Table Pre-training: A Survey on Model Architectures, Pre-training Objectives, and Downstream Tasks

Haoyu Dong[1], Zhoujun Cheng[2], Xinyi He[3], Mengyu Zhou[1], Anda Zhou[4], Fan Zhou[2], Ao Liu[5], Shi Han[1], Dongmei Zhang[1]
[1]Microsoft Research
[2]Shanghai Jiao Tong University
[3]Xi'an Jiaotong University
[4]University of Edinburgh
[5]Tokyo Institute of Technology
{hadong, mezho, shihan, dongmeiz}@microsoft.com, {blankcheng, zhoufan98}@sjtu.edu.cn, hxyhxy@stu.xjtu.edu.cn, a.d.zhou@sms.ed.ac.uk, zeitmond@gmail.com

Abstract

Following the success of pre-training paradigm in the natural language domain, a flurry of table pre-training frameworks have been proposed and have achieved new state-of-the-arts on various downstream tasks such as table question answering, table type recognition, column relation classification, table search, and formula prediction. Various model architectures have been explored to best leverage the characteristics of structured tables, especially specially-designed attention mechanisms. Moreover, to fully exploit the supervision signals in unlabeled tables, diverse pre-training objectives have been designed and evaluated, for example, denoising cell values, predicting numerical relationships, and learning a neural SQL executor. This survey aims to provide a review of model designs, pre-training objectives, and downstream tasks for table pre-training, and we further share our thoughts on existing challenges and future opportunities.

1 Introduction

Tables are widely used to organize and present data in various document types and database systems such as webpages, spreadsheets, PDFs, and MySQL, gaining increasing attention from the research community. Following the success of large-scale pre-training in natural language (NL), a flurry of research works have been proposed to leverage unlabeled tables for self-supervised pre-training and achieve promising results in table type classification [Wang *et al.*, 2021c], cell type classification [Gol *et al.*, 2019; Wang *et al.*, 2020], table question answering (QA) [Herzig *et al.*, 2020; Yin *et al.*, 2020], table search [Wang *et al.*, 2021b], entity linking [Deng *et al.*, 2020], column type identification [Chen *et al.*, 2019; Guo *et al.*, 2020], table augmentation [Deng *et al.*, 2020; Iida *et al.*, 2021], formula prediction [Cheng *et al.*, 2021], etc. On the one hand, similar to NL that has already proved the success of large-scale pre-training, tables have dense semantics stored in textual headers, captions, and notes. On the other hand, different from NL, tables have distinct information (intuitive formats, well-organised numerical values, formulas, etc.) and various structures (relational tables, entity tables, matrix tables, forms, etc.), and thus require special model architectures and pre-training objectives to achieve optimal results.

To best leverage table characteristics while maintaining capabilities to understand text within/out of tables, various

year	GDP			real growth (%)	GDP per capita (GDPpc) *based on mid-year population*			Reference index	
	GDP *in millions*				GDPpc			exchange rate *1 foreign currency to CNY*	
	CNY	USD	PPP (Int'l$.)		CNY	USD	PPP (Int'l$.)	USD 1	Int'l$. 1 (PPP)
2016	2,566,910	386,449	733,214	6.8	118,198	17,795	33,762	6.6423	3.5009
2015	2,368,570	380,285	667,297	6.9	109,602	17,597	30,878	6.2284	3.5495
2014	2,194,410	357,233	618,074	7.4	102,870	16,746	28,974	6.1428	3.5504

(a) A hierarchical matrix web table in Wikipedia

Personal information	
Born	August 23, 1978 Philadelphia, Pennsylvania
Died	January 26, 2020 (aged 41) Calabasas, California
Nationality	American
Listed height	6 ft 6 in (1.98 m)[a]
Listed weight	212 lb (96 kg)

(b) An entity table in Wikipedia

System	MNLI-(m/mm) 392k
Pre-OpenAI SOTA	80.6/80.1
BiLSTM+ELMo+Attn	76.4/76.1
OpenAI GPT	82.1/81.4
$\text{BERT}_{\text{BASE}}$	84.6/83.4
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**

(c) A relational PDF table in [Devlin et al., 2018]

	Type of Injury						
Type of Weapon/ Force Involved	*None*	*Possible Internal Injury*	*Severe Laceration*	*Apparent Minor Injury*	*Other Major Injury*	*Loss of Teeth*	*Unconsciousness*
Firearms	**60,469**	**1,426**	**1,414**	**9,714**	**5,910**	**83**	=SUM(J5:J7)
Firearm	12,652	429	321	2,144	2,260	15	40
Handgun	46,257	960	1,051	7,038	3,517	66	140
Other Firearm	1,560	37	42	532	133	2	4
All Other	**70,409**	**3,228**	**7,630**	**61,043**	**4,735**	**505**	**1624**
Blunt Object	10,049	1,237	4,685	14,982	1,802	256	691

(d) A matrix spreadsheet table

Figure 1: Examples of real-world web, PDF, and spreadsheet tables.

Tabular Language Models (TaLMs) are proposed for table pre-training. For example, TaBERT [Yin *et al.*, 2020] encoded tables and text by concatenating a row-wise transformer with column-wise vertical attention layers, pre-trained with Masked Language Models (MLM) and Masked Column Prediction (MCP), and achieved SOTA results on benchmarks of table QA. TaPas [Herzig *et al.*, 2020] pioneeringly proposed an end-to-end table-text joint reasoning framework using transformers without explicitly generating logical forms for table QA, and TaPas also employed MLM for pre-training. TURL [Deng *et al.*, 2020] was the first work to learn entity representations from relational tables to enhance table knowledge matching and table augmentation, and it restricted each cell to aggregating information from the located row and column via masked attention. TUTA [Wang *et al.*, 2020] then extended the success of table pre-training to generally structured tables using tree-based attention and tree-based positional encoding based on a novel unified bi-tree structure. TUTA achieved new SOTA results on five table structure understanding datasets. Far different from previous pre-training objectives, TaPEx [Liu *et al.*, 2021] explored to learn a neural SQL executor and demonstrated surprising effectiveness on table-text joint reasoning. Recently, UnifiedSKG [Xie *et al.*, 2022] explored to directly fine-tune T5 on 21 datasets across 6 tasks and achieved promising and even SOTA results.

We believe that the structured table provides a distinct perspective to explore frontier neural architectures and inspire new research directions. Since the table often interacts with programming languages such as SQLs and spreadsheet formulas, it additionally spawns cross-domain applications such as semantic parsing [Yu *et al.*, 2018], logic-to-text [Chen *et al.*, 2020], and formula prediction [Chen *et al.*, 2021]. In this paper, we first introduce preliminaries of table types, table structures, cell information, and table corpora in Section 2. Then we give a comprehensive review of table modeling architectures, table pre-training objectives, and downstream tasks in Sections 3,4,5. At last, we share our vision on table pre-training in Section 6.

2 Preliminaries

Tables can be roughly categorized into three forms: well-structured tables, semi-structured tables, and unstructured tables. Database tables are **well structured** with an ordinarily-defined relational schema so that precise support execution of formal languages such as SQL and R. In contrast, **semi-structured** tables are usually human-crafted with markup languages or end-user tools such as HTML code, Latex code, spreadsheets, and Word documents. They have flexible structures but lack meta-information to record them and thus challenge precise and automatic information retrieval. Image tables are even **unstructured** because they only record raw RGB information, e.g., scanned tables from books, screenshots of web tables, and even handwritten drafts. They need to be digitized before any higher-level information retrieval.

In this paper, we mainly focus on semi- and well-structured tables. We **neglect image tables** because they have distinct visual challenges and are desirable to be discussed separately.

2.1 Table Structure

Tables are flexible with various structures, including relational tables, entity tables, matrix tables, layout tables, forms, etc. They also have orientations (horizontal/vertical) and hierarchies (flat/hierarchical). As shown in Figure 1, the structure of a flat relational table is definite and straightforward in a database-like form, wherein each row is a record, each column is a field, and there is no hierarchy. An entity table simply records an entity and its attributes. A matrix table has both horizontal and vertical orientations. In fact, there are various categorization methodologies on the table structure, which are summarized in detail by [Zhang and Balog, 2020].

2.2 Cell Information

Typically, a cell is the intersection of one row and one column in a table. And multiple cells can be merged into a larger cell that occupies multiple rows and columns. Cells are basic units to record text, numerical values, formats, formulas, etc.

1) Text. Text is a critical component in the table to record meta information in headers, notes, and captions, as well as data region cells. Texts in tables are basically in NL but often have short lengths and concise meanings to meet the space restriction in documents.

2) Numerical values. A large proportion of cells store numerical values. Unlike text, numerical values could have arithmetical relationships, such as sum and proportion, and statistical characteristics, such as distribution and trend.

3) Visual formats. Tables have various intuitive formats to present the table structure or content, such as border, alignment, background color, and font [Dong *et al.*, 2020].

4) Formulas. In several popular end-user tools such as Excel and Google Sheet, spreadsheet formulas are used to store the logical and numerical relationships between cells.

5) Other elements such as hyperlinks, images, and icons can also be inserted into a cell. A table can even be nested in Word documents by inserting sub-tables into individual cells.

2.3 Existing Large Table Corpus

Web Tables Large corpora include WDC Web Table Corpus [1] (233M tables), Dresden Web Tables Corpus [Eberius *et al.*, 2015] (174M tables), WebTables [Cafarella *et al.*, 2008] (154M tables), and WikiTables (1.6M tables). More details are summarized by [Zhang and Balog, 2020].

Spreadsheet Tables FUSE [Barik *et al.*, 2015] included 249,376 web-crawled spreadsheets. [Chen and Cafarella, 2013] obtained 410,554 Excel files from 51,252 Internet domains TUTA [Wang *et al.*, 2021c] collected 13.5 million spreadsheet files from 1.75 million web sites.

CSV Tables GitTables [Hulsebos *et al.*, 2021] collected 1M+ tables from CSV files in GitHub repositories. Tables in GitTables are similar with typical database tables.

Other Kinds of Tables TableArXiv [2] contains 341,573 tables extracted from scientific publications on arxiv.org.

[1] http://webdatacommons.org/framework/

[2] http://boston.lti.cs.cmu.edu/eager/table-arxiv/

3 Model

Since two-dimensional information is crucial for understanding table structures, many neural architectures have been proposed to jointly capture structure and semantic information. Table2Vec [Zhang *et al.*, 2019] adopted skip-gram neural network models to train word embeddings with row/column population. CNNs [Dong *et al.*, 2019; Chen *et al.*, 2019] were adapted to capture spatial information in two-dimensional tables. Bidirectional RNNs and LSTMs have been widely used to capture the order of rows/columns [Nishida *et al.*, 2017; Gol *et al.*, 2019; Fetahu *et al.*, 2019; Kardas *et al.*, 2020]. Later works explored graph neural networks for table understanding and question answering [Zayats *et al.*, 2021; Zhang *et al.*, 2020; Koci *et al.*, 2018; Du *et al.*, 2021]. While CNNs, RNNs, and GNNs have been widely used for table modeling, few adopted them for large-scale table pre-training (except a CNN-based TCN [Wang *et al.*, 2021a]). The main reason is that **most of them train (or directly consume) token/word embeddings separately from the CNN/RNN/GNN model**, thus restricting models' capabilities in understanding cell texts together with table structures.

Recently, a flurry of works explored to use transformer-based language models (LMs) for table pre-training, and we call them Tabular Language Models (TaLMs), e.g., TaPas [Herzig *et al.*, 2020], TaBERT [Yin *et al.*, 2020], TURL [Deng *et al.*, 2020], TUTA [Wang *et al.*, 2021c], Tabnet [Arik and Pfister, 2021], VIME [Yoon *et al.*, 2020], KGPT [Wang *et al.*, 2021b], RPT [Tang *et al.*, 2021], StruG [Deng *et al.*, 2021], TabTransformer [Huang *et al.*, 2020], GraPPa [Yu *et al.*, 2020], GAP [Shi *et al.*, 2021], BRIDGE [Lin *et al.*, 2020], TABBIE [Iida *et al.*, 2021], TaPEx [Liu *et al.*, 2021], ForTaP [Cheng *et al.*, 2021], MATE [Eisenschlos *et al.*, 2021], STTP [Xing and Wan, 2021], GTR [Wang *et al.*, 2021b], TableFormer [Yang *et al.*, 2022], and FLAP [Authors, 2021]. The advantage of using transformers is that they can pre-train semantic and structural representations jointly and inherit the text understanding power of existing NL pre-trained models such as BERT, BART [Lewis *et al.*, 2020], and T5 [Raffel *et al.*, 2020]. Works like UnifiedSKG [Xie *et al.*, 2022] and TableGPT [Gong *et al.*, 2020], while without table-specific pre-training, directly fine-tuned LMs on table tasks and achieved promising and even SOTA results, demonstrating that **pre-training is transferable from text to tables**, e.g., in linguistic and world knowledge aspects. Considering that using transformer-based TaLMs is a common choice for table pre-training, **in the following subsections, we dig deeper into TaLMs** on sequence serialization, input embedding, the encoder and decoder architecture, the attention mechanism, and model efficiency.

3.1 Tabular Sequence Serialization

TaLMs require a sequence of tokens to be the model input like LMs. A straightforward yet common method is to linearize raw tables row by row. Most works such as TaPas, MATE, TableFormer, TUTA, and TURL, performed in this way. In TaPEx, the table is also linearized row by row but it additionally inserts several special tokens to indicate table components such as [HEAD] and [ROW] representing the region of table headers and rows respectively. TABBIE linearized tables by rows and columns separately for two transformers.y TableGPT distinctly adapted a template-based table serialization way on relatively simple tables. Experiments conducted by UnifiedSKG showed that putting external text (like questions) ahead of tables could help T5 to generalize better on tabular tasks. [Li *et al.*, 2021] directly encoded markup languages like NL. Some works linearized a specific part of a table, e.g., TaBERT [Yin *et al.*, 2020] linearized most relevant rows to the input utterance, and StruG and GraPPa only took headers as input without data cells.

3.2 Input Featurization and Embedding

Cell Text Encoding Most table pre-training methods tokenized cell text using WordPiece and learned token embeddings [Devlin *et al.*, 2018], such as TaBERT, TaPas, MATE, StruG, TableFormer, TUTA, ForTaP, and TABBIE. Some works also used the BPE tokenization following Roberta [Liu *et al.*, 2019] or BART [Lewis *et al.*, 2020] , e.g., GraPPa and TaPEx. TURL was initialized by TinyBERT [Jiao *et al.*, 2020] and additionally learned embeddings based on an entity vocabulary. Rather than directly using the vocabulary parsed from NL corpora, TUTA constructed a table-specific vocabulary using WordPiece based on large table corpora and merged it with BERT's vocabulary.

Positional Encoding Following NL pre-trained models, most TaLMs embedded 1D sequential positions in the serialized tabular sequence, e.g., TaPas, MATE, StruG, GraPPa, and TaPEx. Some other works divided the whole sequence into multiple pieces and counted positions in each piece separately: TUTA treated each cell as an independent piece and locally encoded positional information of tokens inside a cell; TURL regarded the table caption and the header as two separate pieces, then it used two local positional encodings.

Tables also have two-dimensional row/column and hierarchical information. Works such as TaPas, MATE, and TUTA, learned column/row embeddings based on column/row IDs and showed increased performance. However, considering hierarchical structures, as Figure 1 (c) shows, column/row encoding results in limited representation capability. TUTA further devised explicit/implicit tree-based positional embeddings to jointly encode the spatial and hierarchical positions and showed significant effectiveness on generally structured tables. However, it has not proved helpful for downstream tasks that only involve flat and relational tables.

Numerical Encoding A large number of numerical values are distributed in tables and challenge BERT-based models to learn optimal representations since these methods simply tokenized and encoded numerical values in the same way as NL text. It corrupts the original recording structure of numbers into fragments and introduces difficulties in number representation [Thawani *et al.*, 2021]. Explorations on learning better number representations have surged in the NLP field recently [Thawani *et al.*, 2021], while few works attempted in tabular data. TaPas and MATE devised a unique rank embedding for column-wise number comparison, bringing improvements on answering comparatives or superlative questions. FLAP added extra feature encoding to indicate whether

Encoder	TaPas, TaBERT, TURL, TUTA, StruG, GraPPa, MATE, DoT, ForTaP, TableFormer, TABBIE, BRIDGE, etc.
Encoder+Decoder	KGPT, RPT, TaPEx, GAP, STTP, FLAP, UnifiedSKG, etc.
Decoder	TableGPT, etc.

Table 1: Encoder and decode architectures of TaLMs.

the text summary mentioned the value. TUTA distinguished numerical values from pure text via embedding over four discrete numerical features: magnitude, precision, the first digit, and the last digit. It is highly desirable to explore more numerical embedding methods in future works, e.g., in arithmetical and statistical perspectives.

Format Encoding Formats contain valuable hints about table structures and data highlights, but only a few TaLMs considered them. E.g., TUTA learned format embeddings together with the transformer backbone to distinguish whether a cell has merge, border, font bold, font color, and fill color.

3.3 Encoder and Decoder Architecture

Existing table pre-training models mainly inherit the architectures from language models such as BERT, GPT-2 [Radford *et al.*, 2019], and BART. Depending on the focused downstream tasks, these models adopted different components, i.e., encoders or decoders, as summarized in Table 1. Most of them adopted the **encoder** part of transformers similar to BERT, including TURL, StruG, TaPas, GraPPa, MATE, TUTA, ForTap, and TableFormer. Typically, a single encoder is applied on the sequential inputs constructed from tables and associated texts, if any, to learn the contextual representations of the inputs. Additional modules such as classification layers are applied to the encoder for downstream tasks. Some works even employed more than one encoder to capture the structural information of tables. TaBERT stacked column-wise self-attention layers on the row-wise encoder. TABBIE employed two encoders to encode the table row-wise and column-wise separately, then aggregated the representations obtained from both encoders. DoT [Krichene *et al.*, 2021] also used two encoders, one acting as a pruning model to select the most relevant tokens from the input and the other one used for performing specific tasks. Another branch of TaLMs used an **encoder-decoder** architecture to better enable sequence generation tasks such as table-to-text. KGPT used two alternative encoders, a GNN-based graph encoder and a transformer-based sequential encoder; each can be combined with a transformer decoder. RPT also adopted the encoder-decoder model, similar to a BERT model combined with a GPT model. TaPEx and UnifiedSKG implemented the encoder-decoder (text-to-text) model based on BART and T5, respectively, for downstream tasks such as text-to-SQL parsing, table-based QA, and table fact verification. STTP [Xing and Wan, 2021] focused on table-to-text generation by fine-tuning BART on table-structure-aware self-supervised tasks. Instead of pre-training, TableGPT directly fine-tunes the GPT-2 **decoder** to take advantage of its contextual knowledge learned from linguistic corpora.

Dense	Self attention	TaPas, TaPEx, etc.
	Self attention with attention bias	TableFormer
Sparse	Serial row/column attention layers	TaBERT
	Parallel row/column attention layers	TABBIE
	Parallel row/column attention heads	MATE
	Joint row/column attention	TURL
	Joint bi-tree-based attention	TUTA
	Joint layout-based graph attention	GTR
	Joint knowledge-triple-based graph attention	KGPT

Table 2: Attention mechanisms of TaLMs.

3.4 Structure-based Attention

The attention mechanism is essential for TaLMs to compute contextual representations [Vaswani *et al.*, 2017]. Besides many of them that directly adopt self-attention, e.g., TaPas, StruG, GraPPa, and TaPEx, a series of structure-based attention mechanisms have been proposed to better leverage the tabular structure, as summarized in Table 2.

Self-attention may introduce a lot of irrelevant and even noisy contexts and have lots of unnecessary computations [Wang *et al.*, 2021c], while table structures can be leveraged towards precise and efficient attention. In Table 2, "dense" means all inputs in the table are visible in self-attention, while "sparse" means only parts of the table are visible. TaBERT learned tabular representations serially by first producing row-wise encoding with a transformer and then column representation with vertical attention layers using row-wise encodings as inputs. TURL designed a restricted attention mechanism in which each token/entity attends to tokens in the same row/column. TUTA proposed joint bi-tree-based attention, which took in both spatial and hierarchical information from tables. More specifically, for a structured table, TUTA defines cell coordinates and cell-to-cell distance from a bi-dimensional tree generated from the table. The bi-tree-based attention is restricted to attending tokens within the tree-based distance threshold. Cell embedding in TABBIE is an average of its row and column embeddings, where row/column embeddings are separately calculated by row/column transformers. MATE used different types of attention heads, i.e., row attention head and column attention head, which are restricted to attending tokens in the same row and column (query tokens can attend to all tokens). GTR, which mainly focused on table retrieval, transformed a table into a tabular graph and used joint layout-based graph attention similar to the graph transformer to capture structural information. The graph transformer in KGPT enforced to use the structure (knowledge triple) as a hard constraint of attention, e.g., in the first encoder layer, each node was restricted to attend to tokens in its located knowledge triple. Tabnet [Arik and Pfister, 2021] applied a sequential attention mechanism to generate an interpretable feature selection mask during each decision step. Rather than hard attention masking, TableFormer proposed to use soft attention biases when computing attention scores between two structural components. Structure-based attention not only improves model performance but potentially benefits model efficiency, and we will introduce it in Section 3.5.

3.5 Model Efficiency

Most TaLMs are inefficient at dealing with long sequences due to the quadratic complexity of self-attention [Vaswani *et al.*, 2017; Tay *et al.*, 2020]. Unfortunately, tables from webs and spreadsheets usually contain dozens of rows or columns, posing a significant challenge to the memory and computational efficiency [Eisenschlos *et al.*, 2021]. A naive way [Herzig *et al.*, 2020; Liu *et al.*, 2021] is truncating the input by a maximum sequence length, but it may lose critical information. So there emerge many other strategies:

Input selection One intuitive way is to filter the important part of the table before feeding them to the model. TaBERT extracted *content snapshot*, the most relevant table rows to the NL sentence(s) regarding n-gram overlap ratio. Similarly, TaPas with intermediate pre-training [Eisenschlos *et al.*, 2020] ranked columns by Jaccard coefficient between the NL and each column tokens. The model was twice as fast to train as TaPas while achieving similar performance. TUTA randomly sampled out 50% text cells and 90% numeric cells during pre-training since spreadsheets are usually large while limited semantics are introduced by similar data cells. Instead of using heuristics to prune inputs, DoT presented a model with two transformers: a pruning transformer selected top-K tokens, and a task-specific transformer took them as inputs. The architecture was two times faster to train, while the memory bottleneck depended on the size of the pruning transformer. A small or medium-size pruning transformer was usually enough to achieve comparative performance with large-size TaPas, but falls behind on more challenging datasets like WTQ [Pasupat and Liang, 2015]. While input selection is effective for tasks with table-text joint input like table QA and fact verification, it may fail for tasks that (1) all cells are required to be predicted, e.g., cell type classification; (2) table is the only input, e.g., table-to-text and formula prediction.

Input splitting An alternative way is to split a large table into multiple chunks and feed chunks separately into the model. TUTA split the table into chunks containing the same header row(s) and non-overlapped data rows in its downstream task, cell type classification [Dong *et al.*, 2019]. For formula prediction, SpreadsheetCoder [Chen *et al.*, 2021] (not a pre-training method but a typical case) split the target table by chunks with $N = 3$ adjacent table rows/columns. The chunks were encoded by a BERT [Devlin *et al.*, 2018] encoder and then aggregated by convolutional layers. Splitting the input table allows encoding all table cells, while it costs more time due to multiple inferences of the encoder and is thus not used by most table pre-training methods.

Sparse attention MATE leveraged the sparse attention of table encoding by an efficient implementation to reduce memory cost. In MATE, row and column attentions were designed for different attention heads, implemented following ETC [Ainslie *et al.*, 2020] by dividing the input into a global part G attending to and from everything and a local part attending to G and tokens within a radius R. The model scaled linearly concerning memory ($8,000$ tokens at most) and speed with this efficient attention implementation. Note that though sparse attention based on table structure is widely adopted by TaLMs, they mainly aimed to improve performance (e.g., accuracy) instead of efficiency. Replacing these sparse attentions with efficient implementations can largely mitigate the memory issue: ForTaP used the TVM [Chen *et al.*, 2018] framework to compile a CUDA kernel to implement sparse tree-attention in TUTA, and the maximum input sequence length was up to 8,000 tokens.

4 Pre-training Objectives

The pre-training objectives of TaLMs fall into two categories: Denoising autoencoder and task-specific objectives. Following the idea of Masked Language Modeling (MLM) [Devlin *et al.*, 2018], many objectives adopt self-supervised labels for a TaLM to remove synthetic noises as autoencoder. Meanwhile, a variety of other pre-training objectives take inspirations from specific downstream tasks to design new supervisions. The former apply self-supervised learning and denoise the table itself, and the latter build supervision according to external supervision signals or specific tasks.

4.1 Denoising Autoencoder Objectives

For denoising autoencoder objectives, TaLMs take partially corrupted inputs and recover the original ones. Most TaLMs applied token-level MLM on tabular sequences in the same way as NL sequences. More advanced denoising objectives considered the table structure such as cell and column.

Token-level Most pre-training models used token MLM [Devlin *et al.*, 2018] by masking the input tokens at random and then predicting those masked tokens. TaPas, MATE, and TableFormer followed the standard MLM procedure to randomly mask 15% of tokens. Larger ratio was taken by TURL (20%) and TUTA (30%) to make recovering more challenging. Certain restrictions could also be applied on what tokens to mask. E.g., MLM used in TaBERT only masked tokens in external NL context, and MLM used in TURL and GraPPa masked both NL context and table headers.

Cell-level (1) Masking and recovery. A cell could correspond to one or multiple token(s) in the tabular sequence. Slightly different masking strategies were designed. TUTA used a whole-cell masking strategy to capture relationships of cells. Cell Value Recovery (CVR) in TaBERT applied the span-based prediction objective to deal with multiple value tokens. In TCN, each token represented a cell, and 10% cells were masked for recovery from the set of cell values. (2) Cell cloze. TUTA sampled cell positions based on the bi-tree structure as candidate choices. At each position, TUTA was encouraged to retrieve its corresponding cell string. (3) Classifying cell corruption. TABBIE corrupted cells in two ways, frequency-based and intra-table cell swapping.

Column-level Masked Column Prediction (MCP) was introduced by TaBERT to recover the names and data types of masked columns. GAP proposed to recover columns names using column values or the input utterance. Both of them assumed that tables were vertically-oriented and relational.

4.2 Task-specific Objectives

To achieve SOTA performances on downstream task(s), denoising objectives might not be enough. Then task-specific

Task	Description	Model
Table question answering	Given a table and a question in NL, output an answer. The question-answer pair should be supported by the table.	TaPas, TaBERT, StruG, GraPPa, TaPEx, GAP, MATE, ForTaP, BRIDGE, TableFormer, UnifiedSKG
Table fact verification	Verify whether a textual hypothesis holds based on the given table.	MATE, TableFormer, UnifiedSKG
Table-to-text	Generate textual description(s) from the given table.	KGPT, TableGPT, FLAP, STTP, UnifiedSKG
Table type classification	Classify the table into different structural types.	TUTA
Cell type classification	Identify cell structural types in the table.	TUTA, ForTaP
Column type &relation classification	Associate a column in a table with the KB type of entities it contains. Associate a pair of columns in a table with the KB relation that holds between each pair of entities in a given row of the columns.	TURL, TABBIE, TCN
Table augmentation	Expand the table with additional data.	TURL, TABBIE
Formula prediction	Predict a spreadsheet formula for the target cell in the table.	ForTaP
Entity linking	Find phrases of text, called mentions, in cells and associate each with its referent entity.	TURL
Table search	Retrieve semantically relevant tables based on NL queries.	GTR
Data preparation	Include six subtasks: data discovery, data validation, data filtering, data structuring, data enrichment, and data cleaning.	RPT
Machine learning applications	Various tasks using categorical or continuous features stored in tabular format, e.g., the competitions held by Kaggle and KDD Cup.	Tabnet, VIME, TabTransformer

Table 3: Downstream task evaluation for table pre-training. In this table, we try to cluster some similar tasks, e.g.,column type&relation classification and a series of tasks of data preparation. We also merge sub-tasks like logic-to-text, into main tasks like table-to-text.

objectives were proposed and proved to be highly effective.

Objectives by Downstream Tasks

In this section, representative objectives are grouped by tasks. Definitions of the downstream tasks can be found in Table 3.

Table QA and text-to-SQL There are a variety of works on table QA and text-to-SQL, and we list some representative ones here. A critical technical component of text-to-SQL is the alignment between text and tables. GraPPa designed an objective: given an NL sentence and table headers, predicting whether a column appears in the SQL query and what operation is triggered. StruG used several grounding tasks of text-to-SQL as objectives, including selecting columns mentioned in sentences, finding cell values from sentences, and mapping column-value. TaPEx proposed to learn a novel neural SQL executor given a table and a synthesized executable SQL query. PReasM [Yoran *et al.*, 2021] synthesized at scale question-paragraph pairs that required different reasoning skills to enhance numerical reasoning abilities. GAP learned to compose complex SQL on tables.

Table fact verification Entailment check is highly related with QA. [Eisenschlos *et al.*, 2020] used an intermediate pre-training objective of synthetic table fact checking targeting both real-world table fact verification and table QA. TaPEx, as described above, also showed benefits for table fact check.

Entity linking TURL proposed a Masked Entity Recovery objective by masking a certain percentage of entity cells and then recovering the linked entity based on surrounding entity cells and table metadata. It helped the model capture the factual knowledge embedded in the table content and the associations between table metadata and table content.

Table type classification and table search TUTA provided each table with text segments and was pre-trained to retrieve the corresponding tables using text segments.

Numerical reasoning ForTaP proposed to predict numerical reference and numerical calculation relationships, and aimed to benefit all related tasks involving table numerical reasoning, e.g., table QA and formula prediction.

Objectives by Data Sources

The above objectives are possible with human-created and machine-synthesized data on different sources of tables.

Human-created Human-created data usually show higher quality than web-crawled ones which might require careful prepossessing for their size, diversity and noises. It is common to manually add extra labels for existing dataset. E.g., ToTTo, a well-labeled dataset for table-to-text with NL descriptions and corresponding web tables, was used by StruG for pre-training. Also, human-created labels can be collected in smart ways. E.g., ForTaP extracted existing formulas from a large web-crawled spreadsheet corpus and extracted numerical reference and calculation relationships from them for pre-training. And large fine-grained labeled datasets were also used for pre-training, e.g., ToTTo, a well-labeled dataset for table-to-text with NL descriptions and corresponding web tables, was used by StruG for pre-training.

Machine-synthesized Synthesized data are more targeted and controllable, but require careful designs to ensure meaningfulness and diversity. GraPPa proposed an SCFG (synchronous context-free grammar) and applied it to synthesize sentence-SQL pairs over tables. [Eisenschlos *et al.*, 2020] created counterfactual and synthetic statements for existing Wikipedia tables: For the counterfactual ones, it got tables and sentences from Wikipedia as positive examples and created minimally differing refuted examples; For the synthetic ones, it built table-dependent statements by synthesizing them from the pre-defined probabilistic CFG (context-free grammar). TaPEx randomly selected tables from the

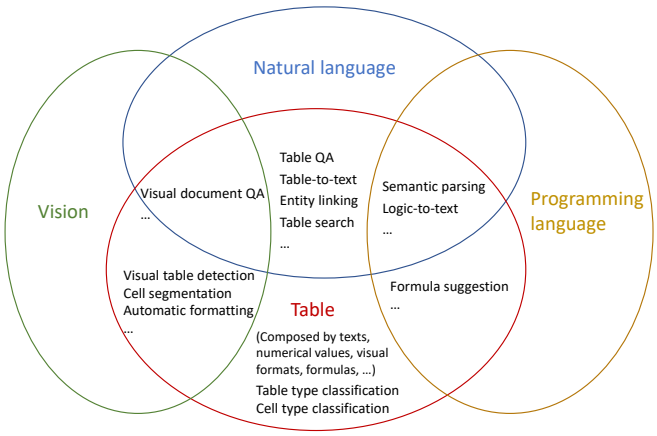

Figure 2: Downstream task categorization by domains.

training set of WIKITQ [Pasupat and Liang, 2015] and instantiated SQL templates to synthesize table-SQL pairs.

5 Downstream Tasks

As shown in Figure 2, tasks of table understanding often have intersections with domains like NL, programming language, and computer vision, and thus prefer different capabilities of table modeling. For example, table question answering is a prevalent cross-domain task that requires models to understand tables and NL questions jointly, and to enable robust reasoning over tables, semantic parsing becomes a widely-studied task of parsing NL questions to programming languages such as SQLs, logical forms, or python code. We think that classifying tasks by domains presents a fresh perspective to future works on multi-modal modeling, but it is not an absolutely strict or static categorization, e.g., table QA can involve programming languages via semantic parsing [Yin *et al.*, 2020], while it can also use end-to-end prediction [Herzig *et al.*, 2020] without explicitly using a programming language. Tasks can also be categorized by their scenarios in addition to domains, e.g., data preparation represents a range of tasks for data preparation. The machine learning application covers various benchmarks or competitions where features and labels are stored in a tabular form.

In Table 3, due to space limitation, we only list downstream tasks that have been evaluated by existing TaLMs. Nonetheless, it is highly desirable for future work to explore more tasks such as table format generation [Dong *et al.*, 2020], data analysis [Zhou *et al.*, 2020], and table error detection [Huang and He, 2018; Zhang *et al.*, 2021].

6 Conclusion and Discussion

This paper presents a review of model architectures, objectives, and downstream tasks of table pre-training. Although the "pre-training and fine-tuning" paradigm has demonstrated its success on many table tasks, there still are key challenges (opportunities) for future research:

Combining Diverse Bi-dimensional Cell Features Effectively Tables are arranged in two-dimension, including not only text but also quantities, visual formats, hyperlinks, and even spreadsheet formulas, it is non-trivial to learn high-level representations from such diverse and raw information. It particularly challenges existing language models that directly consume a flat sequence. Should tables be linearized like NL text? How to maintain and combine the structural, textual, formatting, and numerical information in a most effective way is still an open challenge.

Universal Framework for Downstream Tasks Almost all TaLMs only focus on one or two downstream tasks (as shown in Table 3) so that they have sufficient flexibility on model designs (e.g., Section 4.2) to achieve the best performance. But it is desirable to unify the advantages of existing methods and support various downstream tasks simultaneously like what BERT and GPT did in the NL domain. However, the diversity of table downstream tasks presents a significant challenge, e.g., entity linking and formula prediction and entity linking may need far different feature sets, sampling mechanisms, and model capabilities. It's a highly demanding direction to explore a universal framework by integrating the advantages of different TaLMs.

Visualizing, Probing, and Comparing What Aspects TaLMs Learned from Table Pre-training Attention layers in transformers are often challenged for being opaque. In the NL domain, to uncover linguistic and world knowledge learned by pre-trained LMs, there exist attentive works on studying the outputs of pre-trained LMs on carefully designed input sentences [Linzen *et al.*, 2016], examining the internal vector representations of pre-trained LMs through methods such as probing tasks [Adi *et al.*, 2016; Belinkov *et al.*, 2017], and visualizing attention maps of a pre-trained LMs [Bahdanau *et al.*, 2014; Vig, 2019; Rogers *et al.*, 2020]. Later works even studied and probed attention layers head-by-head and found substantial syntactic information captured in BERT [Clark *et al.*, 2019]. Some TaLMs claimed their abilities on table-text joint reasoning (TaPas, TaBERT, TaPEx, ...), table structure understanding (TUTA, TableFormer, ...), and numerical reasoning (ForTaP, ...), but there still lack attentive works on visualizing, probing, and comparing these intangible pre-trained TaLMs.

Consistency and Discrepancy Between LMs and TaLMs Recently, UnifiedSKG explored to directly fine-tune T5 on 21 datasets across 6 tasks. On the one hand, the frontier LM (T5) could easily achieve promising or even SOTA results on table tasks, demonstrating a strong relationship between text and tables, e.g., in linguistic and world knowledge aspects. On the other hand, without table-specific model design and pre-training, it still fell behind TaLMs with table pre-training on WikiTQ (-8.2%) and SQA (-12.1%); even larger margins may exist on untested tasks, such as formula prediction. It shows the necessity of table-specific pre-training, e.g., in structural and reasoning aspects. (1) Is the initialization from advanced LMs necessary for table pre-training? (2) When zero-shot LMs are large enough (considering that GPT-3 has not been fully exploited on table tasks yet), do existing table pre-training strategies still have performance gains? (3) During table pre-training, how to best inherit the knowledge from LMs while exploit table-specific capabilities?

References

[Adi *et al.*, 2016] Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*, 2016.

[Ainslie *et al.*, 2020] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, 2020.

[Arik and Pfister, 2021] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6679–6687, 2021.

[Authors, 2021] Anonymous Authors. Flap: Table-to-text generation with feature indication and numerical reasoning pretraining. *ACL Rolling Review Nov.*, 2021.

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[Barik *et al.*, 2015] Titus Barik, Kevin Lubick, Justin Smith, John Slankas, and Emerson Murphy-Hill. Fuse: a reproducible, extendable, internet-scale corpus of spreadsheets. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 486–489. IEEE, 2015.

[Belinkov *et al.*, 2017] Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*, 2017.

[Cafarella *et al.*, 2008] Michael J Cafarella, Alon Y Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. Uncovering the relational web. In *WebDB*, 2008.

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

[Chen *et al.*, 2018] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, 2018.

[Chen *et al.*, 2019] 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*, pages 29–36, 2019.

[Chen *et al.*, 2020] Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. Logic2text: High-fidelity natural language generation from logical forms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020.

[Chen *et al.*, 2021] Xinyun Chen, Petros Maniatis, Rishabh Singh, Charles Sutton, Hanjun Dai, Max Lin, and Denny Zhou. Spreadsheetcoder: Formula prediction from semi-structured context. In *International Conference on Machine Learning*. PMLR, 2021.

[Cheng *et al.*, 2021] Zhoujun Cheng, Haoyu Dong, Fan Cheng, Ran Jia, Pengfei Wu, Shi Han, and Dongmei Zhang. Fortap: Using formulae for numerical-reasoning-aware table pretraining. *preprint arXiv:2109.07323*, 2021.

[Clark *et al.*, 2019] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

[Deng *et al.*, 2020] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. Turl: table understanding through representation learning. *Proceedings of the VLDB Endowment*, 2020.

[Deng *et al.*, 2021] Xiang Deng, Ahmed Hassan, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-grounded pretraining for text-to-sql. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

[Devlin *et al.*, 2018] 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.

[Dong *et al.*, 2019] 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.

[Dong *et al.*, 2020] Haoyu Dong Dong, Jinyu Wang, Zhouyu Fu, Shi Han, and Dongmei Zhang. Neural formatting for spreadsheet tables. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 305–314, 2020.

[Du *et al.*, 2021] Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei Zhang. Tabularnet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 322–331, 2021.

[Eberius *et al.*, 2015] Julian Eberius, Katrin Braunschweig, and Others. Building the dresden web table corpus: A classification approach. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, pages 41–50. IEEE, 2015.

[Eisenschlos *et al.*, 2020] Julian Eisenschlos, Syrine Krichene, and Thomas Mueller. Understanding tables with intermediate pretraining. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020.

[Eisenschlos *et al.*, 2021] Julian Eisenschlos, Maharshi Gor, Thomas Mueller, and William Cohen. Mate: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

[Fetahu *et al.*, 2019] 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.

[Gol *et al.*, 2019] 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.

[Gong *et al.*, 2020] Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1978–1988, 2020.

[Guo *et al.*, 2020] 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.

[Herzig *et al.*, 2020] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, 2020.

[Huang and He, 2018] Zhipeng Huang and Yeye He. Auto-detect: Data-driven error detection in tables. In *Proceedings of the 2018 International Conference on Management of Data*, 2018.

[Huang *et al.*, 2020] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint:2012.06678*, 2020.

[Hulsebos *et al.*, 2021] Madelon Hulsebos, Çağatay Demiralp, and Paul Groth. Gittables: A large-scale corpus of relational tables. *arXiv preprint arXiv:2106.07258*, 2021.

[Iida *et al.*, 2021] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. Tabbie: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456, 2021.

[Jiao *et al.*, 2020] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tiny-BERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP*, 2020.

[Kardas *et al.*, 2020] Marcin Kardas, Piotr Czapla, Pontus Stenetorp, Sebastian Ruder, Sebastian Riedel, Ross Taylor, and Robert Stojnic. Axcell: Automatic extraction of results from machine learning papers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[Koci *et al.*, 2018] Elvis Koci, Maik Thiele, Wolfgang Lehner, and Oscar Romero. Table recognition in spreadsheets via a graph representation. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 139–144. IEEE, 2018.

[Krichene *et al.*, 2021] Syrine Krichene, Thomas Müller, and Julian Martin Eisenschlos. Dot: An efficient double transformer for nlp tasks with tables. *arXiv preprint arXiv:2106.00479*, 2021.

[Lewis *et al.*, 2020] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[Li *et al.*, 2021] Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. Markuplm: Pre-training of text and markup language for visually-rich document understanding. *arXiv preprint arXiv:2110.08518*, 2021.

[Lin *et al.*, 2020] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, 2020.

[Linzen *et al.*, 2016] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[Liu *et al.*, 2021] Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jian-guang Lou. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*, 2021.

[Nishida *et al.*, 2017] 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.

[Pasupat and Liang, 2015] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140), 2020.

[Rogers *et al.*, 2020] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[Shi *et al.*, 2021] Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cicero Nogueira dos Santos, and Bing Xiang. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[Tang *et al.*, 2021] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. Rpt: relational pre-trained transformer is almost all you need towards democratizing data preparation. *Proceedings of the VLDB Endowment*, 14(8), 2021.

[Tay *et al.*, 2020] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

[Thawani *et al.*, 2021] Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. Representing numbers in nlp: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–656, 2021.

[Vaswani *et al.*, 2017] 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.

[Vig, 2019] Jesse Vig. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*, 2019.

[Wang *et al.*, 2020] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. Tuta: Tree-based transformers for generally structured table pre-training. *arXiv preprint arXiv:2010.12537*, 2020.

[Wang *et al.*, 2021a] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. Tcn: Table convolutional network for web table interpretation. In *Proceedings of the Web Conference 2021*, pages 4020–4032, 2021.

[Wang *et al.*, 2021b] Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro Szekely. Retrieving complex tables with multi-granular graph representation learning. *arXiv preprint arXiv:2105.01736*, 2021.

[Wang *et al.*, 2021c] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790, 2021.

[Xie *et al.*, 2022] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*, 2022.

[Xing and Wan, 2021] Xinyu Xing and Xiaojun Wan. Structure-aware pre-training for table-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2273–2278, 2021.

[Yang *et al.*, 2022] Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. Tableformer: Robust transformer modeling for table-text encoding. *arXiv preprint arXiv:2203.00274*, 2022.

[Yin *et al.*, 2020] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[Yoon *et al.*, 2020] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 2020.

[Yoran *et al.*, 2021] Ori Yoran, Alon Talmor, and Jonathan Berant. Turning tables: Generating examples from semi-structured tables for endowing language models with reasoning skills. *arXiv preprint arXiv:2107.07261*, 2021.

[Yu *et al.*, 2018] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, 2018.

[Yu *et al.*, 2020] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Caiming Xiong, et al. Grappa: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*, 2020.

[Zayats *et al.*, 2021] Vicky Zayats, Kristina Toutanova, and Mari Ostendorf. Representations for question answering from documents with tables and text. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2895–2906, 2021.

[Zhang and Balog, 2020] Shuo Zhang and Krisztian Balog. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2020.

[Zhang *et al.*, 2019] Li Zhang, Shuo Zhang, and Krisztian Balog. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.

[Zhang *et al.*, 2020] Xingyao Zhang, Linjun Shou, Jian Pei, Ming Gong, Lijie Wen, and Daxin Jiang. A graph representation of semi-structured data for web question answering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 51–61, 2020.

[Zhang *et al.*, 2021] Yakun Zhang, Xiao Lv, Haoyu Dong, Wensheng Dou, Shi Han, Dongmei Zhang, Jun Wei, and Dan Ye. Semantic table structure identification in spreadsheets. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 283–295, 2021.

[Zhou *et al.*, 2020] Mengyu Zhou, Wang Tao, Ji Pengxin, Han Shi, and Zhang Dongmei. Table2analysis: Modeling and recommendation of common analysis patterns for multi-dimensional data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):320–328, Apr. 2020.