HiTab: A Hierarchical Table Dataset for Question Answering and Natural Language Generation

Zhoujun Cheng*
Shanghai Jiao Tong University
blankcheng@sjtu.edu.cn

Haoyu Dong* †
Microsoft Research Asia
hadong@microsoft.com

Zhiruo Wang*
Carnegie Mellon University
zhiruowandrew.cmu.edu

Ran Jia
Microsoft Research Asia
jia.ran@microsoft.com

Jiaqi Guo
Xi’an Jiaotong University
jasperguo2013@stu.xjtu.edu.cn

Yan Gao
Microsoft Research Asia
Yan.Gao@microsoft.com

Shi Han
Microsoft Research Asia
shihan@microsoft.com

Jian-Guang Lou
Microsoft Research Asia
jlou@microsoft.com

Dongmei Zhang
Microsoft Research Asia
dongmeiz@microsoft.com

ABSTRACT

Tables are often created with hierarchies, but existing works on table reasoning mainly focus on flat tables and neglect hierarchical tables. Hierarchical tables challenge existing methods by hierarchical indexing, as well as implicit relationships of calculation and semantics. This work presents HiTab, a free and open dataset for the research community to study question answering (QA) and natural language generation (NLG) over hierarchical tables. HiTab is a cross-domain dataset constructed from a wealth of statistical reports and Wikipedia pages, and has unique characteristics: (1) nearly all tables are hierarchical, and (2) both target sentences for NLG and questions for QA are revised from high-quality descriptions in statistical reports that are meaningful and diverse. (3) HiTab provides fine-grained annotations on both entity and quantity alignment. Targeting hierarchical structure, we devise a novel hierarchy-aware logical form for symbolic reasoning over tables, which shows high effectiveness. Then given annotations of entity and quantity alignment, we propose partially supervised training, which helps models to largely reduce spurious predictions in the QA task. In the NLG task, we find that entity and quantity alignment also helps NLG models to generate better results in a conditional generation setting. Experiment results of state-of-the-art baselines suggest that this dataset presents a strong challenge and a valuable benchmark for future research.

CCS CONCEPTS

• Information systems → Information retrieval.

KEYWORDS

semi-structured data, question answering, data-to-text

ACM Reference Format:
Zhoujun Cheng*, Haoyu Dong* †, Zhiruo Wang*, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A Hierarchical Table Dataset for Question Answering and Natural Language Generation. In Proceedings of ACM Conference (Conference’17). ACM, New York, NY, USA, 10 pages.

1 INTRODUCTION

In recent several years, there are a flurry of works on reasoning over semi-structured tables, e.g., answering questions over tables [38, 53] and generating fluent and faithful text from tables [24, 37]. But they mainly focus on simple flat tables and neglect complex tables, e.g., hierarchical tables. A table is regarded as hierarchical if its header exhibits a multi-level structure [6, 30, 49]. Hierarchical tables are widely used, especially in data products, statistical reports, and research papers in government, finance, and science-related domains.

Hierarchical tables challenge QA and NLG due to: (1) Hierarchical indexing. Hierarchical headers, such as D2:G3 and A4:A25 in Figure 1, are informative and intuitive for readers, but cell selection in hierarchical tables is much more compositional than flat tables, requiring multi-level and bi-dimensional indexing. For example, to select the cell E5 (“66.6”), one needs to specify two top header cells, “Master’s” and “Percent”, and two left header cells, “All full-time” and “Self-support”. (2) Implicit calculation relationships among quantities. In hierarchical tables, it is common to insert various aggregated rows and columns, e.g., total (columns B,D,F and rows 4,6,7,20) and proportion (columns C,E,G). But hierarchical tables lack explicit indications to quantity relationships, and thus challenge precise numerical inference in QA and NLG. (3) Implicit semantic relationships among entities. Hierarchical tables lack explicit indications to entity relationships, e.g., “source” and “mechanism” in A2 describe A6:A19 and A20:A25 respectively, and D2 (“Master’s”) and F2 (“Doctoral”) can be jointly described by a virtual entity, “Degree”. How to identify semantic relationships and link entities correctly for QA and NLG is also a challenge.

In this paper, we aim to build a dataset for hierarchical table QA and NLG. But without sufficient data analysts, it’s hard to ensure

https://www.nsf.gov/statistics/2019/nsf19319/
• Teaching assistantships were most commonly reported as the primary mechanism of support for master’s students (11%).

Figure 1: A hierarchical table and accompanied descriptions in an National Science Foundation report.†

1. Teaching assistantships were as the primary mechanism of support for master’s students (11%).

We devise a process to construct QA pairs based on existing high-quality sentence descriptions instead of asking labelers to propose questions from scratch. Annotators convert sentence descriptions to question-answering pairs and use spreadsheet formulas to record the calculation process of answering, as Table 1 shows.

Experiment results suggest that HiTab presents a strong challenge to state-of-the-art baselines. For the QA task, TAPAS [18] only achieves 38.9% accuracy; MAPO [29] performs even worse (29.2% accuracy) due to the ineffectiveness of the logical form customized for flat tables. For the NLG task, models also have a great difficulty in understanding hierarchies and generating meaningful texts. To leverage characteristics of hierarchical tables, we first devise a hierarchy-aware logical form for table QA, which shows high effectiveness. Then we propose partially supervised training given annotations of linked mentions and formulas, which helps models to largely reduce spurious predictions and achieve 45.1% accuracy in the QA task. In the NLG task, we dig deeper into controllable generation [37], showing that both aligned cells and the calculation process help models to generated meaningful texts.

Code and data are provided in https://github.com/microsoft/HiTab.

2 DATASET CONSTRUCTION AND ANALYSIS

To well-handle the complexity of our annotation task, we recruit 18 students or graduates (13 females and 5 males) in computer science, finance, and English majors from top universities. Each student is paid $7.8 an hour, and they totally spend 2,400 hours. We propose an annotation process with six steps (Section 2.1-2.6).

2.1 Hierarchical Table Collection

A large number of reports from various organizations are publicly available. We select two representative organizations, Statistics Canada [45] and National Science Foundation [35]. Different from [2–4, 20] that only provide PDF reports, StAtCan and NSF also additionally provide HTML reports, in which cell information such as text and formats can be extracted in precise using HTML tags.

First, we crawl English HTML statistical reports published in recent five years from StatCan (1,083 reports in 27 well-categorized domains) and NSF (208 reports from 11 organizations in science foundation domain). We merge StatCan and NSF and get a total of 28 domains. In addition, we find that ToTTo [37] contains a small proportion (5.03%) of hierarchical tables, then we include them into HiTab so that HiTab has additional open domain tables from Wikipedia. To keep the balance between tables from statistical reports and Wikipedia pages, we only randomly include 40% (1,851) of tables in ToTTo. Next, we transform HTML tables to spreadsheet tables using a preprocessing script. Thus annotators can use Excel formulas to align quantities and answer questions. To enable correct formula execution in Excel, we normalize quantities in data cells by excluding surrounding superscripts, internal commas, etc.

We filter tables using these constraints: (1) number of rows and columns are more than 2 and less than 64; (2) cell strings have no more than one non-ASCII character and 20 tokens; (3) hierarchies are successfully parsed via the method in 2.6. (4) hierarchies have no more than four levels. Finally, 85% tables meet all constraints.

2.2 Sentence Extraction and Revision

In this step, annotators manually go through the reports and extract all sentence descriptions for each table. Sentences consisting of multiple semantic-independent sub-sentences will be carefully split into multiple ones. Annotators are instructed to eliminate redundancy...
and ambiguity in sentences through revisions including decontextualization and phrase deletion like [37]. Fortunately, most sentences in statistical reports are clean and fully supported by table data, so few revisions are needed to get high-quality target text for NLG.

2.3 Entity and Quantity Alignment
In this phase, annotators are instructed to align mentions in text with corresponding cells in tables. It has two parts, entity alignment and quantity alignment, as shown in Table 1. For entity alignment, we record the mappings from entity mentions in text to corresponding cells. Single-cell quantity mentions can be linked similar with entity mentions, but composite quantity mentions are calculated from two or more cells through operators like max/sum/div/diff. The spreadsheet formula is powerful and easy-to-use for tabular data calculation, so we use the formula to record the calculations process of composite quantities in text, e.g., ‘10 points higher’ (=G23-G24).

Although quantities are often rounded in descriptions, we neglect rounding and refer to precise quantities in table cells.

2.4 Converting Declarative Sentences to QA Pairs
Existing QA datasets instruct annotators to propose questions from scratch, but it’s hard to guarantee the meaningfulness and diversity of proposed questions. In HiTab, we simply convert declarative sentences to produce question-answering pairs. For each sentence, annotators need to identify a target key part to question about (according to the underlying logic of the sentence), then convert it to a QA form. All questions are answered by formulas that reflect the numerical inference process. For example, the ‘XLOOKUP’ operator is frequently used to retrieve the header cells of superlatives, as shown in Table 1. To keep sentences as natural as they are, we do not encourage unnecessary sentence modification during the conversion. If an annotator finds multiple ways to question regarding a sentence, she only needs to choose the way that best reflects the overall meaning.

2.5 Regular Inspections and the Final Review
We ask two most experienced annotators to perform regular inspections and the final review. (1) In the labeling process, they regularly sample annotations (about 10%) from all labelers to give timely feedback on labeling issues. (2) Finally, they review all annotations and fix labeling errors. Also, to assist the final review, we use an automatic script to identify spelling issues and formula issues.

2.6 Hierarchy Extraction
We follow existing work [6, 30, 49] and use the tree structure to model hierarchical headers. Since cell formats such as merging, indentation, and font bold are commonly used to present hierarchies, we adapt heuristics in [49] to extract top and left hierarchical trees, which has high accuracy. We go through 50 randomly sampled tables in HiTab. 94% of them are precisely extracted.

Table 1: Examples of the annotation process. All sentences describe the table in Figure 1.

| Original                                                                 | After revision                                                                 | Entity & quantity alignment                                                                 | Question-answering conversion                                                                 |
|------------------------------------------------------------------------|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| Two-thirds (67%) of master’s students and only one-tenth (10%) of doctoral students were self-supported (table 3). | Two-thirds (67%) of master’s students and only one-tenth (10%) of doctoral students were self-supported. | teaching assistantships $\rightarrow$ =A24                                                | Which is the primary mechanism of support for master’s students?                             |
| Teaching assistantships were most commonly reported as the primary mechanism of support for master’s students (11%). | teaching assistantships $\rightarrow$ =A24 mechanism of support $\rightarrow$ =A20 | master’s $\rightarrow$ =D2                                                          | $\times$XLOOKUP(MAX(E21:E24), E21:E24, A21:A24)                                           |
| For doctoral students, the proportion of support from research assistantships is 10 points higher than that from teaching assistantships. | For doctoral students, the proportion of support from research assistantships is 10 points higher than that from teaching assistantships. | doctoral $\rightarrow$ =F2                                                                 | For doctoral students, what is the difference between the proportions of research assistantships and teaching assistantships? |

Table 2: Example formula templates for operators.
We propose a hierarchy-aware logical form that exploits hierarchies in Table Questions [38]. The question-answer pair should be fully supported by the table. Our dataset \( \mathcal{D} = \{(x_i, t_i, y_i)\}, i \in [1, N] \) is a set of \( N \) question-table-answer triples.

Table QA is usually formulated as a semantic parsing problem [28, 38]. A parser converts question into executable logical forms, and an executor applies logical forms on the table to produce the answer denotation. However, existing logical forms on Table QA [28, 38, 54] are customized for flat or relational database tables. The three challenges mentioned in Section 1 make QA more difficult on hierarchical tables, which are hierarchical indexing, implicit calculation and semantic relationships.

3.1 Hierarchy-aware Logical Forms

We propose a hierarchy-aware logical form that exploits hierarchies to mitigate these challenges. Specifically, we define region as an operating object, and design two functions for hierarchical region selection.

Definitions. Given extracted tree hierarchies of tables introduced in Section 2.6, we define header as a header cell (e.g. A7 (“Federal”) in Figure 1), and level as a level in the left/top tree (e.g. A5,A6,A20 are on the same level). Existing logical forms on tables [29, 38] treat rows as operating objects, limiting operations on the same row. However, a row in hierarchical tables does not represent a record with column names as attributes, thus operations can be applied on cells in the same row. Motivated by this, we define region as our operating object, which is a data region in table indexed by both left and top headers (e.g. A6:A19 is a region, and a region can also be discrete). The logical form execution process is divided into two phases: region selection and region operation.

Region Selection. We design two functions (\( \text{filter\_tree} \) and \( \text{filter\_level} \)) to do region selection, where \( h \) is a header, \( l \) is a level. Each function applies on the return region of previous function by intersection. \( \text{filter\_tree} \) selects a subtree region according to \( h \): if \( h \) is a leaf header (e.g. A8), selected region is the row/column indexed by \( h \) (row 8); if \( h \) is a non-leaf header (e.g. A7), selected region is the rows/columns indexed by both \( h \) and its children headers (row 7-16). \( \text{filter\_level} \) selects the region indexed by headers on target level \( l \) of previously selected subtree region. Design of these two functions mitigate aforementioned three challenges: (1) hierarchical indexing is achieved by tree path selection when applying these two functions sequentially; (2) data with different calculation types (e.g. rows 4,5) will not be co-selected, thus not operated together; (3) a level \( l \) can obtain its semantics by gathering header cell embeddings on it in model. Some logical form execution examples are shown in Appendix A.3.

Region Operation. Operators are applied on the selected region to produce the answer. Composite operators or no operator are both allowed. We define 19 operators mainly following MAPO [29], where some operators (e.g. difference rate) are unique to hierarchical tables. Complete logical form functions are shown in Appendix A.1.

3.2 Experimental Setup

3.2.1 Baselines. We present baselines in two branches on question answering. One is logical form-based semantic parsing, and the other is the recently proposed end-to-end table parsing without logical forms.

Neural Symbolic Machine [28] A powerful semantic parsing framework that consists of a programmer to generate programs from natural language and save intermediate results in memory, and a computer to execute programs. We replace the LSTM encoder of seq2seq programmer with BERT [11], and follow NSM to use a lisp interpreter implementing our logical forms as computer. Table is linearized by placing headers in level order, which is illustrated in detail in Appendix A.2. Note that we do not use TaBERT [52] as the encoder because its core mechanisms are best designed for flat tables and coupled with logical forms for flat tables.

TaPas [18] A state-of-the-art end-to-end table parsing model without generating logical forms. Its power to select cells and reasoning over
We apply pruning rules following [29] in searching. Finally, we further explore to guide training in a partially supervised way. We compare three widely-studied learning paradigms.

Table 4: h.a. stands for hierarchy-aware. QA execution accuracy on dev and test, and spurious program rate of selected 150 samples on dev.

| Method               | Weak Supervision | Partial Supervision |
|----------------------|------------------|---------------------|
|                      | Dev | Test | %Spurious | Dev | Test | %Spurious |
| MAPO w. original logical form | 31.9 | 29.2 | - | - | - | - |
| TaPas w/o. logical form | 39.7 | 38.9 | - | - | - | - |
| MML w. h.a. logical form | 38.9 | 36.7 | 22.7 | - | - | - |
| REINFORCE w. h.a. logical form | 42.7 | 38.4 | 39.3 | - | - | - |
| MAPO w. h.a. logical form | 43.5 | 40.7 | 19.0 | - | - | - |

Table 4: h.a. stands for hierarchy-aware. QA execution accuracy on dev and test, and spurious program rate of selected 150 samples on dev.

3.2.2 Weak Supervision. In weak supervision, the model is trained with QA pairs, without golden logical forms. For NSM, we compare three widely-studied learning paradigms. MML [10] maximizes marginal likelihood of observed programs. REINFORCE [50] maximizes the reward of on-policy samples. MAPO [29] alleviates the biased gradient problem by learning from trajectories both inside and outside the buffer, and samples with high efficiency by systematic exploration.

MML needs to learn from consistent programs, i.e. programs that produce correct answers. REINFORCE and MAPO need consistent programs for warm up. Thus we randomly search 300 iterations (about 15000 programs per sample) for all samples in training set. We apply pruning rules following [29] in searching. Finally, 6.12 consistent programs are searched for each sample on average.

3.2.3 Partial Supervision. Given labeled entity links, quantity links and calculation types (inferred from the annotated formula), we further explore to guide training in a partially supervised way. These three annotations instantiate as selected headers, region and operators in QA. For NSM, we exploit them to prune spurious programs, i.e. incorrect programs that accidentally produce correct answers, in two ways. (1) In searching consistent programs, besides producing correct answers, programs are required to satisfy these three conditions. If no program is found, we slack the constraint to satisfying two conditions. In this way, the average number of consistent programs reduces from 6.12 to 2.13 per sample. (2) In training, we modify the binary reward function: satisfying each condition will add 0.2 to total reward. The sampled programs with reward \( r \geq 1.4 \) are added to the program buffer.

For TaPas, we additionally provide answer coordinates and calculation types in training following its WikiSQL setting.

3.2.4 Evaluation Metrics. We use Execution Accuracy as our metric following [38, 53], which measures the percentage of samples that the method produces correct answers. We also report Spurious Program Rate to study the percentage that the method generates correct answers with false logical forms. Since we do not have golden logical forms, we manually annotate our logical forms for 150 random samples in dev set for evaluation.

3.2.5 Implementations. We split 3,597 tables into train (70%), dev (15%) and test (15%). We download pre-trained models from huggingface\(^2\) library. In training, we use Adam optimizer with learning rate \( 5e^{-5} \). For NSM, we utilize bert-base-uncased to initialize encoder, and fine-tune 20K steps on HiTab. Beam size is 5 for both training and inference. To test with MAPO original logical form, we transform tables to flat ones just like what we do in TaPas. For TaPas, we adopt the PyTorch version provided by huggingface. We utilize tapas-base as initialization, and fine-tune 40 epochs on HiTab. All experiments are run on four V100 GPUs.

3.3 Results

Table 4 summarizes our evaluation results.

Weak Supervision First, MAPO with our hierarchy-aware logical form largely outperforms that using its original logical form by 11.5%, indicating the necessity of designing a logical form leveraging hierarchies. Second, MAPO achieves the best execution accuracy (40.7%) with the lowest spurious program rate (19%), but still more than half of questions can not be answered correctly, which proves QA on HiTab is challenging. Third, though TaPas benefits from pretraining on tables, it performs worse than the best logical form-based method without table pretraining.

Partial Supervision From Table 4, we can conclude the effectiveness of partial supervision in two aspects. First, it improves execution accuracy. The model learns how to deal with more cases given high-quality programs. Second, it largely lowers spurious rate. The model learns to generate correct programs instead of some tricks. MML, whose performance highly depends on the quality of searched programs, benefits the most (36.7% to 45.1%), indicating partial supervision improves the quality of consistent programs by pruning spurious ones. However, TaPas does not gain much improvements from partial supervision, which we will discuss in error analysis.

Error Analysis For TaPas, 98.7% of success cases are cell selections, which means TaPas benefits little from partial supervision. This may be caused by: (1) TaPas does not support some common operators on hierarchical table like difference; (2) the coarse-to-fine cell selection strategy first selects columns then cells, but cells in different columns may also aggregate in hierarchical tables.

For MAPO under partial supervision, we select 100 error cases and analyze them manually. We divide error cases into four categories: (1) entity missing (23%); the header to filter is not mentioned in question, where a common case is omitted; (2) model failure: this includes (2) failing to select correct regions (38%) and (3) failing to generate correct operations (20%); (4) out of coverage (19%); question types can not be handled by logical form, which is explained in Appendix A.1. Spurious programs occur mostly in two patterns. In cell selection, there may exist multiple data cells with correct answers (e.g. G9,G16 in Figure 1), while only one is golden. In superlatives, the model can

\(^2\)https://huggingface.co/transformers/
produce the target answer by operating on different regions (e.g. in both B21:B25 and B23:B25, B23 is the largest).

4 HIERARCHICAL TABLE TO TEXT

4.1 Problem Statement

The dataset \( H = \{T_i, S_i\}, i \in [1, N]\) is a set of \( N\) table-description instances. Description \( S_i\) is a sentence about a hierarchical table \( T_i\). \( S_i\) should be fully supported by the content of \( T_i\), and can be described in greater detail by a series of operations \( O_i = [O_{i1}, O_{i2}, \ldots, O_{im}]\) on certain table cells \( C_i = [c_{i1}, c_{i2}, \ldots, c_{im}]\). We now define the task of Hierarchical-Table-to-Text as: given a hierarchical table \( T\), one needs to generate a description \( S\), with controls on cells \( C\) and operators \( O\).

Full tables often contain quite general information. Some works frame table-to-text as a summarization problem. However, its subjectivity often renders the task unconstrained and the evaluation difficult. To accurately state facts or perform operations based on user intents, extra guidance from target cells and operators can be of great help. We place our task at a controlled setting, where models are provided with certain guidance at generation.

Besides the unique hierarchical table structure and meaningful texts, our task distinguishes for it owns valuable annotations of entities and quantities. They can enable more detailed and diversified attempts on table NLG.

4.2 Controlled Generation

Full tables have sufficient yet general contents. Often by highlighting table cells [37] and specifying the calculation process [19], models produce more specific and logical generations. Highlighted cells can point out the informative cells and exclude irrelevant ones. Operators clarify numerical intents and reduce factual ambiguity, pushing generations beyond simple data record statements. For accurate generations towards specific user intents, we experiment with two controlled settings: 1) with cells of interest, and 2) further with the operators that indicate the calculation process on cells.

4.2.1 With Highlighted Cells. An entity or quantity in text can be supported by cells if it is directly stated in cell contents, or can be logically inferred by them. Motivated by [37], cell highlights help models to produce more specific generations. Different from only taking data cells as highlighted cells [37], we additionally support highlighted cells in header regions as conditions, and it is usually the case for superlative ARG-type operations on a specific header level in hierarchical tables. In our training and testing phases, highlighted cells are extracted from annotations of the entity and quantity alignment, then use the extracted table hierarchy to group the selected cells into the top header, left header, and data region. (2) based on the extracted table hierarchy, we use the source set of top and left header cells to include corresponding data cells, and we also use the source set of data cells to include corresponding header cells. (3) we leverage the table hierarchy to include their parent header cells to construct a full set of headers. In the end, we take the union of them as the result of sub table selection.

4.2.2 With Operators that Indicate the Calculation Process. Highlighted cells can tell the target for text generation, but is not sufficient. Some works use logical forms [9] or mathematical expressions [19] to ground quantities with their calculation process. It motivates us to use formulas as additional controls for text generation. Different from logicNLG [8], where logical forms are hard to write by users without the computer science background, we propose to use operators as conditions that are very easy to apply by users.

This extra control contributes to text clarity and meaningfulness in two ways. 1) It clarifies the numerical reasoning intent on cells. For example, given the same set of data cells, applying SUM or COUNT conveys different meanings thus should yield different texts. 2) Operation results on highlighted cells are additional input sources. Nowadays, seq2seq language models are not good at doing arithmetic operations, e.g., calculating the average of a group of numbers, and it greatly limits their ability to generate correct numerical values in sentences. Explicitly pre-computing calculation results is a promising way to mitigate this gap in seq2seq models.

Even with these controls, text generation on hierarchical tables is still a challenge due to the complex hierarchical indexing and implicit semantic relationships among cells.

4.2.3 Sub Table Selection and Input Serialization.

Sub Table Selection Under controls of selected cells and operators, we devise a heuristic to retrieve all contextual cells as a sub table. (1) we start with highlighted cells extracted from our entity and quantity alignment, then use the extracted table hierarchy to group the selected cells into the top header, left header, and data region. (2) based on the extracted table hierarchy, we use the source set of top and left header cells to include corresponding data cells, and we also use the source set of data cells to include corresponding header cells. (3) we leverage the table hierarchy to include their parent header cells to construct a full set of headers. In the end, we take the union of them as the result of sub table selection.

Serialization On each controlled table, we do a row-turn traversal on linked cells and concatenate their cell strings using [SEP] tokens. Operator tokens and calculation results are also concatenated with the input sequence when conditioning on operators. We also experiment with other serialization methods, such as header-data pairing or template-based method, yet none reported superiority over the simple concatenation that we end up with.

4.3 Experiments

4.3.1 Baseline. We present baseline results on HiTab by examining three representative methods on text generation.

Pointer Generator [43] A LSTM-based seq2seq model with copy mechanism. The model uses two-layer bi-directional LSTMs for the encoder and 300-dim word embeddings, 300 hidden units. We perform fine-tuning using batch size 2 and learning rate 0.05.

BERT-to-BERT [42] A transformer encoder-decoder model [47] where the encoder and decoder are both initialized with BERT [11] by loading the checkpoint named ‘bert-base-uncased’ provided by the huggingface/transformers repository. We perform fine-tuning using batch-size 2 and learning rate 3e-5.

BART [26] BART is a pre-trained denoising autoencoder for seq2seq language modeling. It uses standard Transformer-based architecture and shows effectiveness in NLG. We align model configuration with the BASE version of BART, and use the model ‘facebook/bart-base’ in huggingface/transformers. During fine-tuning, we use a batch size of 8 and a learning rate of 2e-4.

T5 [41] T5 is also a transformer-based pre-training LM. It trains extensively on text-to-text tasks and scores high on generation tasks. We use the pre-trained model ‘t5-base’ in huggingface/transformers. For fine-tuning, we set batch size to 8 and learning rate to 2e-4.

4.3.2 Evaluation Metrics. We use two automatic evaluation metrics, BLEU and PARENT, to evaluate text generations. The BLEU
metrics [36] is broadly used for evaluations of text generation. All experiments report the most common BLEU-4 by default. Besides, PARENT [12] is a metric proposed specifically for data-to-text evaluation that takes the table into account. It additionally aligns n-grams from the reference and generated texts to the structured table.

### 4.3.3 Experiment Setup.

Samples are randomly split into train (70%), validation (15%), and test (15%) sets. To ensure generalization difficulty, tables have no overlap between splits, i.e., samples of a table always appear in the same split. Unless otherwise stated, we allow inputs of at most 512 tokens per instance and use a beam size of 5 to search decoded outputs from 60 to 60 tokens.

### 4.3.4 Result and Analysis.

First, from an overall point of view, both metrics report relatively low scores. This well proves the difficulty of HiTab. It could be from the complex table hierarchy, as well as statements with logical and numerical complexity.

Second, results across models are quite consistent. Replacing the traditional LSTM with Attention module shows increases of +5.6 in BLEU and +7.9 in PARENT. Leveraging seq2seq-like training further yields a rise of +6.5 BLEU and +11.3 PARENT. Lastly, between seq2seq-trained Transformers, T5 reports higher scores over BART, probably for T5 is more extensively tuned during pre-training.

Third, by comparing two controlled scenarios, we see that: augmenting quantity cells with calculation process using formula greatly helps, in both metrics and with all models. So, to produce texts in specific intents, the more controlled input is, the more meaningful a generated sentence can be.

Further, to study the generation difficulty concerning table hierarchy, we respectively evaluate samples at different hierarchical depth, i.e. table’s maximum depths in top and left header trees. In groups of 2, 3, 4+ depth, BLEU scores 31.7, 26.5, 21.3 and PARENT scores 40.9, 36.5, 31.6. As table headers grow deeper, they often involve more complex hierarchies, making it harder for data indexing, cell relationship discrimination, and more.

### 5 RELATED WORK

#### Table-to-Text

Existing datasets for table-to-text are restricted in flat tables or specific subjects [1, 7, 24, 27, 31, 34, 51]. The most related table-to-text dataset to HiTab is ToTTo [37], in which complex tables are also included. There are two main differences between HiTab and ToTTo: (1) hierarchical tables in ToTTo only account for a small proportion (5%); (2) there are no indication and usage of table hierarchies in ToTTo. In contrast, hierarchies are explicitly extracted and studied for public usage in HiTab.

Table QA focuses on relational DB tables [48, 53, 54] and semi-structured tables [38, 46], while hierarchical tables are common but not involved. There exist two popular methodologies, logical form-based semantic parsing [28, 29, 52], and end-to-end parsing without logical form [18]. Recently, SLSQL [25] and SQUALL [44] prove that schema linking is important to table QA, motivating us to annotate fine-grained entity and quantity alignments.

Table structure understanding involves a series of tasks: table detection [14], table recognition [15, 33], hierarchy extraction [6, 49], cell classification [13, 16, 40], etc. By stringing them together, [5, 22, 23] explored extracting relational data from semi-structured tables, but need human interactions to get precise results.

### 6 DISCUSSION

HiTab also presents cross-domain and complicated-calculation challenges. (1) To explore cross-domain generalizability, we randomly split train/dev/test by domains for three times and present the average results of our best methods in Table 6. We found decreases in all metrics in QA and NLG. (2) Figure 3 shows a bad case that challenges existing methods due to the complicated calculations. Performing complicated calculations needs to jointly consider quantity relationships, header semantics, and hierarchies.

### 7 CONCLUSION

We present a new dataset, HiTab, that simultaneously supports QA and NLG on hierarchical tables. Importantly, we provide fine-grained annotations both on entity and quantity alignment. Experiment results suggest that HiTab can serve as a useful and challenging benchmark for question-answering and table-to-text on hierarchical tables.
A HIERARCHICAL TABLE QA

A.1 Logical Form Functions

We list our logical form functions in Table 8.

Union selection is required for comparative and arithmetic operations. It is achieved by allowing variable number of headers in \textit{filter\_tree}, where “variable” is one or two in practice.

In our implementation, a function by default takes the selected region of last function as input region \(R\) to prune search space. Thus argument \(R\) is omitted in main part of the paper for brevity. And we deactivate order relation functions (e.g. \textit{eq} function) and the order argument \(k\) in \textit{argmax/argmin} because there are few questions in these types and activating them will largely increase number of spurious programs when searching.

The logical form coverage after deactivation is 78.3\% in 300 iterations of random exploration. Some typical question types that can not be covered are: (1) scale conversion, e.g. 0.984 to 98.4\%, (2) operating data indexed by different levels of headers, e.g. proportion of total, (3) complex composite operations, e.g. Figure 3.

A.2 Table Linearization

We linearize the question and table according to Figure 4.

The input is concatenation of question and table. Table is linearized by putting headers in level order. Each level is led by a \textit{[LEVEL]} token to gather current level embedding. The first \textit{[LEVEL]} token stands for level zero of left. Each header is linearized as \textit{name | type}. \textit{name} is the tokenized header string. \textit{type} is the entity type parsed by Stanford CoreNLP, which includes “string”, “number”, “datetime” in our case. Headers with the same \textit{name} will gather token embeddings by mean pooling.

A.3 Examples of Logical Form Execution

Take the table in Figure 4 as input table, we demonstrate three types of questions with complete logical forms in Table 7.

| Question                                                                 | Logical Forms |
|--------------------------------------------------------------------------|---------------|
| **Cell Selection**                                                       |               |
| Q: What is the GDP of China in 2012?                                    | \begin{itemize} \item \textit{filter\_tree} 2012 \item \textit{filter\_tree} china \item \textit{filter\_level} LEFT\_1 \item \textit{filter\_tree} gdp \item \textit{filter\_level} TOP\_1 \end{itemize} |
| **Superlative**                                                          |               |
| Q: Which country has the highest GDP in 2012?                           | \begin{itemize} \item \textit{filter\_tree} 2012 \item \textit{filter\_level} LEFT\_2 \item \textit{filter\_tree} gdp \item \textit{filter\_level} TOP\_1 \item \textit{argmax} 1 \end{itemize} |
| **Arithmetic**                                                           |               |
| Q: How much more is U.S. GDP higher than China in 2013?                  | \begin{itemize} \item \textit{filter\_tree} 2013 \item \textit{filter\_tree} U.S. China \item \textit{filter\_level} LEFT\_2 \item \textit{filter\_tree} GDP \item \textit{filter\_level} TOP\_1 \item \textit{difference} \end{itemize} |

Table 7: Examples of our logical form. Argument \(R\) is omitted since by default a function takes the return region of last function as input. \textit{LEFT\_1} is a symbol for the first level on the left.
Table 8: Logical Form Function List

Function | Arguments | Returns | Description
--- | --- | --- | ---
(filter_tree R h) | R: a region; h: a header | a region | Select a region indexed by sub-tree of the given header in the given region.
(filter_level R l) | R: a region; l: a level | a region | Select a region indexed by headers on the given level in the given region.
(argmax R k) | R: a region; k: a number | a list of headers | Find the header(s) with k-th largest/smallest value in the region. [Input region should have one row/column of data]
(min R l) | R: a region; l: a level | a region | Maximum/minimum/sum/average of the given region, group by the given level of headers, i.e. data values aggregate according to their header strings on the given level.
(count R l) | R: a region; l: a level | a number | Count number of headers on the given level of the given region.
(difference R) | R: a region | a number | Absolute difference, proportion and difference rate of given two elements a and b in region. rev means changing order of operands. e.g. proportion applies b/a and proportion_rev applies a/b. [Input region should have two data elements]
(greater_than R n) | R: a region; n: a number | a list of headers | Find the header(s) with data value that have certain order relation with given value. [Input region should have one row/column of data]
(greater_eq_than R n) | R: a region; n: a number | a list of headers | Take opposite value of data in given region. [Input region should have one data element]

Example

Q: What is the GDP of China in 2012?

A: 8229

Model

Figure 4: An example table with hierarchy and its linearized input to the encoder. LEFT_0 means the 0 level on the left tree.

REFERENCES

[1] Eva Banik, Claire Gardent, and Eric Kow. The kbgen challenge. In the 14th European Workshop on Natural Language Generation (ENLG), pages 94–97, 2013.

[2] BLS. U.s. bureau of labor statistics. https://www.bls.gov Accessed July 4, 2021.
