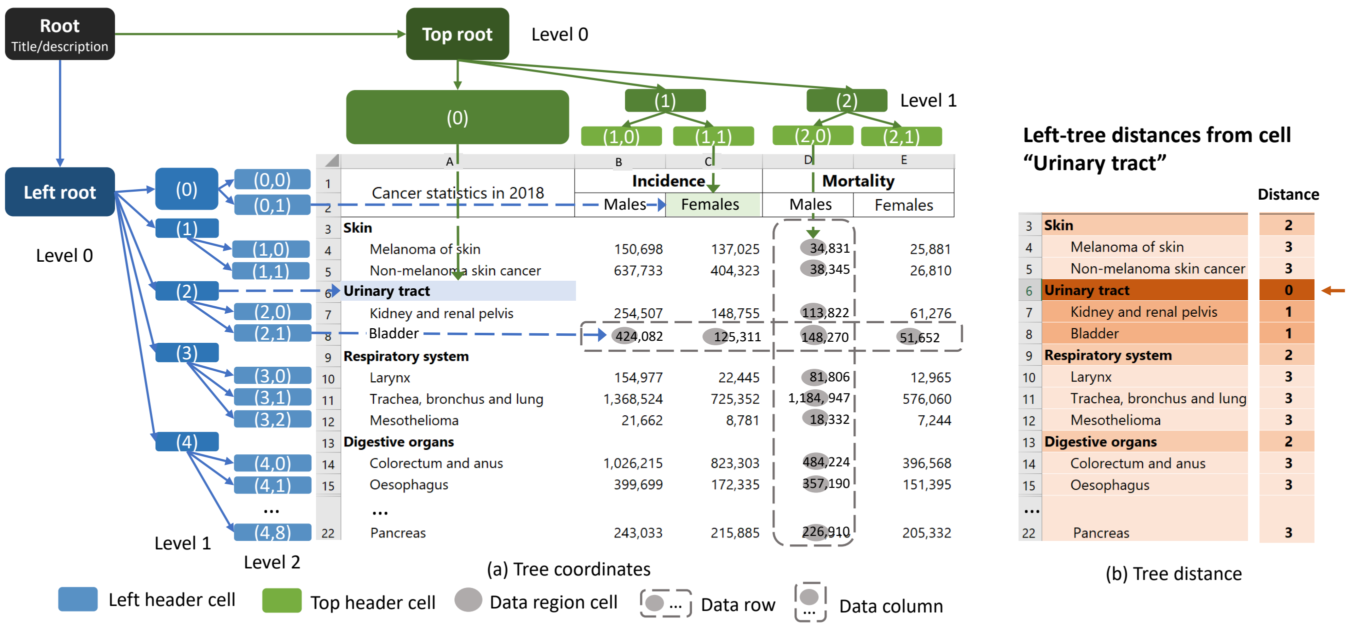
Tree-based position The bi-dimensional coordinate tree is defined to be a directed tree with ordered lists of children, where each node has a unique parent and an ordered finite list of children. It contains two perpendicular subtrees, namely a top tree and a left tree. In the left tree, each node's position can be defined from its path from the left root node. And the top-tree works similarly. Next, given the positions of corresponding tree nodes on top and left, each cell in the table is uniquely presented in **bi-dimensional tree coordinates**. For example, as shown in Figure 3 (a), the top and left coordinates of cell "D8" (148,270) are (2,0) and (2,1) respectively; the left coordinate of cell "A6" ("Urinary tract") is (2) since it corresponds to the parent node of the third subtree in the left header.

Tree-based distance In each tree, the relation between two nodes is a path, i.e., a series of steps along tree branches. Each step either moves up to the parent or goes down to a child. Motivated by [34], we define the **top/left tree distance** for every two nodes to be the step number of the shortest path between them, and upon this, the **bi-tree distance** of two cells as the **sum** of their left tree distance and top tree distance. Figure 3 (b) shows the distances from cell "A6" ("Urinary tract") in the left tree. Cell "A6" ("Urinary tract")is the parent node of cell "A8" ("Bladder"), so they have a very short left-tree distance of 1. Cell "A6" ("Urinary tract") and cell "A10" ("Larynx"), however, have a rather long distance of 3, since cell "A6" ("Urinary tract") is the "uncle node" of the cell "A10" ("Larynx"). Furthermore, cell "A6" ("Urinary tract") and cell "C2" ("Females") have an even longer distance of 6 (both the distances on the top and left are 3), indicating that they are only remotely related. Specifically, distances from table descriptions (e.g. table title, table caption, page title, text segments before/after a table) to any table cell are set to 0, since table descriptions contain global information of a table. The tree-based definition of distance enables effective spatial and hierarchical dataflow via our proposed tree attention in Section 3.3.

Note that the bi-dimensional coordinate tree is quite general for both hierarchical tables and flat tables. In flat tables, the top and left tree coordinates **degenerate** to rectangular Cartesian coordinates. Hence, tree distance between two cells in the same row or column is 2, while between two cells from different rows or columns, is 4.

Tree extraction Existing header detection methods [11, 14] have already achieved desirable accuracy for various structured tables, so we adopt the method introduced by [11] for header detection.

[3]By estimation, there are around 800 million spreadsheet users. https://medium.com/grid-spreadsheets-run-the-world/excel-vs-google-sheets-usage-nature-and-numbers-9dfa5d1cadbd



	A	B	C	D	E
1	Cancer statistics in 2018	Incidence		Mortality	
2		Males	Females	Males	Females
3	**Skin**				
4	Melanoma of skin	150,698	137,025	34,831	25,881
5	Non-melanoma skin cancer	637,733	404,323	38,345	26,810
6	**Urinary tract**				
7	Kidney and renal pelvis	254,507	148,755	113,822	61,276
8	Bladder	424,082	125,311	148,270	51,652
9	**Respiratory system**				
10	Larynx	154,977	22,445	81,806	12,965
11	Trachea, bronchus and lung	1,368,524	725,352	1,184,947	576,060
12	Mesothelioma	21,662	8,781	18,332	7,244
13	**Digestive organs**				
14	Colorectum and anus	1,026,215	823,303	484,224	396,568
15	Oesophagus	399,699	172,335	357,190	151,395
	...				
22	Pancreas	243,033	215,885	226,910	205,332

(a) Tree coordinates

		Distance
3	**Skin**	2
4	Melanoma of skin	3
5	Non-melanoma skin cancer	3
6	**Urinary tract**	0
7	Kidney and renal pelvis	1
8	Bladder	1
9	**Respiratory system**	2
10	Larynx	3
11	Trachea, bronchus and lung	3
12	Mesothelioma	3
13	**Digestive organs**	2
14	Colorectum and anus	3
15	Oesophagus	3
...		
22	Pancreas	3

(b) Tree distance

Figure 3: An example of bi-dimensional coordinate tree. In this example, both the top tree and the left tree contain three levels. Cell "A6" ("Urinary tract") is the "parent" node of cell "A7" and cell "A8", and has "brothers" including "A3", "A9" and "A13".

For those detected header regions, we implement a rule-based strategy to extract header hierarchies by consolidating effective expert-engineered heuristics. We extract the hierarchy trees based on the merged cells in the top header, indentation levels in the left header, and formulas in the data region. To be specific, we build a hierarchical relationship in the top header between a merged cell and those cells beneath it. It works similarly in the left header, for a cell and its subsequent cells with deeper indentation levels. Note that formulas having "SUM" or "AVERAGE" are also strong indications of hierarchy, so we incorporated them into our algorithm as well. This approach exhibits high efficiency and desirable explainability at processing large unlabeled corpus. Since TUTA is a general table pre-training framework based on bi-dimensional trees, one can also employ other techniques such as [3, 26, 30] to extract hierarchies for TUTA. More details about tree extraction can be found in **Appendix B**.

3 TUTA MODEL

Our model's architecture adapts from BERT's encoder with four key enhancements: (1) building the first dedicated vocabulary for table understanding domain to better encode commonly used tokens in real-world tables. (2) leveraging tree-based positional embeddings to better encode cell locations and hierarchies. (3) proposing a structure-aware attention mechanism to help cells attend to structurally neighboring contexts efficiently. (4) devising three pre-training objectives to enable representation learning at the token, cell, and table levels to capture table information in a progressive manner. The architecture overview is shown in Figure 4.

3.1 Vocabulary construction

Different from long texts in NL documents, cell strings in tables often have short lengths and concise semantics, which makes the word distribution in tables very different from that in NL. For example, tables often include lots of measures such as "quantity" and "yards", or statistical terms such as "average" and "difference". To emphasize and being more compact, tables often use a lot of abbreviations such as "num" and "qtr". Hence it is not appropriate to directly use NL vocabulary when parsing cell strings.

In this paper, we build the first vocabulary for general tables, which includes widely used tokens in real-world tables. Based on the table corpus introduced in Section 2.1, we build our vocabulary using the same WordPiece model with BERT [10] and get 9,754 new tokens compared with the BERT vocabulary. We directly list the top 50 newly added tokens in Table 2 and try to manually classify them into different semantic groups, which shows meaningful and intuitive results. Intriguingly, there are indeed plenty of abbreviations in the dedicated vocabulary, e.g., "pos", "fg" and "pts" in the sports domain. Since NL-form table descriptions (e.g., titles) also need to be modeled, we merge our table vocabulary (top 996 tokens) with the NL vocabulary to achieve better generosity. Based on the merged vocabulary, we perform sub-tokenization for table contexts and cell strings using the Wordpiece tokenizer [10].

3.2 Embedding layer

Table cells encode rich information. Whether such information is well extracted and leveraged can lead to remarkable differences in table understanding tasks [12, 16]. Since transformer-based methods need to first linearize the input into a sequence of tokens, we

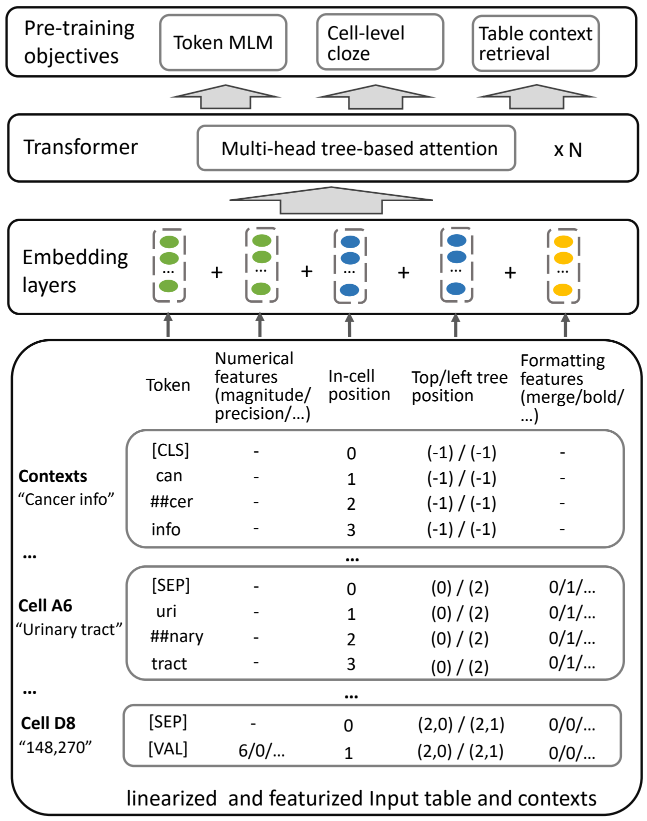


Figure 4: Overview of TUTA architecture. It contains (1) embedding layers to convert an input table into input embeddings, (2) N stacked transformer layers to capture table semantics and structures, and (3) final projection layers for pre-training objectives.

Table 2: Examples of the top 50 newly added tokens in TUTA. We rank them based on their frequencies and try to sort them into several groups based on our understanding.

Measures & units	num (number), qty (quantity), yds, dist, att, eur...
Statistical terms	avg (average), pct (percentage), tot, chg, div, rnd...
Date & time	fy (fiscal year), thu (Thursday), the, fri, qtr, yr...
Sports	pos (position), fg (field goal), pim, slg, obp...
Unsorted	pld, nd, ast, xxl, px, blk, xxs, ret, lmsc, stl, ef, wkts, fga, wm, adj, tweet, comp, mdns, ppg, bcs, sog, chr, xs, fta, mpg, xxxl, sym, url, msrp, lng

perform sub-tokenization for each cell using the Wordpiece tokenizer and then flatten a table to a sequence. We add a special $[SEP]$ token for each cell and a special $[CLS]$ token for table textual descriptions. As shown in Figure 4, we extract additional key information from each cell including numbers, positions and formats, then construct embeddings for each token as follows:

Tree-based positional embedding As introduced in 2.2, each table cell has a pair of tree coordinates to capture hierarchical

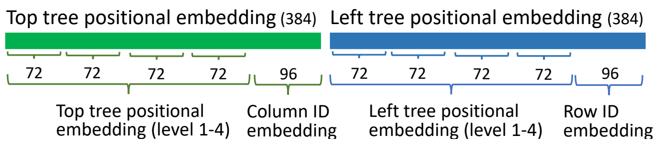

Top tree positional embedding (384)
Left tree positional embedding (384)
72
72
72
72
96
72
72
72
72
96
Top tree positional embedding (level 1-4)
Column ID embedding
Left tree positional embedding (level 1-4)
Row ID embedding

Figure 5: Tree-based positional embedding (total length is 768 in our setting).

information. To unify representation for tables in different tree depths, we extend top/left tree coordinates to the same length (in this paper, the maximum length is 4) by padding with -1. To further incorporate absolute cell positions, column and row indexes are respectively concatenated with top and left tree coordinates. Take the cell "D8" in Figure 3 as an example, its original top coordinate (2,0) and a left coordinate (2,1) are extended to (2,0,-1,-1,3) and (2,1,-1,-1,7), respectively. As shown in Figure 5, we assign each level of top and left coordinates with a sub-embedding, then concatenate them together as a joint position embedding. We also compare two methods to embed this compositional position. For one, we experiment with explicit tree embeddings as introduced by [34], it has good explainability and can be readily-adapted to our bi-dimensional tree. While for another, we use randomly initialized implicit embeddings and jointly train it with encoding layers as [10].

Internal positional embedding Internal position is the index of a token in the sub-tokenized cell (in this paper, the maximum internal index is 8). Each internal position is assigned with a learnable embedding.

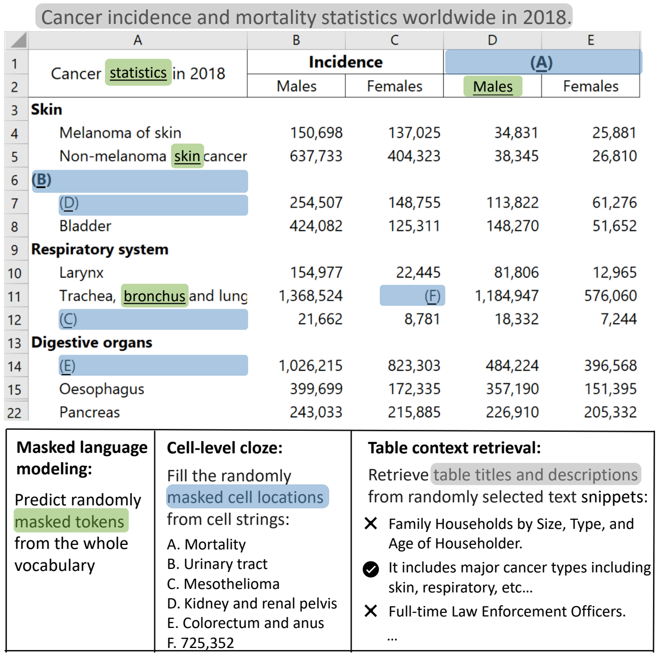
Token and number embedding Since the token vocabulary is a finite set, each token in the vocabulary can be directly assigned with a learnable embedding. But different from tokens, numbers constitute an infinite set, so we assign each number with a special $[VAL]$ token and extract four discrete features including the numbers' magnitude, precision, the first digit, and the last digit. We assign each discrete feature with a learnable sub-embedding (1/4 of the total embedding length), then concatenate them to an aggregated number embedding.

Format embedding Format helps humans to understand tables intuitively, so we add format embeddings that marks whether a cell has vertically/horizontally merging, top/left /bottom/right border, formula, font bold, non-white background color, and non-black font color. A simple multilayer perceptron (MLP) is used to map these cell-level formatting features to a joint format embedding.

3.3 Tree attention

Based on definitions of the bi-dimensional tree coordinate and distance (Section 2.2), tree attention can be naturally introduced in this section. Self-attention is considered to be more parallelizable and efficient than dedicated structure-aware models like tree-LSTMs and RNNs [27, 34], but is challenged by lots of "distraction" in large two-dimensional tables, because in its most general formulation, self-attention allows every token to attend to every other token, dropping all structural information (e.g., whether two tokens appear in the same cell, appear in the same row/column, or have a hierarchical relationship). Since spatial and hierarchical information is highly important for local cells to aggregate their structural related contexts and ignore unrelated or noisy contexts, we devise a structure-aware attention mechanism to help local cells to attend to their structural neighboring cells efficiently. Motivated by the general idea of graph attention networks [38], we inject the tree structure into the attention mechanism by performing masked attention — we only set $Mi, j = 1$ for cells j belonging to Ci , where Ci is a structural neighborhood of cell i in the bi-tree, and Mi, j is a symmetric binary matrix to indicate visibility between cell i and cell j.

Based on the proposed tree distance in Section 2.2, structural neighboring cells Ci can be filtered out by a predefined distance threshold D. Smaller D can make local cells more "focus", but when D is large enough (16 is the maximum distance in our tree definition), tree attention will perform in the same way with global attention. Different from existing practice such as bottom-up tree-based attention [27] and constituent tree attention [39] in NL domain, our encoder with tree attention enables both top-down, bottom-up and peer-to-peer data flow in the tree structure. **Sicne TUTA has N stacked encoders with tree attention, each cell can represent information from its neighborhood with nearly arbitrary depth (N×D).** In experiments, we implement ablation studies to compare different D values.



Cancer incidence and mortality statistics worldwide in 2018.

	A	B	C	D	E
1	Cancer statistics in 2018	Incidence		(A)	
2		Males	Females	Males	Females
3	**Skin**				
4	Melanoma of skin	150,698	137,025	34,831	25,881
5	Non-melanoma skin cancer	637,733	404,323	38,345	26,810
6	(B)				
7	(D)	254,507	148,755	113,822	61,276
8	Bladder	424,082	125,311	148,270	51,652
9	**Respiratory system**				
10	Larynx	154,977	22,445	81,806	12,965
11	Trachea, bronchus and lung	1,368,524	(F)	1,184,947	576,060
12	(C)	21,662	8,781	18,332	7,244
13	**Digestive organs**				
14	(E)	1,026,215	823,303	484,224	396,568
15	Oesophagus	399,699	172,335	357,190	151,395
22	Pancreas	243,033	215,885	226,910	205,332

Masked language modeling:

Predict randomly masked tokens from the whole vocabulary

Cell-level cloze:

Fill the randomly masked cell locations from cell strings:

A. Mortality
B. Urinary tract
C. Mesothelioma
D. Kidney and renal pelvis
E. Colorectum and anus
F. 725,352

Table context retrieval:

Retrieve table titles and descriptions from randomly selected text snippets:

✗ Family Households by Size, Type, and Age of Householder.
✓ It includes major cancer types including skin, respiratory, etc...
✗ Full-time Law Enforcement Officers.
...

Figure 6: An intuitive example for pre-training objectives.

3.4 Pre-training objectives

Tables naturally have progressive levels — token level, cell level, and table level. To capture table information in such a progressive manner, we devise novel cell-level and table-level objectives in addition to commonly used token-level objective.

Masked language modeling (MLM) MLM [10, 24] is widely used for NL pre-training. Motivated by [19], we train token representations by predicting masked tokens in both table cells and text segments. We further adapt the masking strategy by: (1) randomly selecting 15% of cells; (2) randomly masking one token in 70% of selected cells, and masking all tokens in the other 30% selected cells. Strategy (2) is mainly used to learn contextual information from neighboring cells. MLM is modeled as a multi-classification problem for each masked token with the cross-entropy loss $\mathcal{L}_{\text{mlm}}$.

Cell-level Cloze (CLC) Cells are the basic units in tables to record text, position and format. So, cell-level representations are important for various tasks such as cell type classification, entity linking and table question answering. Existing methods [9, 19, 42] directly take the averaged token representation to represent cells, but lack a systematic pre-training objective for cells in a whole way. In this paper, we devise a novel cell-level task by randomly masking some selected cells in table headers (with higher probability in our setting) and data regions (with lower probability) and encourage the model to retrieve the correct cell strings based on their locations. This task can be regarded as a cell-level one-to-one mapping problem from cell locations to cell strings as shown in Figure 6. Based on embeddings of leading tokens of selected cell locations and cell strings, $[SEP]_{\text{celllocation}}$ and $[SEP]_{\text{cellstring}}$, we use a dot-product attention module [37] to compute the mapping probabilities. Then we model CTC as a multi-classification task for each cell location with the cross-entropy loss $\mathcal{L}_{\text{clc}}$. Note that in our attention mechanism, locations of selected cells can still "see" their structural contexts in tables while strings of selected cells can only "see" its internal tokens via attention masks.

Around 20% of the cells are randomly selected for each table. To leverage more training on structural information, we apply two strategies during cell selection. In one way, we randomly choose cells on the same sub-tree based on our bi-tree structure, while in another, cells are randomly taken from different top header rows and left header columns.

Table context retrieval (TCR) Table level representations are important in two aspects. On the one hand, table titles and descriptions can help one to better understand table cells. On the other hand, all cells in a table constitute an overall meaning of this table. To both ends, we propose a novel objective to capture table-level representation using the leading token $[CLS]$. We split table titles and descriptions into text segments, then, provide each table with positive segments from its own contexts and negative ones from others' context. Each table has at most three positive segments and three negative segments, from which the $[CLS]$ needs to retrieve its correct belongings. We assign one randomly selected positive segments to follow the leading token $[CLS]$, assign other positive and negative segments with leading $[SEP]$s. Then, TCR can be viewed as a table-level one-to-many mapping problem from $[CLS]$ to $[SEP]$s of text segments. It models as a binary classification problem for each pair of $[CLS]$ and $[SEP]$ with cross-entropy loss $\mathcal{L}_{\text{tcr}}$. Note that with our attention mechanism, $[CLS]$ learns table-level representations and can "see" all table cells, while the $[SEP]$ of each text segment learns text segment representations and only "see"s its internal tokens.

The final objective is the summarization of $\mathcal{L}_{\text{mlm}}$, $\mathcal{L}_{\text{clc}}$, and $\mathcal{L}_{\text{tcr}}$ with the same weight.

3.5 Pre-training details

Data processing configuration In our setting, top/left tree structures embed in a maximum depth of 4, as nodes hang deeper onto the tree, we allow an increasing number of degrees from 32, 32, 64, to 256 for the 1st, 2nd, 3rd, and 4th tree levels, respectively. The largest supporting degree, 256, accords with both the maximum number of rows and columns to incorporate large relational tables. When encountering large tables in the downstream task, we can split the table into several smaller ones (share the same top or left header) based on detected table headers. Table corpus for pre-training is a mixture of spreadsheets and web tables. Since different datasets have different structure distributions and data characteristics, in the pre-training process, we feed table samples from these datasets (WikiTable, WDC and Spreadsheet) in parallel to our model to learn from diverse tables simultaneously. Since the size of spreadsheet corpus and WikiTable corpus are not as big as WDC, they will be cycled for several times in the pre-training process.

Tables are serialized into token sequences in turns of rows, in which each row of cell strings are tokenized and concatenated with jointing $[SEP]$ tokens. Note that cells in data region often express similar numerical information yet introduce limited semantics, therefore, they are randomly sampled out in the pre-training process, and we adopt heuristics to categorize data region cells into text- and value- dominated types, sampling out 50% in the former and 90% in the latter. We bound cells in 8 tokens and allow 64 tokens for context pieces.

Model configuration TUTA is a N-layer Transformer encoder with hidden size H, in which each layer performs self-attention with A heads. We align hyperparameters with BERT$_{BASE}$ (N = 12, H = 768, A = 12) and initialize with token embeddings and encoder of BERT before adapting to the tabular domain. TUTA first pre-trains with table sequences under a maximum of 256 tokens for 1M steps in a batch size of 12, then, it extends the supporting sequence length to 512 and continues training for another 1M steps with a batch size of 4 (totally going through 16M tables). We implement the above procedures with PyTorch [4] distributed training. We train TUTA and its variants on 64 Tesla V100 GPUs, and each TUTA variant takes nine days on four Tesla V100 GPUs.

4 EXPERIMENTS

Understanding the semantic structures of tables is the initial and critical step for plenty of tasks on tables. One critical task towards understanding table semantic structures is to identify the structural type of table cells. Cell type classification (CTC) is a widely studied task in table structure understanding domain [11, 16, 17, 22] with several well-annotated datasets. Therefore, we use cell type classification to validate the effectiveness of TUTA on understanding various structured tables.

Datasets Existing CTC datasets includes WebSheet [11], deexcelerator (DeEx) [22], SAUS [16], and CIUS [16]. These datasets are collected from different domains (financial, business, crime, agricultural and health-care) and include tables with various structures. Table 3 shows the statistics on table size and structure for each annotated dataset. Note that these datasets have different definitions on cell types. DeEx, SAUS and CIUS categorize cells into general types including metadata (MD), notes (N), data (D), top attribute (TA), left attribute (LA) and derived (B). To perform automatic table transformation, WebSheet further defines three fine-grained semantic cell types in table headers, namely index, index name and value name [11]. An intuitive example of these definitions is shown in Figure 7, cells of different types play different roles in the relational data extraction process.

Table 3: Statistics of CTC datasets.

	WebSheet	SAUS	DeEX	CIUS
Number of labeled tables	3,503	221	284	248
Number of labeled cells	1075k	192k	711k	216k
Avg. number of rows	33.2	52.5	220.2	68.4
Avg. number of columns	9.9	17.7	12.7	12.7
Avg. number of cells	307	871	2,506	869
Prop. of hierarchical tables	53.7%	93.7%	43.7%	72.1%
Prop. of hierarchical top	35.8%	68.8%	28.9%	46.8%
Prop. of hierarchical left	29.3%	76.0%	29.2%	30.2%

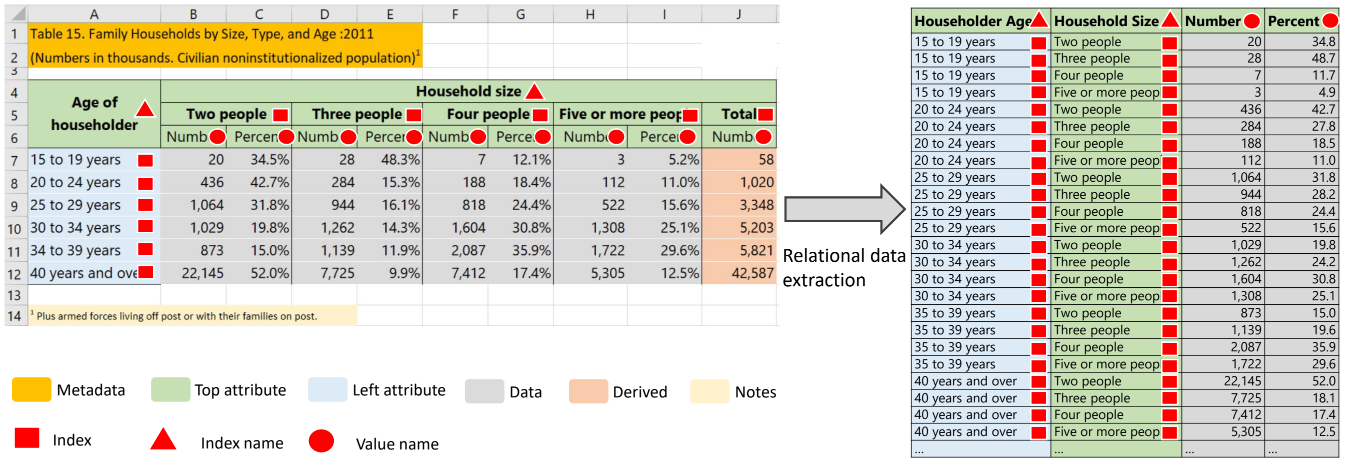
Baselines To verify the effectiveness of TUTA, we compare it with four representative baselines. CNN^{BERT} [11] and Bi-LSTM [16] are two state-of-the-art methods for CTC. CNN^{BERT} is a CNN-based method for both cell classification and table range detection with pre-trained BERT embedding. RNN^{C+S} is a bidirectional LSTM-based method for cell classification using pre-trained cell and format embeddings. TAPAS [19] and TaBERT [42] are two recently proposed transformer-based methods in table-text joint pre-training. To ensure an unbiased comparison, we download the pre-trained models of TAPAS and TaBERT, then fine-tune them using the same CTC head and loss function with TUTA. TURL [9] also pre-trains on relational tables but has no datasets and models publicly available, so we have not compared with it in this paper.

4.1 Implementation details

Fine-tune TUTA For CTC task, tables are tokenized, embedded and encoded in the same way as introduced in section 3. Recognizing this task as a cell-level multi-classification problem, we design a fine-tuning head as described below. Continuing with the encoder output of hidden size H, we introduce two linear transformation layers, one with weights $W_1 \in \mathbb{R}^{H \times H}$ and bias $b_1 \in \mathbb{R}^{H}$, another with weights $W_2 \in \mathbb{R}^{H \times C}$ and bias $b_2 \in \mathbb{R}^{C}$ respectively, where C is the number of cell types. To predict the type of each cell, both the leading $[SEP]$ token and the aggregation of other tokens are treated as potential cell-level representations. Given one of these vector representation x, we calculate the prediction distribution as $W_2 \cdot gelu(W_1 \cdot x + b_1) + b_2$, then calculate the cross-entropy loss with the label y. Since we simultaneously fine-tune them in downstream tasks, they have comparable performance. Unless further noted, we always report the accuracy of $[SEP]$ predictors.

Experiment details Following the method of [11] on WebSheet and [16] on DeEx, SAUS and CIUS, we use the same train/validation /test sets for TUTA and all of the baseline methods. When splitting datasets into train, validation and test, we adopt a table-wise, rather than a cell-wise manner. Since cells in a table are always considered

[4]https://pytorch.org/



	A	B	C	D	E	F	G	H	I	J
1	Table 15. Family Households by Size, Type, and Age :2011									
2	(Numbers in thousands. Civilian noninstitutionalized population)[1]									
3										
4	Age of householder ▲	Household size ▲								
5		Two people 🟥		Three people 🟥		Four people 🟥		Five or more peop🟥		Total 🟥
6		Numb●	Percen●	Numb●	Percen●	Numb●	Perce●	Numb●	Perce●	Numb●
7	15 to 19 years 🟥	20	34.5%	28	48.3%	7	12.1%	3	5.2%	58
8	20 to 24 years 🟥	436	42.7%	284	15.3%	188	18.4%	112	11.0%	1,020
9	25 to 29 years 🟥	1,064	31.8%	944	16.1%	818	24.4%	522	15.6%	3,348
10	30 to 34 years 🟥	1,029	19.8%	1,262	14.3%	1,604	30.8%	1,308	25.1%	5,203
11	34 to 39 years 🟥	873	15.0%	1,139	11.9%	2,087	35.9%	1,722	29.6%	5,821
12	40 years and ove🟥	22,145	52.0%	7,725	9.9%	7,412	17.4%	5,305	12.5%	42,587
13										
14	[1] Plus armed forces living off post or with their families on post.									

Householder Age ▲	Household Size ▲	Number ●	Percent ●
15 to 19 years 🟥	Two people 🟥	20	34.8
15 to 19 years 🟥	Three people 🟥	28	48.7
15 to 19 years 🟥	Four people 🟥	7	11.7
15 to 19 years 🟥	Five or more peop🟥	3	4.9
20 to 24 years 🟥	Two people 🟥	436	42.7
20 to 24 years 🟥	Three people 🟥	284	27.8
20 to 24 years 🟥	Four people 🟥	188	18.5
20 to 24 years 🟥	Five or more peop🟥	112	11.0
25 to 29 years 🟥	Two people 🟥	1,064	31.8
25 to 29 years 🟥	Three people 🟥	944	28.2
25 to 29 years 🟥	Four people 🟥	818	24.4
25 to 29 years 🟥	Five or more peop🟥	522	15.6
30 to 34 years 🟥	Two people 🟥	1,029	19.8
30 to 34 years 🟥	Three people 🟥	1,262	24.2
30 to 34 years 🟥	Four people 🟥	1,604	30.8
30 to 34 years 🟥	Five or more peop🟥	1,308	25.1
35 to 39 years 🟥	Two people 🟥	873	15.0
35 to 39 years 🟥	Three people 🟥	1,139	19.6
35 to 39 years 🟥	Four people 🟥	2,087	35.9
35 to 39 years 🟥	Five or more peop🟥	1,722	29.6
40 years and over 🟥	Two people 🟥	22,145	52.0
40 years and over 🟥	Three people 🟥	7,725	18.1
40 years and over 🟥	Four people 🟥	7,412	17.4
40 years and over 🟥	Five or more peop🟥	5,305	12.5
...	...	...	...

Figure 7: A real example of cell type classification from SAUS. The figure in the left shows the well-annotated cell types. Different colors represent different cell types. The figure in the right shows the relational data contained in this table. It can be seen that different cell types play different roles in the relational data extraction process.

together, none of the cells in test tables have been used for training. For WebSheet, we follow [11] and tune our model for 4 epochs. For DeEx, SAUS and CIUS, since the amount of tables is not as big, we separately tune TUTA on five randomly split folds of data with 100 epochs and report their averaged macro f1. All of the downstream experiments set batch size to 4 and learning rate to 8e-6.

4.2 Experiment results

The macro-F1 score is a commonly used evaluation metric for the overall accuracy of different cell types. As shown in Table 4, TUTA achieves an averaged macro-F1 of 88.1% on four datasets, outperforming all baselines by a large margin (3.6%+). We observe that RNN^{C+S} also outperforms TaBERT and TAPAS. It is probably because RNN^{C+S} has a greater ability to capture spatial information than TAPAS and TaBERT. TAPAS takes spatial information by encoding only the row and column indexes, which, however, is insufficient for hierarchical tables with complicated headers. TaBERT, without joint coordinates from two-dimensions, employs row-wise attention and subsequent column-wise attention. Due to such indirect position encoding, it performs not as well on CTC.

We also list detailed F1-scores for different cell types in Table 5. As Table 5 shows, TUTA achieves the highest score for every cell type in WebSheet. Note that WebSheet, to perform relational data extraction, further defines fine-grained cell types inside table headers. This is a quite challenging task, for table headers often contain complicated hierarchy. Hence, it significantly demonstrates the superiority of TUTA in recognizing fine-grained cell types in table headers.

4.3 Ablation studies

To validate the effectiveness of Tree-based Attention (TA), Position Embeddings (PE) and three pre-training objectives, we evaluate 8 variants of TUTA.

We first start with TUTA-base before explicitly using positions. To test the effectiveness of introducing and varying TA distances,

Table 4: Comparison results of macro-F1 scores.

	WebSheet	DeEx	SAUS	CIUS	Average
CNN^{BERT}	78.4%	60.8%	89.1%	95.1%	80.9%
RNN^{C+S}	79.6%	70.5%	89.8%	97.2%	84.3%
TaBERT-large	79.3%	50.0%	78.9%	92.9%	75.3%
TAPAS-large	82.3%	68.6%	83.9%	94.1%	82.2%
TUTA	**86.6%**	**76.6%**	**90.2%**	**99.0%**	**88.1%**

Table 5: Results of F1 scores on WebSheet by cell types.

	Index name	Index	Value name
CNN^{BERT}	69.9%	86.9%	78.4%
RNN^{C+S}	75.0%	86.6%	77.1%
TaBERT-large	76.8%	85.2%	75.8%
TAPAS-large	74.3%	88.1%	84.6%
TUTA	**83.4%**	**91.6%**	**84.8%**

one variant without TA and three under visible distances 2, 4, 8 are tested.

- TUTA-base, w/o TA: cells are globally visible.
- TUTA-base, TA-8: cells are visible in a distance of 8.
- TUTA-base, TA-4: cells are visible in a distance of 4.
- TUTA-base, TA-2: cells are visible in a distance of 2.

Upon this comparison, we keep TA distance to 2 and augment TUTA-base with positional information, forming TUTA in both implicit and explicit embedding methods.

- TUTA-implicit: implicitly embed positions using trainable weights.
- TUTA-explicit: calculate positions in explicitly as [37].

Table 6: Experiment results of ablation studies.

TUTA Variants	WebSheet	DeEx	SAUS	CIUS	Average
TUTA-base, w/o TA	76.3%	70.5%	80.0%	92.8%	79.9%
TUTA-base, TA-8	80.0%	71.1%	80.5%	93.2%	81.2%
TUTA-base, TA-4	81.5%	73.8%	80.9%	95.6%	83.0%
TUTA-base, TA-2	84.5%	75.5%	84.6%	96.7%	85.3%
TUTA-explicit	86.5%	76.0%	89.7%	98.8 %	87.8 %
TUTA-implicit	**86.6%**	**76.6%**	**90.2%**	**99.0%**	**88.1%**
TUTA, w/o MLM	85.4%	**76.6%**	89.2%	**99.0%**	87.6%
TUTA, w/o CLC	83.0%	76.4%	88.7%	98.9%	86.8%
TUTA, w/o TCR	85.8%	**76.6%**	88.2%	**99.0%**	87.4%

Given the best TUTA-implicit, we dig deeper into the contributions of three objectives by separately removing each of them.

- TUTA, w/o MLM
- TUTA, w/o CLC
- TUTA, w/o TCR

Experiment results of ablation studies Table 6 shows the results of ablation studies. It is clear that smaller attention distances help our method to achieve better results. TUTA-base can only achieve 79.9% averaged macro-F1, lower than 82.0% of TAPAS. But when attention distance decreases to 2, TUTA-base, TA-2 has a significant improvement (5%) and outperforms TAPAS. Further, tree position embeddings improve accuracy as well. By augmenting with implicit tree embeddings, TUTA-implicit achieves an overall F1 of 88.1%. Although TUTA-implicit shows higher F1 than TUTA-explicit, the difference is relatively small. We thus conclude that tree positional embeddings, by encoding spatial and hierarchical information, can significantly improve the accuracy of TUTA. Rather, the embedding method for positions is not that critical. Moreover, we examine the effectiveness of three progressive objectives by removing each of them to keep the rest two of during pre-training. It is shown that removing the CLC objective has the biggest impact, with an accuracy drop of 1.3%. But the overall effect of removing MLM or TCR is not as significant. Note that for SAUS, removing TCR can also cause an obvious accuracy drop of 2%, and after our study, we find almost all tables in SAUS have informative titles and descriptions, which can serve as strong hints for metadata classification and global table semantics.

4.4 Case studies

To facilitate an intuitive understanding of the experiment results, we show two typical cases in the test set to help illustrate key concepts intuitively.

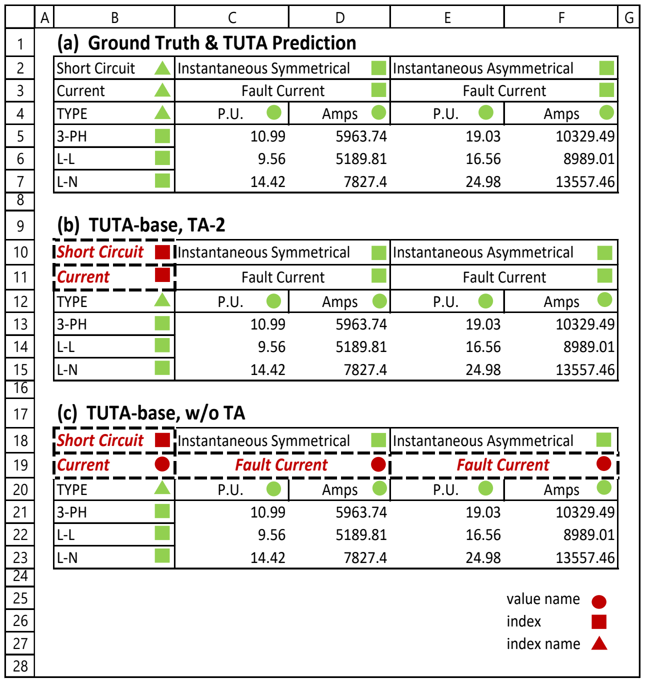
- **Tree Attention with different distances** Take the cells 'Fault Current' in Figure 8 (column C:D and E:F) as an example, without tree-attention, TUTA-base-w/o-TA tends to learn surrounding semantics with little discrimination, and mis-classifies cell 'Fault Current' to 'Value Name', as shown in Figure 8 (c). As we narrows its attention distance for each encoder from global to distance 2 as shown in Figure 8 (b), it successfully identifies 'Fault Current', we guess the underlying reason is that it captures greater similarity between 'Fault Current' and its hierarchical parents ('Instantaneous Symmetrical' and 'Instantaneous Asymmetrical'), rather than its hierarchical children, 'P.U.' and 'Amps' below.
- **Tree-based positional embeddings** Smaller attention distances help local cells to learn from spatial and structural related neighbors. To further help cells precisely identify spatial/ hierarchical relationships and avoid collective faults, we augment our attention mechanism with tree-based positional embeddings. Comparing TUTA and its no-position version in Figure 8 (a) and (b), respectively, we find B10 and B11 are misclassified to 'Index's (ground truth is 'Index Name') by no-position variant in (b), even when their neighbors in the right (C10, C12, D10, and D12) largely fall to the 'Index' type and are semantically described by B10 and B11. Beyond tree attention, tree-based positional embeddings help cells to find precise spatial and hierarchical relationships.
- **Interactive functions between headers and values** As for the case in Figure 9, TUTA categorize D28 and E28 to the 'Derived' type, that is, an aggregation from other cells of the 'Data' type. This result indicates an interactive phenomenon between header and value regions, such that a header often, targets and poses assumptions to certain data regions given the structural and semantic information. For example, the top header D26:D27 headlines the data D28. Its merged formatting, as well as its hierarchical level (as a direct child of top-tree root), readily sets a high aggregation priority to D28, who especially, also locates in the first data row. However, only by calculating possible aggregation results of data can we know that D28 is actually not 'Derived' from any other cells. Though it reaches beyond the scope of table semantic understanding, we do think it reasonable to augment numerical computations for tables in the future.

Although D28 is mis-classified by TUTA, E28 is successfully classified by TUTA — E28 is a **ground truth error caused by human labelers**. The ground truth for E28 should be 'Derived', but not 'Data' since E28 is indeed the summary of F28:G28 even if there is no formula to indicate it. We guess that TUTA makes the successfully classification for E28 based on the key word 'Total' in the top cell and the large number magnitude of E28.

5 RELATED WORK

Table pre-training methods Although TUTA is the first effort for pre-training table representations on various structured tables, there has been a range of prior work on relational table pre-training. Table-BERT linearized a table as a sentence so that a table can be directly processed by the pre-trained BERT model [7]. [16] adopted continuous bag-of-words and skip-gram to learn embeddings of table cells over 8 neighboring cells in local contexts and used the resulting embeddings as features in cell type classification, but it did not leverage structural information. TAPAS and TaBERT target question answering over relational tables via joint pre-training of tables and their text [19, 42]. Another work, TURL, attempted to pre-train embeddings on relational tables to enhance table knowledge matching and table augmentation [9]. But In TURL, each cell can only aggregate information from the located row and column due to the masked self-attention.

Neural networks dedicated for tables Since two-dimensional spatial information is crucial for table understanding tasks, lots of neural architectures have been proposed to capture spatial information. CNNs [5, 11, 12, 29, 41] are widely adopted to capture spatial information for spreadsheet tables, web tables, PDF tables,



(a) Ground Truth & TUTA Prediction

Short Circuit ▲	Instantaneous Symmetrical 🟩		Instantaneous Asymmetrical 🟩	
Current ▲	Fault Current 🟩		Fault Current 🟩	
TYPE ▲	P.U. 🟢	Amps 🟢	P.U. 🟢	Amps 🟢
3-PH 🟩	10.99	5963.74	19.03	10329.49
L-L 🟩	9.56	5189.81	16.56	8989.01
L-N 🟩	14.42	7827.4	24.98	13557.46

(b) TUTA-base, TA-2

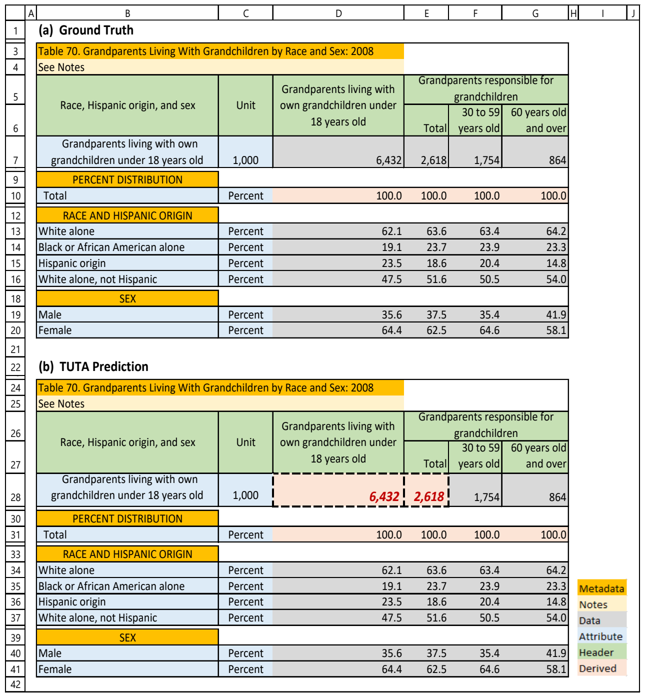
Short Circuit 🟥	Instantaneous Symmetrical 🟩		Instantaneous Asymmetrical 🟩	
Current 🟥	Fault Current 🟩		Fault Current 🟩	
TYPE ▲	P.U. 🟢	Amps 🟢	P.U. 🟢	Amps 🟢
3-PH 🟩	10.99	5963.74	19.03	10329.49
L-L 🟩	9.56	5189.81	16.56	8989.01
L-N 🟩	14.42	7827.4	24.98	13557.46

(c) TUTA-base, w/o TA

Short Circuit 🟥	Instantaneous Symmetrical 🟩		Instantaneous Asymmetrical 🟩	
Current 🔴	*Fault Current* 🔴		*Fault Current* 🔴	
TYPE ▲	P.U. 🟢	Amps 🟢	P.U. 🟢	Amps 🟢
3-PH 🟩	10.99	5963.74	19.03	10329.49
L-L 🟩	9.56	5189.81	16.56	8989.01
L-N 🟩	14.42	7827.4	24.98	13557.46

value name	🔴
index	🟥
index name	🔺

Figure 8: Case no.1, shows smaller attention distance and position embedding improve accuracy. Green icons mark correct predictions, while red ones mark mistakes.



(a) Ground Truth

Table 70. Grandparents Living With Grandchildren by Race and Sex: 2008					
See Notes					
Race, Hispanic origin, and sex	Unit	Grandparents living with own grandchildren under 18 years old	Grandparents responsible for grandchildren		
			Total	30 to 59 years old	60 years old and over
Grandparents living with own grandchildren under 18 years old	1,000	6,432	2,618	1,754	864
PERCENT DISTRIBUTION					
Total	Percent	100.0	100.0	100.0	100.0
RACE AND HISPANIC ORIGIN					
White alone	Percent	62.1	63.6	63.4	64.2
Black or African American alone	Percent	19.1	23.7	23.9	23.3
Hispanic origin	Percent	23.5	18.6	20.4	14.8
White alone, not Hispanic	Percent	47.5	51.6	50.5	54.0
SEX					
Male	Percent	35.6	37.5	35.4	41.9
Female	Percent	64.4	62.5	64.6	58.1

(b) TUTA Prediction

Table 70. Grandparents Living With Grandchildren by Race and Sex: 2008					
See Notes					
Race, Hispanic origin, and sex	Unit	Grandparents living with own grandchildren under 18 years old	Grandparents responsible for grandchildren		
			Total	30 to 59 years old	60 years old and over
Grandparents living with own grandchildren under 18 years old	1,000	*6,432*	*2,618*	1,754	864
PERCENT DISTRIBUTION					
Total	Percent	100.0	100.0	100.0	100.0
RACE AND HISPANIC ORIGIN					
White alone	Percent	62.1	63.6	63.4	64.2
Black or African American alone	Percent	19.1	23.7	23.9	23.3
Hispanic origin	Percent	23.5	18.6	20.4	14.8
White alone, not Hispanic	Percent	47.5	51.6	50.5	54.0
SEX					
Male	Percent	35.6	37.5	35.4	41.9
Female	Percent	64.4	62.5	64.6	58.1

Metadata
Notes
Data
Attribute
Header
Derived

Figure 9: Case no.2, shows an interactive pattern between header and data cells.

and scanned tables, and bidirectional RNNs and LSTMs are frequently adopted in web tables to capture the order of rows and columns [15, 16, 21, 28]. Later work has proposed a hybrid neural network by combining bidirectional RNNs and CNNs in the task of column type prediction [6]. However, Our proposed method is the first transformer-based method for table semantic structure understanding.

Structure-aware neural networks in NL domain In NL domain, a sentence can be represented via a dependency tree or a constituency tree structure. For this reason, a variant of LSTMs, named Tree-LSTM [4, 36], has been proposed to work on tree topology. Since dedicated models like the Tree-LSTM and RNNs are not as efficient and parallelizable as attention-based methods, [27] has devised tree-structured attention with bottom-up information accumulation and outperformed Tree-LSTMs on three text classification tasks. [39] has leveraged a learning-based constituent tree prior to guide the self-attention process. And tree-based positional encodings have also been proposed to help transformer better exploit tree-structured information [34]. In addition to tree-based methods, graph-based methods also have been widely adopted recently, for example, [20] used GATs to model the dependency graph in a NL sentence for sentiment classification, and [45] proposed graph-based evidence aggregating and reasoning model to capture relational and logical information among the evidence from plain text.

6 CONCLUSION AND DISCUSSION

In this paper, we propose a novel structure-aware pre-training framework, TUTA, for understanding tables with various semantic structures. TUTA is the first transformer-based method for table semantic structure understanding, which is enhanced with two core mechanisms to capture spatial and hierarchical information in tables, including tree attention and tree positional embeddings. Moreover, we devise three pre-training objectives to enable representation learning at token, cell and table levels. TUTA is pre-trained on large volume of unlabeled tables in an unsupervised manner and then fine-tuned on four well-annotated datasets for table semantic structure understanding. TUTA achieves state-of-the-art on all public datasets in the task of cell type classification, and shows a large margin of improvements over all baselines. Although we only validate TUTA through the task of table structure understanding, we believe TUTA is a general pre-training framework that can be applied to other table understanding tasks with minor modifications. So, in the future, we plan to demonstrate the effectiveness of TUTA on more tasks.

REFERENCES

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

[2] Zhe Chen and Michael Cafarella. Automatic web spreadsheet data extraction. In *Proceedings of the 3rd International Workshop on Semantic Search over the Web*, pages 1–8, 2013.

[3] Zhe Chen and Michael Cafarella. Integrating spreadsheet data via accurate and low-effort extraction. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1126–1135, 2014.

[4] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.
[5] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. Colnet: Embedding the semantics of web tables for column type prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 29–36, 2019.
[6] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. Learning semantic annotations for tabular data. *arXiv preprint arXiv:1906.00781*, 2019.
[7] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*, 2019.
[8] Eric Crestan and Patrick Pantel. Web-scale table census and classification. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 545–554, 2011.
[9] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. Turl: Table understanding through representation learning. *arXiv preprint arXiv:2006.14806*, 2020.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
[11] Haoyu Dong, Shijie Liu, Zhouyu Fu, Shi Han, and Dongmei Zhang. Semantic structure extraction for spreadsheet tables with a multi-task learning architecture. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019.
[12] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. Tablesense: Spreadsheet table detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 69–76, 2019.
[13] Wensheng Dou, Shi Han, Liang Xu, Dongmei Zhang, and Jun Wei. Expandable group identification in spreadsheets. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 498–508, 2018.
[14] Jing Fang, Prasenjit Mitra, Zhi Tang, and C Lee Giles. Table header detection and classification. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
[15] Besnik Fetahu, Avishek Anand, and Maria Koutraki. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference*, pages 2736–2742, 2019.
[16] Majid Ghasemi Gol, Jay Pujara, and Pedro Szekely. Tabular cell classification using pre-trained cell embeddings. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 230–239. IEEE, 2019.
[17] Julius Gonsior, Josephine Rehak, Maik Thiele, Elvis Koci, Michael Günther, and Wolfgang Lehner. Active learning for spreadsheet cell classification. In *EDBT/ICDT Workshops*, 2020.
[18] Tong Guo, Derong Shen, Tiezheng Nie, and Yue Kou. Web table column type detection using deep learning and probability graph model. In *International Conference on Web Information Systems and Applications*, pages 401–414. Springer, 2020.
[19] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*, 2020.
[20] Binxuan Huang and Kathleen M Carley. Syntax-aware aspect level sentiment classification with graph attention networks. *arXiv preprint arXiv:1909.02606*, 2019.
[21] Saqib Ali Khan, Syed Muhammad Daniyal Khalid, Muhammad Ali Shahzad, and Faisal Shafait. Table structure extraction with bi-directional gated recurrent unit networks. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1366–1371. IEEE, 2019.
[22] Elvis Koci, Maik Thiele, Josephine Rehak, Oscar Romero, and Wolfgang Lehner. Deco: A dataset of annotated spreadsheets for layout and table recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1280–1285. IEEE, 2019.
[23] Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, 2017.
[24] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.
[25] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76, 2016.
[26] Seung-Jin Lim and Yiu-Kai Ng. An automated approach for retrieving hierarchical data from html tables. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 466–474, 1999.
[27] Xuan-Phi Nguyen, Shafiq Joty, Steven CH Hoi, and Richard Socher. Tree-structured attention with hierarchical accumulation. *arXiv preprint arXiv:2002.08046*, 2020.
[28] Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
[29] Shubham Singh Paliwal, D Vishwanath, Rohit Rahul, Monika Sharma, and Lovekesh Vig. Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 128–133. IEEE, 2019.
[30] Viacheslav Paramonov, Alexey Shigarov, and Varvara Vetrova. Table header correction algorithm based on heuristics for improving spreadsheet data extraction. In *International Conference on Information and Software Technologies*, pages 147–158. Springer, 2020.
[31] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*, 2015.
[32] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.
[33] Dominique Ritze and Christian Bizer. Matching web tables to dbpedia-a feature utility study. *context*, 42(41):19–31, 2017.
[34] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems*, pages 12081–12091, 2019.
[35] Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 771–782, 2016.
[36] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.
[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
[39] Yau-Shian Wang, Hung-Yi Lee, and Yun-Nung Chen. Tree transformer: Integrating tree structures into self-attention. *arXiv preprint arXiv:1909.06639*, 2019.
[40] Xinxin Wang. Tabular abstraction, editing, and formatting. 2016.
[41] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5315–5324, 2017.
[42] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*, 2020.
[43] Richard Zanibbi, Dorothea Blostein, and James R Cordy. A survey of table recognition. *Document Analysis and Recognition*, 7(1):1–16, 2004.
[44] Chen Zhao and Yeye He. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The World Wide Web Conference*, pages 2413–2424, 2019.
[45] Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Gear: Graph-based evidence aggregating and reasoning for fact verification. *arXiv preprint arXiv:1908.01843*, 2019.
[46] Mengyu Zhou, Wang Tao, Ji Pengxin, Han Shi, and Zhang Dongmei. Table2analysis: Modeling and recommendation of common analysis patterns for multi-dimensional data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 320–328, 2020.

A DATASET CONSTRUCTION

Two kinds of table corpus are used for pre-training:

Web tables We collected 2.62 million web tables from WikiTable [5] and 50.82 million web tables from WDC WebTable Corpus [25]. We also kept their titles, captions, and surrounding NL contexts.

Spreadsheet tables We crawled about 13.5 million public spreadsheet files (.xls and .xlsx) from more than 1.75 million web sites. Then we utilized techniques of TableSense [12] to detect tables from sheets of each file. Totally we got about 115 million tables.

Pre-processing As the spreadsheet files are crawled from various web sites, the tables detected from them are very noisy. We cleaned the data and detected language for them to build a clean table corpus for pre-training. Firstly, we filtered out tables with extreme table size (number of rows/columns < 4, number of rows > 512 or number of columns > 128), tables with very deep hierarchical headers (number of top/left header rows/columns > 5) and tables without any headers. With these rules, we filtered 52.34% tables from the original dataset. Secondly, we de-duplicated the filtered data based on table content. After removing the duplicated tables, 23.49% tables are left. Thirdly, we used Microsoft Azure Text Analytics[6] to detect language for spreadsheet tables. Among all filtered and de-duplicated tables, about 31.76% tables are English, which were used for pre-training. Finally we got 4.49 million spreadsheet tables for TUTA.

Feature extraction We used ClosedXML[7] to parse spreadsheet files and extract features. For the two web table corpus, tables are serialized as JSON files, so we just load and parse the JSON files for feature extraction. We unified the featurization schema for web tables and spreadsheet tables as shown in Table7.

B TREE EXTRACTION

We adopt the method introduced by [11] for header detection. For those detected header regions, we develop a rule-based method to extract header hierarchies based on merged cells, indentation levels and formulas of table. By this method, we could get two hierarchical trees for each table, which are used to build the bi-dimensional coordinate tree for TUTA.

Merged cells Merged cells provide spatial alignment information between cells, with which we can build the hierarchical relationships between corresponding header cells. For example, the cells under a merged area belong to the child nodes of the merged area. By this way, we can build a hierarchical tree for top header cells based on merged cells. It is similar for building left header hierarchies based on the merged cells but in a different direction.

Indentation levels Indentation is commonly used for indicating hierarchical relationships of left headers both in web tables and spreadsheet tables. Generally, indentation refers to the visual indentation effect, which includes various operation methods. Users can use different amount of spaces and tabs to create indentation effect, which exists in both spreadsheet tables and web tables. Or they can write different levels of cell strings into different columns to create a visual effect of indentation. In spreadsheet tables, Excel provides an operation to set indentation level in cell format menu. We apply these expert knowledge as effective heuristics to extract indentation levels for left header cells. Based on the three before-mentioned operations, and further on the extracted indentation levels, we built hierarchical relationships in the left header for each cell and its subsequent cells with one more indentation levels.

Formulas Formula is an important feature in spreadsheet tables, which contains information about the calculation relationship between cells. Some formulas indicate the aggregation relationship between cells, such as SUM, AVERAGE, etc. What's more, if the formulas of all cells in a row share the same aggregation formula pattern, the left header node corresponding to this row should be treated as the parent node of other reference rows. It is similar to columns. Then we can get the hierarchical relationship based on formulas, which has higher priority than indentation levels in our method.

Table 7: Feature set for each cell.

Description	Feature value	Default value
Merged region		
The number of merged rows	positive integer	1
The number of merged columns	positive integer	1
Cell border		
If cell has a top border	0 or 1	0
If cell has a bottom border	0 or 1	0
If cell has a left border	0 or 1	0
If cell has a right border	0 or 1	0
Data type		
If cell string matches a date template	0 or 1	0
If formula exists in the cell	0 or 1	0
Cell format		
If the bold font is applied	0 or 1	0
If the the background color is white	0 or 1	1
If the the font color is white	0 or 1	1

[5]https://github.com/bfetahu/wiki_tables
[6]https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/
[7]https://github.com/ClosedXML/ClosedXML