FORTAP: Using Formulas for Numerical-Reasoning-Aware Table Pretraining

Zhoujun Cheng[1*], **Haoyu Dong**[2*†], **Ran Jia**[2],
Pengfei Wu[3], **Shi Han**[2], **Fan Cheng**[1†], **Dongmei Zhang**[2]
[1]MoE Key Laboratory of Artificial Intelligence, AI Institute,
Shanghai Jiao Tong University, Shanghai 200240, China
[2]Microsoft Research Asia, China [3]Fudan University, China
{blankcheng,chengfan}@sjtu.edu.cn, 17307130207@fudan.edu.cn
{hadong,raji,shihan,dongmeiz}@microsoft.com

Abstract

Tables store rich numerical data, but numerical reasoning over tables is still a challenge. In this paper, we find that the spreadsheet formula, a commonly used language to perform computations on numerical values in spreadsheets, is valuable supervision for numerical reasoning in tables. Considering large amounts of spreadsheets available on the web, we propose FORTAP , the first exploration to leverage spreadsheet formulas for table pretraining. Two novel self-supervised pretraining objectives are derived from formulas, numerical reference prediction (NRP) and numerical calculation prediction (NCP). While our proposed objectives are generic for encoders, to better capture spreadsheet table layouts and structures, we build FORTAP upon TUTA, the first transformer-based method for spreadsheet&web table pretraining with tree attention. FORTAP outperforms state-of-the-art methods by large margins on three representative datasets of formula prediction, question answering, and cell type classification, showing the great potential of leveraging formulas for table pretraining. The code will be released at https://github.com/microsoft/TUTA_table_understanding.

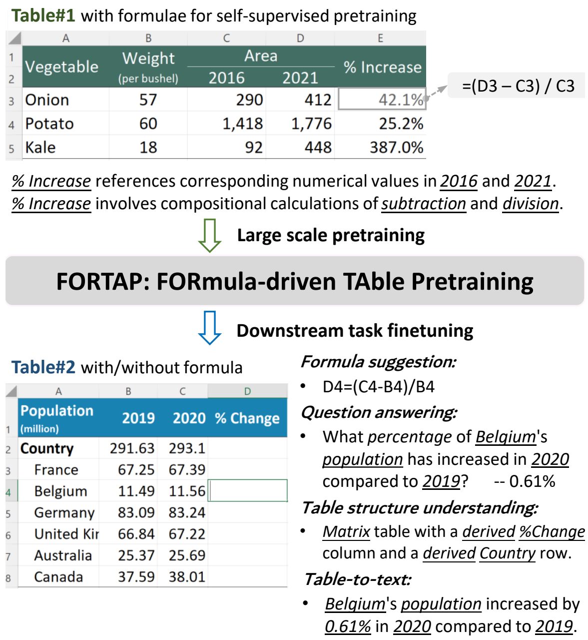

Figure 1: It's desirable to learn numerical reasoning via formula pretraining and generalize it to various tasks.

1 Introduction

Tables store rich numerical data, so a wide range of tasks require numerical reasoning over (semi-)structured tabular context, such as question answering over tables (Chen *et al.*, 2021b; Zhu *et al.*, 2021; Cheng *et al.*, 2021), table-to-text (Suadaa *et al.*, 2021; Moosavi *et al.*, 2021; Cheng *et al.*, 2021), spreadsheet formula prediction (Chen *et al.*, 2021a), and table structure understanding (Koci *et al.*, 2019). Take Table#2 in Figure 1 as an example, both suggesting the formula `(C4-B4)/B4` for cell D4 and answering "0.61%" to the question require numerical reasoning capabilities of **(1)** understanding the contextual meaning of individual numerical cells, e.g., "11.49" at `B4` and "11.56" at `C4` are "population"s of "Belgium" in "2019" and "2020"; **(2)** inferring calculational relationships of numerical cells, e.g., percentage change from "11.49" to "11.56". As Figure 1 shows, same capabilities also benefit table structure recognition and table-to-text. So it's a fundamental need to empower table modeling with stronger numerical reasoning capabilities.

However, it is challenging to endow a tabular model with robust numerical reasoning capabilities. First, understanding a local numerical cell needs dimension inference (Chambers and Erwig, 2008), unit inference (Shbita *et al.*, 2019), and index inference (Dong *et al.*, 2019a), e.g., "population" (dimension), "million" (unit), "2020" (index), and "Belgium" (index) jointly describe "11.56" in Figure 1. It is non-trivial concerning the great flexibility of table semantic structures (Wang *et al.*,

*The first two authors contribute equally.
†Corresponding authors.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1150 - 1166

2021b). Second, calculational relationships among two or more numerical cells are various and often compositional, e.g., "F1 Score = 2 × (Recall × Precision) / (Recall + Precision)" in machine learning papers and "Profit Margin = Net Income / Sales" in financial reports. To make matters more challenging, **human labeling** for numerical reasoning in relevant tasks (Chen *et al.*, 2020; Suadaa *et al.*, 2021; Koci *et al.*, 2019) is labor-intensive and error-prone, largely restricting the generalization ability of large models that are rather data-hungry.

Recently, table pretraining on large amount of unlabeled tables shows promising results on table understanding and reasoning. Self-supervised objectives are derived from tables and text such as Masked Language Models (MLM) (Herzig *et al.*, 2020), masked column prediction (Yin *et al.*, 2020), masked entity recovery (Deng *et al.*, 2020b), cell cloze and corrupt detection (Wang *et al.*, 2021b; Tang *et al.*, 2020; Iida *et al.*, 2021), table-text matching and alignment (Wang *et al.*, 2021a,b; Deng *et al.*, 2020a). However, numerical and calculational relationships of cells lack sufficient attention. Then (Yoran *et al.*, 2021) and (Liu *et al.*, 2021; Yu *et al.*, 2020) synthesize questions and SQL queries, respectively, as training corpus for reasoning purpose, but SQL is only applicable to database-like relational tables, and importantly, it's challenging to ensure synthesized questions and SQLs be realistic, meaningful, and diverse.

Gladly, tens of millions of real spreadsheet formulas are publicly available on the web and can be valuable for numerical reasoning in tables. The spreadsheet formula is an expressive yet simple language consisting of operators (e.g., `+`, `/`, `%`), functions (e.g., `SUM`, `MAX`, `COUNT`), referenced cells (e.g., `B4`), and constant values (e.g., 100) (Aivaloglou *et al.*, 2015). Since writing the formula **does not** require formal programming education, it's widely used by non-programmers such as business professionals or other kinds of domain specialists whose jobs involve computational tasks. So spreadsheet formulas cover real numerical calculations in a great variety of domains.

To this end, we propose FORmula-driven TAble Pretraining (FORTAP) for numerical reasoning. One should master two basic concepts to use the formula language: cells as variables and operators/functions as relationships between variables. So we explicitly decompose information in formulas into *numerical reference* and *numerical calculation* and devise two complementary tasks. Given a table as well as a formula cell in it, we mask the formula and then (1) the model classifies whether "header A references header B" (we consider that "header A references header B" if the formula cell belonging to header A references a numerical cell belonging to header B, as illustrated in Figure 2); (2) the model predicts the operator/function of two or more referenced numerical cells. Furthermore, to better encode and represent formulas, we also apply MLM to the token sequence of formulas.

Considering the flexibility of table structures in spreadsheets, we base FORTAP on TUTA (Wang *et al.*, 2021b), the first transformer-based method for spreadsheet tables with carefully-designed textual, numerical, positional, and formatting embedding layers. Importantly, its tree-based position encoding and attention are highly effective in representing generally structured tables. TUTA is pretrained with MLM, cell cloze, and table-text matching.

Experiment results on three tasks demonstrate that the significance of leveraging formulas for table pretraining. For formula prediction, FORTAP achieves 55.8% top-1 accuracy, significantly surpassing TUTA (48.5%), TaPEx (43.2%), and SpreadsheetCoder (40.4%) on Enron. For table question answering, TUTA achieves comparable accuracy with the best system on HiTab. After pretraining with formulas, FORTAP delivers a huge improvement of $+6.3\%$ as over previous SOTA, comparable to TaPEx. For cell type classification, on dataset DeEx, FORTAP largely improves TUTA by $+6.6\%$ on `derived` type and $+3.2\%$ on overall Macro-F1.

2 Preliminaries

2.1 TUTA as Encoder

TUTA (Wang *et al.*, 2021b) is the first pretraining architecture for spreadsheet tables. It is effective in capturing table semantic structures, achieving SOTA results on cell type and table type classification. As mentioned in Section 1, understanding table semantic structures is critical to numerical reasoning, so we choose TUTA to be the encoder of FORTAP . Since our pretraining tasks are generic for encoders of tables, future works can also explore other encoders such as (Herzig *et al.*, 2020).

Header Recognition. Headers usually provide short yet informative descriptions of table contents in Natural Language (NL), so TUTA leverages the detected header regions and hierarchies, as presented in Section 2.2. (Chen *et al.*, 2021a) also shows that using headers (even without considering hierarchies) greatly helps formula prediction. FORTAP follows to place detected headers in inputs.

Architecture. TUTA bases on BERT (Devlin *et al.*, 2019) with several enhancements: (1) a *positional encoding layer* based on a unified *bi-dimensional coordinate tree* to describe both the spatial and hierarchical information of cells; (2) a *number encoding layer* to encode magnitude, precision, the first digit, and the last digit; (3) a *tree-based attention* mechanism that enables local cells to aggregate their structurally neighbouring contexts within a *tree-based distance* threshold.

Model Input/Output. The input consists of a table T and optional NL texts C. By traversing the cell matrix of a table from left to right and from top to bottom, the input is linearized to "`[CLS]`, C_0, ..., C_{K-1}, `[SEP]`, $T_{(0,0)}$, `[SEP]`, $T_{(0,1)}$, ..., `[SEP]`, $T_{(M-1,N-1)}$", where K is the token length of NL texts, and M and N are the numbers of rows and columns of the table, respectively. Note that $T_{(i,j)}$ refers to the token sequence of the cell string in the $(i+1)^{th}$ row and $(j+1)^{th}$ column, and each token has token, number, position, and format input embeddings. The output of the encoder contains token-level, cell-level, and table-level embeddings. FORTAP follows these input/output settings except when inputting formula token sequence.

2.2 Pretraining Corpus

Spreadsheet Source and Preprocessing. We use the same spreadsheet table corpus as TUTA: (1) 13.5 million public spreadsheet files are crawled from 1.75 million websites; (2) table ranges and headers are detected using TableSense (Dong *et al.*, 2019b,a); (3) header hierarchies are extracted with effective heuristics; (4) extreme size tables are filtered out; (5) duplicated tables are discarded. In the end, 4.5 million spreadsheet tables are left.

Formula Preprocessing. Spreadsheet Formula is a widely-used end-user language for table organization and calculation. A formula consists of four types of formula tokens: operator (e.g., `+`, `/`, `%`), functions (e.g., `SUM`), referenced cells (e.g., `B4`) and constant values (e.g., 100), which we denote as `OP`, `FUNC`, `CELL` and `CONST` in the rest part of the paper. We use XLParser (Aivaloglou *et al.*, 2015), a highly-compatible formula parser with compact grammar, to analyze formula. In this way, we derive the AST of each formula (an example AST in Figure 2) and the type of each formula token. Since we focus on single table setting, we discard the cross-table, cross-sheet, and cross-file formulas. Formulas with *Array* or *User-Defined-Function* are also discarded. The absolute reference sign "$" is deleted from formula strings, without changing their meanings. We only keep the first five occurrences of formulas in the same row/column because some spreadsheets contain hundreds of duplicated or dragged formulas in one row/column, which are inefficient for training. Formulas are linearized as formula token sequences in prefix representation of AST following SpreadsheetCoder (Chen *et al.*, 2021a). Finally, 10.8 million formulas are derived.

3 Pretraining Tasks

As mentioned in Section 1, empowering table modeling with stronger numerical reasoning capabilities is a fundamental need. Spreadsheet formulas naturally contain information of numerical references (`CELL`) and calculations (`OP`/`FUNC`), motivating us to devise effective tasks to leverage them for numerical-reasoning-aware pretraining.

Based on information parsed from the formula expression, we carefully devise two complementary objectives, Numerical Reference Prediction (NRP) and Numerical Calculation Prediction (NCP), to exploit the reasoning process behind referencing local cells (as operands) and applying calculations (on operands), respectively. Meanwhile, to get better representations of the spreadsheet formula, which could be further used in downstream applications like formula error detection (Cheung *et al.*, 2016), we extend MLM (Devlin *et al.*, 2019) from NL contexts to formulas. Figure 2 gives an illustration of these tasks.

Numerical Reference Predication (NRP) We consider "header A references header B" in a table if: in a formula, the formula cell (cell with formula) belonging to header A references a cell belonging to header B. Take the table in Figure 2 as an example, the header *"%Increase"* references headers *"2016"* and *"2021"* since `E3` in column *"%Increase"* references `C3` and `D3` in columns *"2016"* and *"2021"*. We let the model learn header reference relationship since a cell belonging to a referenced header is more likely to be involved in the calculation. It is important but usually unknown as a priori, especially when tables are from diverse or unfamiliar domains. Note that we use header cells instead of data cells in this task since headers provide high-

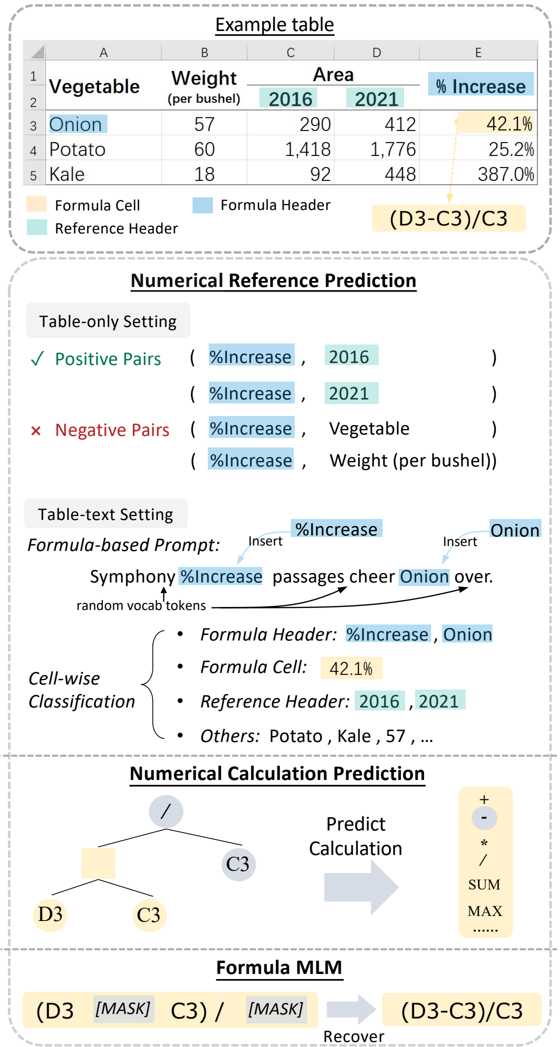

Figure 2: An illustration of formula pretraining tasks.

level descriptions of the data (Chen *et al.*, 2021a) and thus header reference relationships have more generic semantics across tables.

Given extracted header regions and hierarchies in corpus preprocessing, we first formulate NRP as a binary classification task over header pairs: given a formula cell t_f and its referenced cells $\{t_p^{(i)}\}$, we first find their non-shared headers h_f (for t_f) and $\{h_p^{(i)}\}$ (for $\{t_p^{(i)}\}$), then we group them as positive pairs $\{(h_f, h_p^{(i)})\}$. Usually a formula cell shares a header with referenced cells in the same row/column (e.g., in Figure 2, *"Onion"* is the shared header for E3, C3, D3). As it does not reflect header reference relationships, we exclude the shared header in this task. The negative pairs $\{(h_f, h_n^{(i)})\}$ are sampled among those unreferenced headers on the same direction (either on top or left headers) of h_f . Number of negative samples is at most 3:1 to positive ones to balance samples. The binary classification probability of the i^{th} pair $p^{(i)} = f(\mathbf{h}_f, \mathbf{h}_{p/n}^{(i)})$, where $\mathbf{h}$ is the header cell embedding derived by the encoder and $f(\cdot)$ is a two-layer binary classification module.

To inject table-text joint reasoning skills into FORTAP , which TUTA does not excel at, we further extend NRP task to table-text setting. Given a table with a formula cell, we first construct a formula-based prompt as context by picking 1 to 10 tokens randomly from the vocabulary as a noisy sentence and then inserting the row and column header of formula cell into it at random positions. Next, we jointly input the formula-based prompt and the table, and the task is to classify (1) formula header cell, (2) formula cell, (3) reference header cell, (4) other cells from the table. To precisely classify these cells, model needs to first align formula header cells in table with prompt (alignment skill), then infer the intersection cell of formula header cells as formula cell (spatial reasoning). Finally, it has to identify referenced cells (numerical reasoning) by the formula headers.

The NRP loss $\mathcal{L}_{nr}$ is calculated as the sum of binary cross entropy loss and multi-class cross entropy loss under table-only and table-text setting.

Numerical Calculation Prediction (NCP) Given data cells as operands, a model then needs to find out which operators/functions should be applied. For example, in Figure 2, subtraction and division are applied on C3 and D3 in the formula. We hope the model can speculate the target operator/function based on the semantics, numeracy, and positions of given operands (data cells). Thus, we design the task to predict the operator/function for a group of data cells with their contextual cell embeddings produced by the encoder.

We formulate it as a multi-class classification task: given a formula and its AST parsed in prerpocessing, we select the operators/functions $\{o^{(i)}\}$ satisfying that all direct children nodes $\{d^{(j)}\}^{(i)}$ on the formula AST of $o^{(i)}$ are in CELL type with integer or float data. The probability of predicting the operator/function of these data cells is $p^{(i)} = f(\texttt{POOL}(\{\mathbf{d}^{(j)}\}^{(i)}))$, where $\mathbf{d}$ is the output cell embedding by the encoder, $f(\cdot)$ is a two-layer classification module, and POOL is a mean-pooling layer. Note that we only include the operator/function o whose all direct children nodes are in CELL type in this task, because otherwise some descendant data cells will first be calculated via other operators/functions and thus have indirect connections with o (e.g., in Figure 2, "/" is not a target operator since its left child is an operator “$-$”). We include 17 common calculation operators/functions (see Appendix A) covered in spreadsheet formulas in this task. The NCP objective $\mathcal{L}_{nc}$ is the multi-class cross entropy loss.

Formula MLM To encode formulas, we expand 41 tokens in the vocabulary for all four formula token types, covering 99.1% formulas in corpus. Added tokens are listed in Appendix A. Note that a special case is the `CELL` type, like `D4`, because it references another cell. Since referenced cells can be anywhere in a large table, it is infeasible to explicitly insert all cell positions into the vocabulary. Thus, for `CELL` type token in formula, we use a `[RANGE]` tag as input token and copy all cell-level embeddings (position, format, numeric, ...) from the referenced cell to this `CELL` type token.

We then apply MLM to formula tokens. Masking and recovering operators/functions is straightforward. When masking or recovering a referenced cell in a formula, we need to avoid label leakage from embeddings of the referenced cell. Thus, to mask a referenced cell, besides using the `[MASK]` token embedding, the number embedding is set to default to mask the number, and the position and format embeddings are set to the same as the formula cell. To recover a masked referenced cell t_r, the cell $t^{(i)}$ in input sequence with the highest probability $p^{(i)} = \texttt{Softmax}(f(\mathbf{t}_r, \mathbf{t}^{(i)}))$ is selected as the predicted cell, where $\mathbf{t}$ is output cell embedding of the encoder and $f(\cdot)$ is a two-layer classification module. The objective $\mathcal{L}_{fmlm}$ is calculated as the sum of cross entropy loss over operator/function recovery and referenced cell recovery.

Finally, the total pretraining objective is

$$\mathcal{L} = \mathcal{L}_{nr} + \mathcal{L}_{nc} + \mathcal{L}_{fmlm} \tag{1}$$

4 Experiments

In this section, we describe the pretraining details and validate the effectiveness of FORTAP on three downstream tasks: formula prediction, question answering, and cell type classification. The statistics of datasets we use are listed in Table 1.

4.1 Pretrain Implementation

We initialize FORTAP with parameters of the pretrained TUTA. The input is linearized following TUTA by concatenating the text (the prompt built in NRP pretraining task) and the flattened table traversed in row order. Due to memory limit, we only place (1) header cells, (2) data cells on the same row/column of the formula cell, into the input sequence and skip the other cells. Our input pattern is reasonable as a tradeoff between performance and memory since we find that more than 89% formulas only reference cells on the same row/column. To match different downstream tasks, for the cell with formula, we input its formula token sequence (e.g. `(C4-B4)/B4`) with 40% probability, formula tag `[FORMULA]` with 30% (the number embedding is set to default) and cell literal value with 30% (e.g. number 42.1). In experiments, we find it is more effective in Formula MLM to mask either all operators/functions or all referenced cells, so we implement it this way. We first pretrain $400K$ steps on sequence length 256 with batch size 32, and $250K$ steps on sequence length 512 with batch size 8. The whole pretraining phase takes about 4 days on 4 Tesla V100 GPUs.

Dataset	Enron	HiTab	DeEx
# samples (train/dev/test)	$125k$ (formulas)	$10.6k$ (questions)	$711k$ (cells)
% hierarchical tables	51.0%	98.1%	43.7%
Avg. rows per table	25.7	17.1	220.2
Avg. columns per table	12.4	8.2	12.7
Avg. formula sketch length	4.13	-	
Avg. op/func per formula	1.62	-	

Table 1: Statistics of downstream datasets.

4.2 Formula Prediction

Formula prediction (Chen *et al.*, 2021a) facilitates spreadsheet end-users by recommending formulas since writing formulas could be time-consuming and error-prone. Given a table and a target cell in table, the task is to predict a formula for the target cell. Formula prediction requires complex in-table numerical reasoning capabilities to predict both referenced cells and involved calculations.

Datasets. Enron (Hermans and Murphy-Hill) is a massive database of public Excel Spreadsheet, containing over 17K spreadsheets with rich table structures and formula types. We exclude Enron from our pretraining corpus to prevent data leakage. Tables and formulas are preprocessed in the same way as the pretraining corpus. We divide Enron by sheet and the final dataset contains $100.3K$/$12.3K$/$12.9K$ table-formula pairs for train/dev/test. The formula cell in table is regarded as the target cell and the formula is seen as the ground truth in formula prediction task. We follow the evaluation metrics in Spreadsheet-Coder (Chen *et al.*, 2021a): (1) Formula Accuracy, (2) Sketch Accuracy, (3) Range Accuracy measuring the percentage of correctly predicted formulas, formula sketches (formula using placeholder [RANGE] as referenced cells), and formula ranges (only the referenced cells of formula).

Previous to our work, SpreadsheetCoder evaluates formula prediction on collected Google Sheets and Enron. However, we do not directly use its datasets for three reasons: (1) The Google Sheet corpus is not released, and for Enron, SpreadsheetCoder only adopts formulas referencing cells within a limited rectangular neighborhood region (21×20) of the formula cell, while we argue in real tables the referenced cells can be easily beyond this region. (2) A large proportion of table headers are not properly detected (mentioned in its paper), while we adopt ranges and headers detected by TableSense (Dong *et al.*, 2019b) and extract table header hierarchies. (3) Despite the inconsistencies above, we try to backtrack the original file to align with SpreadsheetCoder and apply our preprocessing. However, the document IDs of tables in SpreadsheetCoder are mostly empty. Thus, we build our dataset based on Enron and evaluate SpreadsheetCoder on it for a fair comparison.

Baselines. We adopt SpreadsheetCoder (Chen *et al.*, 2021a), TaPEx (Liu *et al.*, 2021), and TUTA as our baselines. SpreadsheetCoder is a BERT-based model for formula prediction, incorporating headers and contextual information of neighbouring cells of the target cell. TaPEx is a BART-based (Lewis *et al.*) table pretraining model, which implicitly learns a SQL executor.

Fine-tune. FORTAP consumes all header cells in the table and data cells lying on the same row/column of the target cell just like the manner in pretraining, with a max sequence length, 512. The [FORMULA] tag is placed at the target cell position in input, whose number embedding is set to default. A two-stage LSTM formula decoder (Dong and Lapata, 2018; Chen *et al.*, 2021a) accepts the formula cell embedding as input, and generates the formula by first generating formula sketches and then selecting referenced cells. All models in experiments are fine-tuned $800K$ steps on Enron. The beam size is 5 for generating formula. Since SpreadsheetCoder only published part of its code, we re-implement it in PyTorch (Paszke *et al.*, 2019) based on its paper. Appendix B presents details about SpreadsheetCoder. TaPEx is built on BART model and thus naturally supports generation task.

(%)	**Formula**	**Sketch**	**Range**
	20% Train Set		
TUTA	29.8	50.5	59.0
FORTAP	**40.0**	**57.6**	**69.5**
	100% Train Set		
SpreadsheetCoder	40.4	59.6	67.7
TaPEx	43.2	-	-
TUTA	48.5	65.3	75.3
FORTAP	**55.8**	**70.8**	**78.8**

Table 2: Formula prediction accuracy on Enron.

We follow the TaPEx table linearization strategy, assign the formula position in the source, and modify the target vocabulary as SpreadsheetCoder (Chen *et al.*, 2021a) to support generating referenced cells. We use the TaPEx-base model. It is fine-tuned for $30K$ steps (converge at about $25K$) and evaluated on the checkpoint with the best dev performance.

Results. Table 2 summarizes the results of formula prediction on the test set. As shown, FORTAP delivers a big improvement over SpreadsheetCoder by $+15.4\%$ and TaPEx by $+12.6\%$ on formula accuracy. We deduce that TaPEx falls behind TUTA and FORTAP because (1) the learnt executor may not be suitable for formula prediction, (2) it doesn't leverage hierarchical table structures. FORTAP also outperforms TUTA by $+7.3\%$, showing formula pretraining effectively assists formula prediction. We also experiment under a low-resource setting (20% training data), and the improvements of FORTAP are more significant, surpassing TUTA by $+10.2\%$. Since Enron is not included in our pretraining corpus, this result well indicates formula pretraining can largely benefit formula prediction after seeing large numbers of real formulas. Moreover, we conjecture that formula pretraining potentially improves numerical reasoning capabilities of the model, because the two-stage prediction of formula sketches and ranges relies on numerical calculation and reference capabilities, respectively.

4.3 Table Question Answering

Table QA (Pasupat and Liang, 2015; Cheng *et al.*, 2021) contains a table and an NL question over the table as the model input. Its output can be cell value(s) or number(s) calculated over numerical cell value(s). Table QA calls for both in-table numerical reasoning and table-text joint reasoning.

Datasets. There are several datasets (Pasupat and Liang, 2015; Cheng *et al.*, 2021; Zhu *et al.*, 2021; Chen *et al.*, 2021b) focusing on Table QA or Table-text hybrid QA. We choose to evaluate on HiTab (Cheng *et al.*, 2021), a hierarchical web table dataset for question answering and data-to-text. First, tables in HiTab contain rich table structures (98.1% tables are hierarchical) from 29 domains, posing a challenge to numerical reasoning. Second, a large proportion of questions ($\sim$ 40%) from Statistical Reports demands complex numerical inference over table and text. Moreover, questions in HiTab are revised from sentences written by professional analysts to ensure naturalness and meaningfulness. The QA evaluation metric is Execution Accuracy measuring the percentage of correctly predicted answers.

Baselines. We employ TaPas (Herzig *et al.*, 2020), HiTab model (Cheng *et al.*, 2021), TaPEx (Liu *et al.*, 2021), and TUTA as our baselines. TaPas is an end-to-end table parsing model without generating logical forms, which enjoys pretraining on the large-scale table-text corpus from Wikipedia. HiTab devises a hierarchy-aware logical form for hierarchical tables, and predicts the answer using a weakly supervised semantic parser MAPO (Liang *et al.*, 2018), which is a reinforcement learning framework to systematically explore and generate programs. The question and table are encoded by BERT and the logical forms are generated by an LSTM decoder. TaPEx is introduced in Section 4.2.

Fine-tune. We replace the BERT encoder of HiTab model with TUTA and FORTAP , and follow the fine-tuning settings of HiTab. We find that NRP pretrain task under table-text setting mentioned in Section 3 is quite essential for QA performance and thus pretrain 80,000 steps more with it on FORTAP in QA before fine-tuning. For TaPEx, we adopt the same table QA strategy in its paper by inputting the table and text as source, and generating the answer as target. The TaPEx-base model is trained for 20,000 steps on HiTab.

Results. Table 3 summarizes QA results on HiTab. FORTAP achieves SOTA (47.0%) using MAPO as the semantic parser, surpassing the best system in HiTab paper with +6.3%. Meanwhile, replacing BERT with TUTA does not see a significant performance gain. We conjecture one of the reasons is that TUTA may be not skilled at table-text joint reasoning, and FORTAP enhances this skill by the table-text setting of the NRP task. Finally, FORTAP performs comparatively with TaPEx, a recent pretraining tabular model as a powerful neural SQL executor targeting table reasoning. Note that this result is inspiring since FORTAP is pretrained on spreadsheet tables and can generalize to web table domain (HiTab) with SOTA performance, indicating that the numerical reasoning skills learnt by FORTAP are robust to distinct scenarios.

(%)	Development	Test
TaPas	39.7	38.9
BERT (*MAPO*)	43.5	40.7
TUTA (*MAPO*)	43.5	41.3
TaPEx	**48.8**	45.6
FORTAP (*MAPO*)	47.1	**47.0**

Table 3: QA execution accuracy on HiTab. *MAPO* means using MAPO+hierarchical-aware logical forms.

(%)	M	N	Data	LA	TA	Derived	Avg.
CNN^{BERT}	76.3	1.5	95.2	59.0	75.4	57.6	60.8
$\text{RNN}^{\text{C+S}}$	62.7	40.8	98.6	56.9	73.5	48.8	63.6
TaBERT	66.6	5.4	94.3	29.2	59.2	45.1	50.0
TaPas	80.6	20.3	96.5	56.9	**90.1**	56.6	66.8
TUTA	**86.0**	41.6	99.1	76.7	82.0	73.1	76.4
FORTAP	85.2	**49.1**	**99.3**	**78.0**	86.4	**79.7**	**79.6**

Table 4: F1 scores of cell type classification on DeEx: **M**(`metadata`), **N**(`notes`), **Data**, **LA**(`left attribute`), **TA**(`top attribute`) , and **Derived**.

4.4 Cell Type Classification

Cell type classification (CTC) (Koci *et al.*, 2019; Gol *et al.*, 2019; Gonsior *et al.*, 2020) aims to interpret tabular data layouts automatically via classifying table cells by their roles in data layouts (e.g., top attribute, data, derived). It requires understanding of table semantics, structures, and numerical relationships considering diverse table layouts.

Datasets. DeEx (Koci *et al.*, 2019) is a widely-studied CTC dataset with tables of various structures and semantics. DeEx includes tables from various domains by mixing three public corpora: Enron (Hermans and Murphy-Hill), Euses (Fisher and Rothermel, 2005), and Fuse(Barik *et al.*, 2015). Cells in DeEx are categorized into six fine-grained types: `metadata`, `notes`, `data`, `left attribute`, `top attribute`, and `derived`. The evaluation metric is the Macro-F1 score over all cell types.

Baselines. We compare FORTAP with two learning-based methods CNN^{BERT}(Dong *et al.*, 2019a) and Bi-LSTM (Gol *et al.*, 2019), and three table-pretraining methods TaBERT (Yin *et al.*, 2020), TaPas (Herzig *et al.*, 2020), and TUTA.

Fine-tune. To handle large tables in DeEx, we split tables into chunks with a max input sequence length (512) and distribute headers to each chunk. For cells with formulas, `[FORMULA]` tags are used as input tokens. We fine-tune 100 epochs on five folds and report the average scores. All these settings are the same as TUTA.

Table 4 lists the CTC results on DeEx. FORTAP achieves a SOTA Macro-F1 of 79.6%. Specifically, FORTAP largely improves the performance on type `derived` and `notes`, surpassing TUTA by 6.6% and 7.5%. The improvement on `derived` indicates formula pretraining helps identifying cells derived by calculations over some other cells. Note that `derived` in DeEx not only includes cells with explicit formulas, but also those cells with hidden (missing) formulas (Koci *et al.*, 2019), which poses a great challenge to existing methods since it requires discovery of numerical relationships between cells. Thus, this is a strong signal that formula pretraining endows the model with better numerical reasoning capabilities. We think that the improvement on `notes` mainly benefits from the NRP pretraining task with formula-based prompts as the context, enhancing FORTAP 's capability on table-text joint modeling.

4.5 Analysis

In this section, we analyze our method in terms of (1) the effects of different pretraining tasks, (2) whether and to what extent our model learns numerical reasoning skills.

Effects of pretraining tasks. We conduct ablation studies on different pretraining tasks on the formula prediction task. Here we pretrain TUTA with each pretraining task and fine-tune on Enron dataset, as summarized in Table 5. We can see that combining all pretraining tasks brings the most gain on formula accuracy. NRP and NCP improve more on range accuracy and sketch accuracy, respectively. This aligns with our design motivation that NRP targets on how to reference and NCP learns how to calculate. To our surprise, Formula MLM alone also largely benefits formula prediction. We deduce the reason is that both MLM and formula prediction requires encoding and recovering/generating capabilities of the formula token sequence.

Numerical reasoning skills. We have shown our model learns numerical reasoning skills by two facts: (1) NRP and NCP improve more on the range and sketch accuracy on the formula prediction task, respectively; (2) our model boosts the performance of `derived` cell type on cell type classification. Here we further decompose QA accuracy of different operations on HiTab. The comparison between previous SOTA system BERT(MAPO) and our FORTAP (MAPO) is shown in Table 6. As shown, our model improves most on complex cell selection (cell indexed by ≥ 3 headers) and arithmetic (e.g., *difference*, *sum*) problems. Note that complex cell selection not only requires table-text alignment, but also the references between headers considering that mentions of headers in question could be implicit or missing. Meanwhile, our model also handles superlative (e.g., *argmax*) and comparative (e.g., *less than*) problems better than BERT, despite these types are relatively infrequent in our formula pretraining corpus. To summarize, our model mainly improves numerical skills regarding cell reference and arithmetic, as well as other aspects like comparing and ranking.

(%)	**Formula**	**Sketch**	**Range**
TUTA	48.5	65.3	75.3
TUTA + NRP	54.3	69.0	78.7
TUTA + NCP	54.7	**71.2**	76.8
TUTA + FormulaMLM	54.6	70.2	77.7
All (FORTAP)	**55.8**	70.8	**78.8**

Table 5: Ablation study on formula prediction.

Operation	**BERT**	**FORTAP**
Complex Cell Selection	48.4%	56.4% (+8.0%)
Arithmetic	6.0%	13.3% (+7.3%)
Superlative	22.7%	26.8% (+4.1%)
Comparative	27.5%	30.5% (+3.0%)

Table 6: Accuracy on HiTab of different operations.

5 Related Works

Table Pretraining. Table pretraining has been widely studied in recent years. Some works mine large-scale table-text pairs as pretraining corpus (Deng *et al.*, 2020b; Yin *et al.*, 2020; Herzig *et al.*, 2020; Wang *et al.*, 2021b), some leverage annotated table-text datasets (Deng *et al.*, 2021; Yu *et al.*, 2020), and some synthesize a table-text corpus by templates (Yu *et al.*, 2020; Eisenschlos *et al.*, 2020). Regarding pretraining tasks, they either train the model to recover masked tokens/column/cell/entity (Yin *et al.*, 2020; Herzig *et al.*, 2020; Wang *et al.*, 2021b; Deng *et al.*, 2020b), or explicitly learn table-text alignments (Deng *et al.*, 2021; Yu *et al.*, 2020). Recently, TaPEx (Liu *et al.*, 2021) adopts BART (Lewis *et al.*) as a neural executor for synthesized SQLs to improve table reasoning. Whereas, our method explores to use real spreadsheet formulas to guide table pretraining.

Numerical reasoning over Natural Language. Numerical reasoning is important in NL domain (Dua *et al.*, 2019). Numbers even account for 6.15% of all unique tokens in English Wikipedia (Thawani *et al.*, 2021). Various works target improving numerical reasoning skills on NL (Andor *et al.*, 2019; Geva *et al.*, 2020; Jin *et al.*, 2021). Except using pure NL, MathBERT (Peng *et al.*, 2021) pretrains NL documents with mathematical formulas. In this paper, we target numerical reasoning over (semi-) structured tables.

6 Conclusion

In this paper, we present FORTAP , a numerical-reasoning-aware table pretraining model that learns numerical reasoning capabilities from spreadsheet formulas. Specifically, we design two pretraining tasks to capture numerical reasoning capabilities by explicitly predicting cell reference and calculation relations. Experiments show that FORTAP achieves new SOTA on formula prediction, question answering, and cell type classification. Further analyses indicate that formula pretraining indeed improves numerical reasoning skills of the model. One limitation of FORTAP is that we haven't fully exploit spreadsheet formulas beyond numerical reasoning. For example, logic functions like `VLOOKUP` and text functions like `LEN` can be leveraged to guide complex logic and text reasoning, which will be a promising direction in the future.

7 Ethical Considerations

In this work, we present a table pretraining method leveraging spreadsheet formulas.

Dataset. Our pretraing corpus is built upon public English spreadsheet files crawled from webs via the search engine (Wang *et al.*, 2021b), covers various domains, and has been checked by a compliance team in a company to ensure that does not contain sensitive names or uniquely identifies individual people or offensive content. All datasets used for evaluation are licensed public datasets, e.g., for formula prediction, Enron (Hermans and Murphy-Hill) is a public spreadsheet dataset consisting of over 17K spreadsheet files, and we re-purpose it for formula prediction following (Chen *et al.*, 2021a).

Application. Our model shows its effectiveness in three representative table-related tasks. Formula prediction helps spreadsheet end-users to write formulas which could be tedious and error-prone. Table QA enables users to query on the table without the need of domain background knowledge. Cell type classification assists interpreting fine-grained table semantic structures, which help users to better understand table structures and contents. There may be risks that crooks use tabular models to automatically parse tables/forms to obtain private personal or company data in bulk, which should be prevented.

References

Efthimia Aivaloglou, David Hoepelman, and Felienne Hermans. A grammar for spreadsheet formulas evaluated on two large datasets. In *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 121–130. IEEE, 2015.

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. Giving bert a calculator: Finding operations and arguments with reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5947–5952, 2019.

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.

Chris Chambers and Martin Erwig. Dimension inference in spreadsheets. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 123–130. IEEE, 2008.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W Cohen. Open question answering over tables and text. *arXiv preprint arXiv:2010.10439*, 2020.

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*, pages 1661–1672. PMLR, 2021.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*, 2021.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. Hitab: A hierarchical table dataset for question answering and natural language generation. *arXiv preprint arXiv:2108.06712*, 2021.

Shing-Chi Cheung, Wanjun Chen, Yepang Liu, and Chang Xu. Custodes: automatic spreadsheet cell clustering and smell detection using strong and weak features. In *Proceedings of the 38th International Conference on Software Engineering*, pages 464–475, 2016.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-grounded pretraining for text-to-sql. *arXiv preprint arXiv:2010.12773*, 2020.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. Turl: Table understanding through representation learning. *arXiv preprint arXiv:2006.14806*, 2020.

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: Human Language Technologies*, pages 1337–1350, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia, July 2018. Association for Computational Linguistics.

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.

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.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, 2019.

Julian Martin Eisenschlos, Syrine Krichene, and Thomas Müller. Understanding tables with intermediate pre-training. *arXiv preprint arXiv:2010.00571*, 2020.

Marc Fisher and Gregg Rothermel. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *Proceedings of the first workshop on End-user software engineering*, pages 1–5, 2005.

Mor Geva, Ankit Gupta, and Jonathan Berant. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, 2020.

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 Computer Society, 2019.

Julius Gonsior, Josephine Rehak, Maik Thiele, Elvis Koci, Michael Günther, and Wolfgang Lehner. Active learning for spreadsheet cell classification. In *EDBT/ICDT Workshops*, 2020.

Felienne Hermans and Emerson Murphy-Hill. Enron's spreadsheets and related emails: A dataset and analysis. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*.

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.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584*, 2021.

Zhihua Jin, Xin Jiang, Xingbo Wang, Qun Liu, Yong Wang, Xiaozhe Ren, and Huamin Qu. Numgpt: Improving numeracy ability of generative pre-trained models. *arXiv preprint arXiv:2109.03137*, 2021.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. Memory augmented policy optimization for program synthesis and semantic parsing. In *NeurIPS*, 2018.

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.

Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. Learning to reason for text generation from scientific tables. *arXiv preprint arXiv:2104.08296*, 2021.

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 (Volume 1: Long Papers)*, pages 1470–1480, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*, 2021.

Basel Shbita, Arunkumar Rajendran, Jay Pujara, and Craig A Knoblock. Parsing, representing and transforming units of measure. *Modeling the World's Systems*, 2019.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. Towards table-to-text generation with numerical reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 1451–1465, 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. *arXiv preprint arXiv:2012.02469*, 2020.

Avijit Thawani, Jay Pujara, and Filip Ilievski. Numeracy enhances the literacy of language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6960–6967, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

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.

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.

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*, pages 8413–8426, 2020.

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.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*, 2020.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*, 2021.

A Involved Operators/Functions of Formula

We include 17 common operators/functions in Numerical Calculation Prediction pretraining task, which consists of all the operators and four most commonly used aggregation functions in spreadsheet formula. The operators/functions are: $+$, $-$, $*$, $/$, $\wedge$, $\%$, $\&$, $=$, $<>$, $>$, $<$, $\geq$, $\leq$, `SUM`, `AVERAGE`, `MAX`, `MIN`.

To encode formula token sequence, we expand 41 tokens in vocabulary for all four formula token types `OP`, `FUNC`, `CELL`, `CONST`, covering 99.1% formulas in corpus. Here we list these tokens: (1) 1 token for `CELL` token type: `[RANGE]`. (2) 3 tokens for `CONST` token type: `[C-STR]`, `[C-NUM]`, `[C-BOOL]`. All constant tokens are categorized according to "string", "number", and "bool". And they are replaced with these three tokens when encoding the formula. (3) 34 tokens for `OP`/`FUNC` token type: [+](32.1%), `[SUM]`(20.6%), [−](17.8%), [/](6.7%), `[IF]`(2.6%), `[ROUND]`(1.2%), `[AVERAGE]`(1.2%), `[VLOOKUP]`(1.0%), [>](0.98%), [=](0.79%), [<](0.57%), `[ABS]`(<0.5%), `[OFFSET]`, `[SUBTOTAL]`, `[MAX]`, [<>], [∧], `[LN]`, `[COUNTA]`, `[SQRT]`, `[MIN]`, `[ISERROR]`, `[EOMONTH]`, `[COUNT]`, `[AND]`, [%], `[INDEX]`, `[YEAR]`, `[MONTH]`, `[MATCH]`, [≥], `[MATCH]`, [≤], [&], `[UNKOP]`. The number in parentheses is the ratio of `OP`/`FUNC` to the total number of `OP`/`FUNC` in corpus. Here `UNKOP` stands for unknown operator/function, similar to `[UNK]` in NL vocabulary. To distinguish formula `OP`/`FUNC` with some eponymous tokens in vocabulary (e.g., "sum", "+"), we enclose formula `OP`/`FUNC` with square brackets. (4) special tokens `[START]`, `[END]`, [:].

B Implementation Details

More on Hyperparameters. For pretraining, we first pretrain $400K$ steps with max sequence length 256, batch size 32, then pretrain $250K$ steps with max sequence length 512, batch size 8. The whole pretraining phase is estimated to 3 epochs, i.e., samples in the corpus are seen 3 times in pretraining. The optimizer is Adam with learning rate $2e$-5.

For formula prediction, we set max sequence length 512 and fine-tune $800K$ steps with batch size 2 on single GPU. The tokens beyond 512 are truncated. If the formula cell is truncated (rare case), we input the `[CLS]` embedding to the formula decoder. The two-stage decoder is first trained $100K$ for generating sketches, and then trained to generate sketches and ranges together. The optimizer is Adam with learning rate $2e$-5.

For table question answering, we follow HiTab hyperparameters except that we find it is unnecessary to freeze encoder parameters at the first $5,000$ steps, so we train the encoder-decoder model together.

For cell type classification, since some tables are extremely large in DeEx, we truncate the tables into sequences of max length 512 by preserving the header cells (both top and left) and traversing the data cells to fill the max sequence length. We fine-tune 100 epochs on five folds with batch size 12. The optimizer is Adam with learning rate $8e$-6.

SpreasheetCoder We implement SpreadsheetCoder mainly following its paper including the BERT-based table context (row/column) encoder, two-stage decoder. One difference is that we did not implement the convolution layers for row and columns which is rather complicated . Instead, since SpreadsheetCoder uses convolution layer aiming to incorporate contextual information from different positions (row/column), we explicitly add row embeddings and column embeddings (Herzig *et al.*, 2020) for input table tokens, which derives the similar accuracy gain of convolution layers (4% according to its paper), from 35.6% to 40.4% on Enron dataset. Furthermore, SpreadsheetCoder can only decode referenced cells in a rectangle window ($[-10, 10]$) of the target cell since it only keeps the formulas of this kind in dataset. We enable SpreadsheetCoder to predict referenced cells in a larger window which it can not solve by extending the vocabulary of range tokens from $[-10, 10]$ to $[-256, 256]$. Different from SpreadsheetCoder, FORTAP predicts ranges by selecting from input table cells instead of from a fixed cell vocabulary. In this way, theoretically (without memory limit) our model can potentially predict referenced cells in an arbitrarily large table. Detailed error analysis of FORTAP on formula prediction is in Appendix C.

C Error Analysis of Formula Prediction

Figure 3 presents the proportion and accuracy regarding different formula sketch lengths in prefix order (parentheses excluded). As shown, sketch length 3 and 4 account for two-thirds of formulas, since length 3 is typical for binary operations like `C4-B4`, and length 4 is a common pattern for ag-

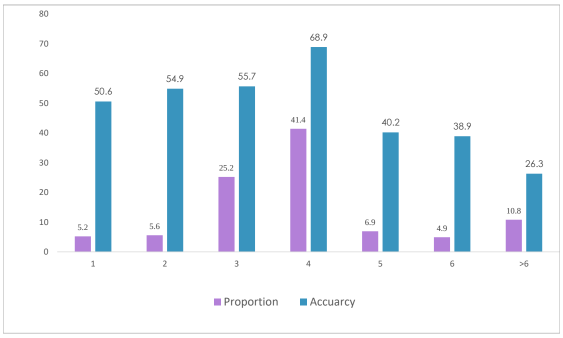

Figure 3: Proportion and accuracy of samples with different formula sketch lengths in formula prediction task.

gregation functions like `SUM(B4:C5)`. Thus, the accuracy of length 3/4 is higher than shorter sketch length 1/2 since more samples in its length are seen in training. And for longer formulas (>6), a significant performance drop occurs because complex nested references and calculations may be involved when the sketch gets longer.

To further analyze the errors in formula prediction, we randomly pick 100 false generation results in dev set and divide these errors into three groups: (i) sketch failure (54%): a wrong sketch is generated, which occurs more frequently when the formula gets longer and nested. A typical case is the formula with function `IF`, involving multiple arguments and nested calculations; (ii) reference unreachable (27%): referenced cells are not in the sequence since we only consider the cells on the same row/column of the target cell as input; (iii) reference failure (19%): wrong referenced cells are selected, which often occurs at the start or end of a cell range. Future works may improve formula prediction in these directions: handling long nested formulas, inputting more cells of table matrix as reference candidates conquering memory issues, and designing a module to match generated sketch with input table cells more accurately.

D Real examples of spreadsheet tables with formulas

Here we show several real examples for spreadsheet tables in Figure [4-6].

E Real examples of formula prediction on Enron

We also developed an Excel plug-in to run formula prediction powered by ForTaP. We simulate that ForTap suggests formulas for a user when she is editing a spreadsheet. Here we show several formula prediction demonstrations on Enron test set in Figure [7-11]. For the fist case, we tried different column names, and the results are promising and robust.

SUM | =N3-K3

	J	K	L	M	N	O	P	Q
1	Next Month Delivery Risk	Current PMTM	Current & Prior Delivery Risk Plus Current PMTM	Net Exposure	PMTM Next Month Forward	Current Next Exposure	Net Exposure as of Next Month	Change in MTM (Next Mth - Current Mth)
2	$0	$0	$0	($2,150)	$0	$0	($2,150)	$0
3	$713,400	$10,850,119	$11,692,315	$10,590,699	$10,772,145	$12,327,741	$11,226,125	=N3-K3
4	$0	$8,600	$8,600	$8,600	$8,600	$8,600	$8,600	$0
5	$4,993,800	$399,422	$8,869,750	$8,869,750	($525,365)	$12,938,763	$12,938,763	($924,787)
6	$140,026	$1,380,990	$1,654,942	$1,654,942	$1,337,462	$1,751,440	$1,751,440	($43,528)
7	$0	$0	$0	($24,311)	$0	$0	($24,311)	$0
8	$274,320	$2,288,770	$2,872,168	$2,872,168	$2,212,131	$3,069,849	$3,069,849	($76,639)
9	$0	$0	$725	$725	$0	$725	$725	$0
10	$25,400,370	$120,817,047	$196,379,266	$196,379,266	$102,679,938	$203,642,527	$203,642,527	($18,137,109)
11	$0	$0	$0	($53,761)	$0	$0	($53,761)	$0
12	$3,909,360	$3,480,004	$9,087,662	$9,087,662	$2,454,403	$11,971,421	$11,971,421	($1,025,601)
13	$0	($18,167)	($683)	($683)	($18,167)	($683)	($683)	$0

Figure 4: Example 1 with a substraction column.

B23 | =B6+B8+B17

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Table 4 -- U.S. sugar: supply and use, by fiscal year (Oct./Sept.)												
2	Items	2000/01	2001/02	2002/03	2003/04	2004/05	2005/06	2006/07	2007/08	2008/09	2009/10	2010/11	2011/12
3													
4													
5							1,000 short tons, raw value						
6	Beginning stocks	2,216	2,180	1,528	1,670	1,897	1,332	1,698	1,799	1,664	1,534	1,498	1,472
7													
8	Total production	8,769	7,900	8,426	8,649	7,876	7,399	8,445	8,152	7,531	7,963	7,831	8,160
9	Beet sugar	4,680	3,915	4,462	4,692	4,611	4,444	5,008	4,721	4,214	4,575	4,659	4,655
10	Cane sugar	4,089	3,985	3,964	3,957	3,265	2,955	3,438	3,431	3,317	3,387	3,172	3,505
11	Florida	2,057	1,980	2,129	2,154	1,693	1,367	1,719	1,645	1,577	1,646	1,433	1,790
12	Louisiana	1,585	1,580	1,367	1,377	1,157	1,190	1,320	1,446	1,397	1,469	1,411	1,400
13	Texas	206	174	191	175	158	175	177	158	152	112	146	145
14	Hawaii	241	251	276	251	258	223	222	182	192	161	182	170
15	Puerto Rico	0	0	0	0	0	0	0	0	0	0		
16													
17	Total imports	1,590	1,535	1,730	1,750	2,100	3,443	2,080	2,620	3,082	3,320	3,738	2,820
18	Tariff-rate quota imports	1,277	1,158	1,210	1,226	1,408	2,588	1,624	1,354	1,370	1,854	1,721	1,580
19	Other Program Imports	238	296	488	464	500	349	390	565	308	448	291	500
20	Non-program imports	76	81	32	60	192	506	66	701	1,404	1,017	1,726	740
21	Mexico							60	694	1,402	807	1,708	730
22													
23	Total Supply	12,575	11,615	11,684	12,070	11,873	12,174	12,223	12,571	12,277	12,817	13,067	12,452

Figure 5: Example 2 with a total row.

G5 | =C5/F5

	A	B	C	D	E	F	G
2		**Table 1.a Utilities/Communities Eligible for PCE, 2010**					
3		**By AEA Energy Regions**					
4		**AEA Energy Region**	**Yes**	**Inactive**	**No**	**Total**	**Percent Active in PCE program**
5		Aleutians	12	1	0	13	92%
6		Bering Straits	17	0	0	17	100%
7		Bristol Bay	25	1	0	26	96%
8		Copper River/Chugach	6	0	2	8	75%
9		Kodiak	4	1	1	6	67%
10		Lower Yukon-Kuskokwim	48	0	0	48	100%
11		North Slope	7	1	0	8	88%
12		Northwest Arctic	12	1	0	13	92%
13		Railbelt	0	0	14	14	0%
14		Southeast	21		10	31	68%
15		Yukon-Koyukuk/Upper Tanana	38	3	2	43	88%
16		**Total**	**190**	**8**	**29**	**227**	**84%**
17		*Note: For utilities that serve many communities with no grid such as AVEC and AP&T, each community is counted as a separate utility.*					
18							

Figure 6: Example 3 with a total row and a proportion column.

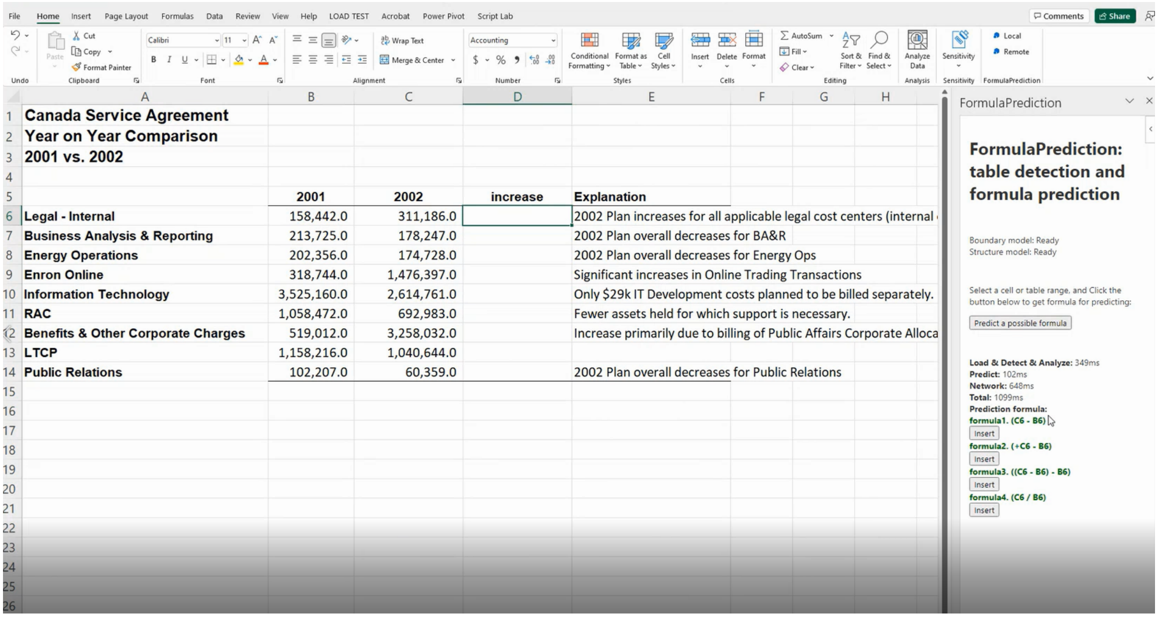

Figure 7: Example 1 modified on Enron test set for formula prediction.

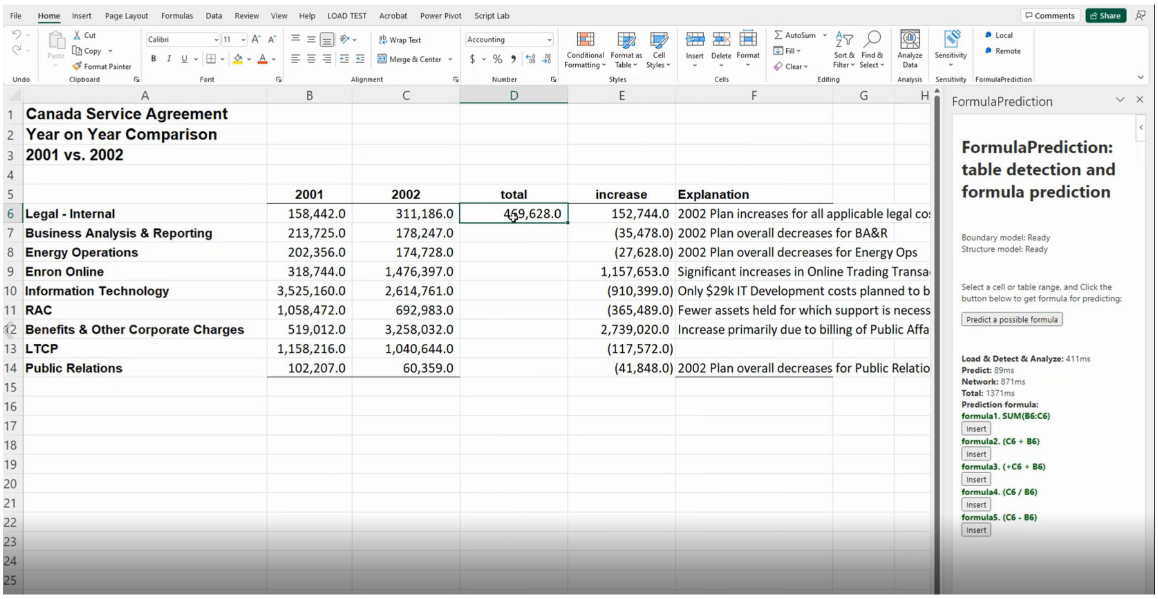

Figure 8: Example 2 modified on Enron test set for formula prediction.

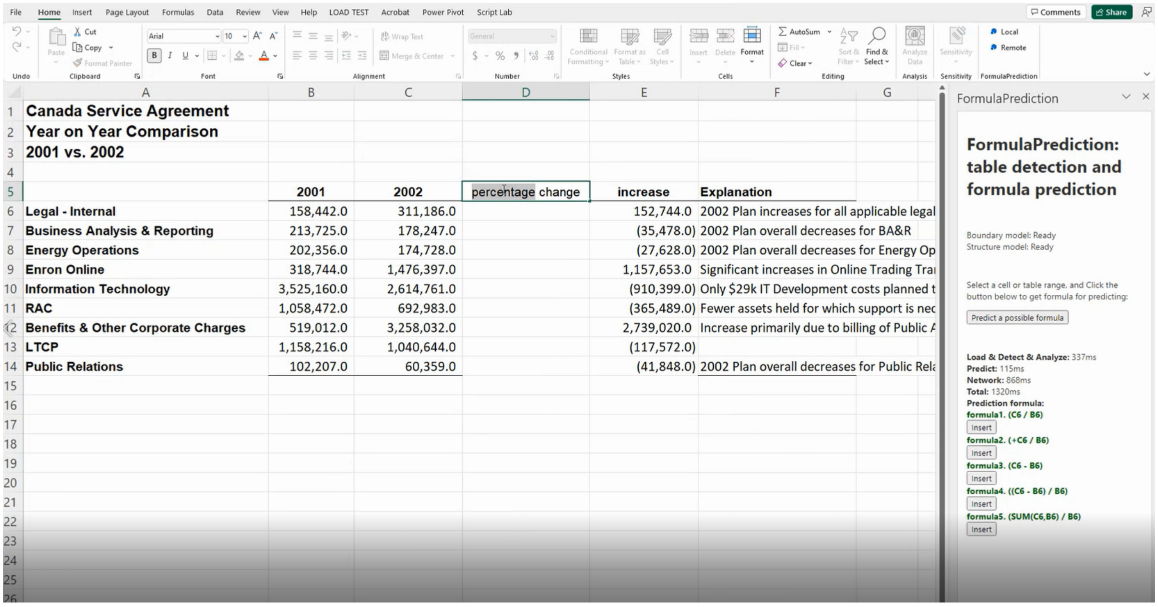

Figure 9: Example 3 modified on Enron test set for formula prediction.

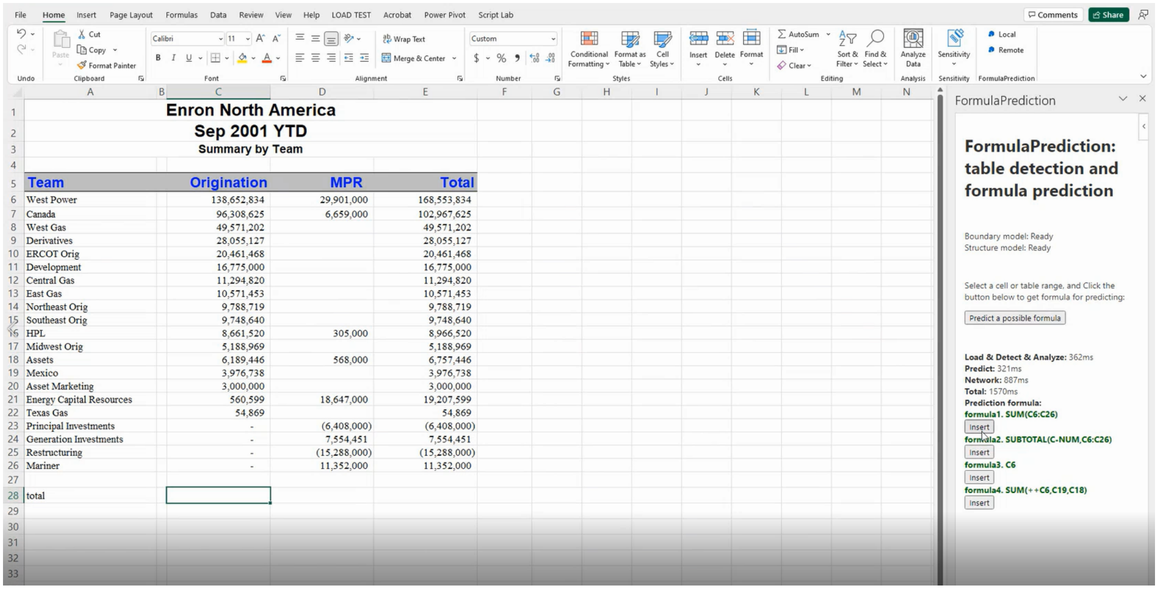

Figure 10: Example 4 modified on Enron test set for formula prediction.

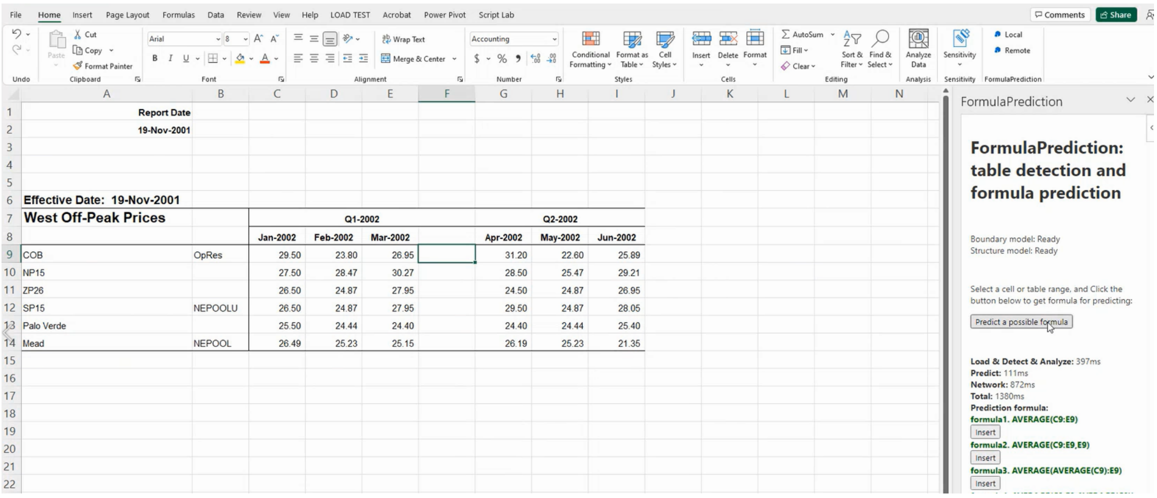

Figure 11: Example 5 on Enron test set for formula prediction.