TaCube: Pre-computing Data Cubes for Answering Numerical-Reasoning Questions over Tabular Data

Fan Zhou[1]*, **Mengkang Hu**[2], **Haoyu Dong**[3]†, **Zhoujun Cheng**[1], **Fan Cheng**[1], **Shi Han**[3], **Dongmei Zhang**[3]

[1] Shanghai Jiao Tong University, [2]Harbin Institute of Technology [3]Microsoft Research

{zhoufan98,blankcheng,chengfan}@sjtu.edu.cn, 1190200505@stu.hit.edu.cn,

{hadong, shihan, dongmeiz}@microsoft.com

Abstract

Existing auto-regressive pre-trained language models (PLMs) like T5 and BART, have been well applied to table question answering by UnifiedSKG and TaPEx, respectively, and demonstrated state-of-the-art results on multiple benchmarks. However, auto-regressive PLMs are challenged by recent emerging numerical reasoning datasets, such as TAT-QA, due to the error-prone implicit calculation. In this paper, we present TaCube, to pre-compute aggregation/arithmetic results for the table in advance, so that they are handy and readily available for PLMs to answer numerical reasoning questions. TaCube systematically and comprehensively covers a collection of computational operations over table segments. By simply concatenating TaCube to the input sequence of PLMs, it shows significant experimental effectiveness. TaCube promotes the F1 score from 49.6% to 66.2% on TAT-QA and achieves new state-of-the-art results on WikiTQ (59.6% denotation accuracy). TaCube 's improvements on numerical reasoning cases are even more notable: on TAT-QA, TaCube promotes the exact match accuracy of BART-large by 39.6% on `sum`, 52.5% on `average`, 36.6% on `subtraction` and 22.2% on `division`. We believe that TaCube is a general and portable pre-computation solution that can be potentially integrated into various numerical reasoning frameworks. Data and code will be available at https://github.com/microsoft/TaCube.

1 Introduction

There are numerous recent works on table-text joint reasoning, e.g., answering questions over tables (Yu et al., 2018; Pasupat and Liang, 2015; Zhong et al., 2017). Meanwhile, pre-trained language models (PLMs), which have demonstrated great success on various natural language (NL) tasks, are also well applied to table-text joint reasoning and show great effectiveness (Yin et al., 2020; Herzig et al., 2020; Cheng et al., 2021a; Liu et al., 2021; Xie et al., 2022; Dong et al., 2022).

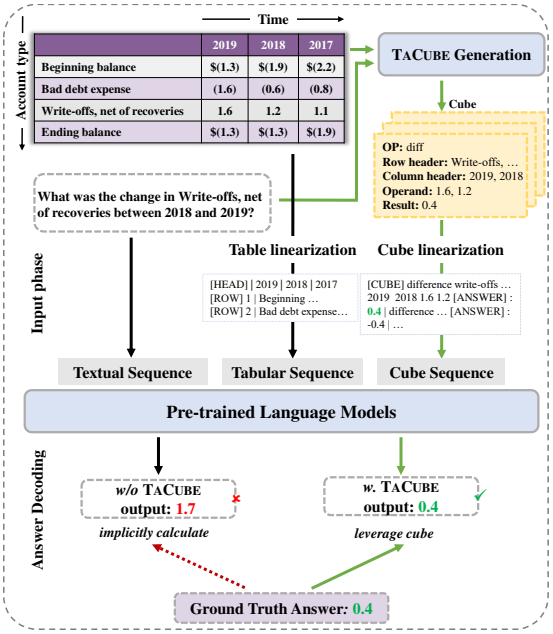

Figure 1: Augmenting auto-regressive PLMs with TaCube. By simply concatenating TaCube to the input sequence, TaCube significantly mitigates the calculation challenge in numerical reasoning over tables.

Recently, numerical reasoning (NR) over tabular data has raised increasing attention and a bunch of table QA datasets targeting numerical reasoning have been collected (Chen et al., 2021b; Zhu et al., 2021; Zhao et al., 2022b; Cheng et al., 2021b), promoting NR to be a hot challenge in table QA. However, faced with flexible calculations such as addition, comparison and aggregation over semi-structured context, PLMs encounter great obstacles (Liu et al., 2021; Zhu et al., 2021), especially when calculation skills have not been fully exploited during large-scale pre-training on NL corpora.

Existing approaches to mitigate this gap can be mainly concluded into two families. One is logical-

*Work done during internship of Fan and Mengkang at Microsoft Research Asia.

†Corresponding author.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 2278 - 2291
December 7-11, 2022 ©2022 Association for Computational Linguistics

form-based method, which formulates table QA as a semantic parsing task. The method first generates the logical form and then applies a post-execution on the logical form to produce the final answer. The answer generation process is much more trackable and explainable than directly decoding answers, but it also has deficiencies. (1) Human annotations of logical forms are expensive and error-prone (Herzig et al., 2020; Cheng et al., 2021a). (2) There lacks a dominant formulation of logical forms for table reasoning: (Pasupat and Liang, 2015; Liang et al., 2018; Guo et al., 2019; Gao et al., 2019; Cheng et al., 2021b) design their own logical forms and semantic parsers; (Herzig et al., 2020; Zhu et al., 2021) simplify the logical form design and use specific classification heads for cell selection, operator prediction and unit prediction, to produce the answer. (3) Due to the representation limitation of logical forms, it lacks sufficient flexibility to generate free-form answers or answers matching special conditions, e.g., "2 million" that augments a numeric value "2" with its textual scale "million" and "2022" that is a part of a cell string "2022/05/20".

The other popular way is directly generating the answer using auto-regressive PLMs. UNIFIEDSKG (Xie et al., 2022) leverages T5 (Raffel et al., 2020) and achieves promising and even state-of-the-art performance on a series of datasets, showing the power of directly fine-tuning large LMs on table-text joint reasoning. TAPEX (Liu et al., 2021), which is based on BART (Lewis et al., 2020) and pre-trained by learning a neural SQL executor, surprisingly outperforms prior works by a large margin. TAPEX provides a new way of using clean and synthetic data for efficient pre-training and shows its promising capacity on several numerical reasoning types, e.g., `count` and `comparison`. However, when it comes to numerical calculations involving `sum`, `subtraction`, or `division`, the accuracy falls to the bottom, showing that implicit numerical calculation is brittle and error-prone. To make things worse, when auto-regressive PLMs produce a wrong number, e.g., "1.7" in Figure 1, it's hard to uncover where the number comes from and improve the model accordingly, because all calculations are implicitly done in transformers.

To address the above shortcomings, we propose to pre-compute aggregation/arithmetic results for the target table in advance, so that they are handy and readily available for answering numerical reasoning questions. This idea is inspired by an important data cube (Gray et al., 1997; Han et al., 2011) concept in the data mining field to facilitate the online analytical processing of multi-dimensional data, so we name our pre-computed aggregation/arithmetic results as TACUBE. TACUBE systematically covers a collection of computational operations over table segments in a comprehensive way. TACUBE can not only be fully materialized, but also be partially materialized for efficiently application to existing models. We propose rule-based and neural-based methods to produce efficient TACUBE while maintaining high coverage on ground truth numerical reasoning. In this paper, we applied TACUBE to auto-regressive PLMs, for their flexibility in decoding answers by leveraging both the original table and TACUBE (as fast access) to avoid most error-prone implicit calculations. We believe that TACUBE is a general pre-computation solution that can be helpful to various numerical reasoning frameworks.

Our experiments show that, by directly augmenting the original table with sequenced TACUBE: (1) on TAT-QA, TACUBE significantly improves BART-large by 18.3% in F1 score and TAPEX by 16.6% in F1 score, and outperforms the logical-form-based state-of-the-art method TagOP by 3.5% in F1 score; (2) on WikiTQ, TACUBE also achieves new state-of-the-art denotation accuracy of 59.6% (+2.1%). In addition to the overall improvements, we analyze TACUBE's EM improvements by different calculation operators on TAT-QA based on BART-large: `sum` increased by 39.6%, `average` increased by 52.5%, `subtraction` increased by 36.6%, and `division` increased by 22.2%, and comparable improvements are also found in TAPEX.

2 Preliminary

2.1 Data Cube As Pre-computation

The concept of data cube is proposed in data mining, which can provide fast access to pre-computed, summarized data and thus benefit the analysis process (Gray et al., 1997; Halevy, 2001; Han et al., 2011). A data cube, which is defined by dimensions and facts, allows stored data records to be modeled and viewed in multiple dimensions. Each dimension corresponds to an attribute or a set of attributes, usually the perspectives or entities with respect to which an organization wants to keep records. Each fact is a numeric measure, organized by fact tables containing the name of the fact, usually the value

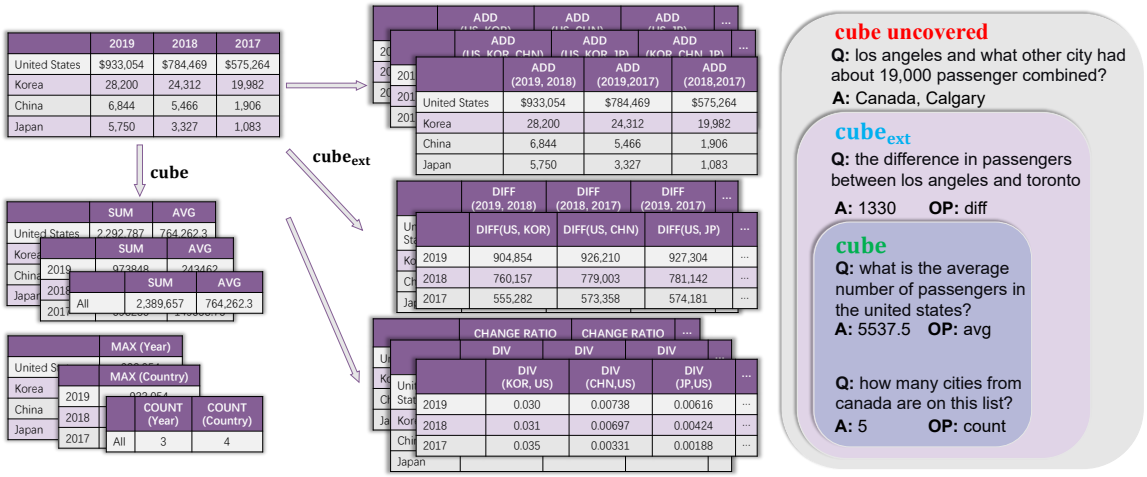

Figure 2: Illustration of the relationship between `cube`, $\texttt{cube}_{\texttt{ext}}$ and cube-uncovered cases.

of some aggregate measure such as `count` and `sum`. Data cube inherently supports operations including drill-down and roll-up, which allow the user to view the data at differing degrees. Also, the roll-up operation can be used to perform controllable data summarization along specified dimension(s).

Semi-structured tables also share part of the features of stored records to generate a data cube. The numeric values stored in table cells can be treated as the stored records, while the header names or textual cell values can be treated as different dimensions corresponding to the records. Take the cross table in Figure 1 for example. Numeric values in the table represent the annual accounts of one corporation. The top row serves as the column header, and the first column serves as the row header. Thus, there are two dimensions for the data cube: time dimension and account type dimension.

The data cube naturally reflects some of the factual numerical statistics of a given table, and can potentially help tackle numerical reasoning problems over tabular data as a pre-computation.

2.2 Datasets

TAT-QA dataset (Zhu et al., 2021) includes 16,552 samples with hybrid context of both tables and textual content from real-world financial reports. Over 40%(7,341 of 16,552) of the data requires numerical-reasoning skills including addition, subtraction, multiplication, division, count, comparison, and compositional operations. TAT-QA combines the correct scale together with the string or numerical value to make up the final answer, which is a unique challenge compared to other datasets.

WikiTQ (Pasupat and Liang, 2015) includes 22,033 samples focusing on table-only QA. WikiTQ does not have annotations for numerical reasoning cases. The author randomly picked 200 examples and classified them based on the types of operations required to answer the question. The case study reflects that WikiTQ contains considerable cases of complicated reasoning. The **Squall** dataset (Shi et al., 2020) manually annotates SQL query on approximately 80% of WikiTQ cases, transferring the unsupervised task into a supervised semantic parsing task. According to Squall, WikiTQ contains multiple types of operations, e.g., `count`, `sum`, `max`, `min`, `group`, `abs` and `avg`.

Apart from the above two datasets studied in this paper, numerous tabular datasets are also closely related to numerical reasoning. Fin-QA (Chen et al., 2021b) collects tables and context from financial domain, and raises financial-related questions including complex numerical reasoning. MultiHiertt (Zhao et al., 2022b) and HiTAB (Cheng et al., 2021b) include numerical reasoning examples on hierarchical table question answering. For table fact verification tasks (Chen et al., 2019; Aly et al., 2021), the numerical reasoning skills are also required for verifying whether the given statement about the table is entailed or refuted.

However, the performance of existing methods tackling such tasks is still far from human performance, which indicates the numerical reasoning problems as an emerging challenge in tabular data.

2.3 PLMs for Table QA

PLMs have been widely leveraged in dealing with table QA tasks. Recent works also prove that powerful PLMs with decoders can be well applied to the table-text joint representation. On a well-recognized table QA benchmark WikiTQ, UNI-

FIEDSKG (Xie et al., 2022) achieves state-of-the-art denotation accuracy of 49.3% among methods without extra pre-training; TAPEX (Liu et al., 2021) with pre-training on SQL corpus surpasses the previous methods by a large margin, obtaining denotation accuracy of 57.5%.

Represented as a form of pre-computed data cube, TACUBE can be easily integrated at the input phase and is theoretically suitable for most of the PLMs. Due to the shortcomings discussed in Section 1, in this paper, we mainly choose PLMs with decoders and apply the generator-based method in follow-up experiments over table QA tasks. The generator-based method can effectively leverage the weakly-supervised data, and flexibly organized the cell value to generate the answer accordingly, e.g., to produce a piece of text inside a cell, or to produce a numeric value with its unit.

Model Input/Output For semi-structured tables, TAPEX (Liu et al., 2021) designs a table linearization to feed the flattened sequence into the model directly. A flattened sequence is denoted as

$$\begin{aligned} T^* = & \texttt{[HEAD]} \mid c_1 \mid \cdots \mid c_N \\ & \texttt{[ROW]}\ 1 \mid r_1 \mid \texttt{[ROW]}\ 2 \mid r_2 \mid \cdots \mid r_M \end{aligned} \quad (1)$$

Notably, TAPEX uses a vertical bar "|" to separate headers or cells in different columns. The final input will concatenate the textual context and the table sequence, and will then be fed into the model. The expected output of TAPEX is the concatenation of answer(s) separated by a comma which is generated autoregressively by the model. It is also simple to extend to more complex answer forms, e.g., using "2 | million" to represent the answers with scale in TAT-QA. TAPEX proved to be effective in finding the factual supporting cells.

We take the above model input/output setting. The pre-computed TACUBE will also be flattened into a sequence, which is discussed in Section 3.2.

3 TACUBE

In this section, we first discuss how pre-computed data cubes help answer numerical reasoning questions over tables. Secondly, we point out that plain implementation via brute force is not favored due to explosive increasing cube size and time consumption. Conventional cube techniques such as iceberg cube and cube shell (Han et al., 2011) are mainly designed for data mining and data analysis and are not applicable to the current QA tasks on numerical reasoning. Thus, we need to design new efficient methods to generate cubes.

Following such observation, we present our approach for cube generation and cube candidate selection. First, we introduce **cube**, which only contains aggregation operators. Second, we further propose $\textbf{cube}_{\texttt{ext}}$, which extends the operators to non-aggregation operators. Furthermore, based on $\texttt{cube}_{\texttt{ext}}$, we develop $\textbf{cube}_{\textbf{k}}$ to reduce the cube size when feeding to the PLMs. We adopt two ranking methods to perform effective filtering to pick out the most likely candidate cube items. The selected cube items of a cube make up $\texttt{cube}_{\texttt{k}}$ which will be finally fed to the model.

We will show that the generated cube can cover a large portion of numerical reasoning cases and remains efficient in search space at the same time.

3.1 Data Cube Application in Table QA

Following the definition of the cube, to apply data cube in table QA, we need to decide on dimension and arithmetic/aggregation operator types. Semi-structure tables contain rich information not only in cell values but also in structural arrangement through headers or hierarchy (Dong et al., 2022; Wang et al., 2021). For example, cells under the same header, or cells in the same table row are most likely to represent the same type of information.

Thus, a data cube for the table can be sets of aggregations over cells under the same **column headers** or **row headers**. We denote such aggregation results over table headers as **cube**, which include operations such as `sum`, `count` and `average`.

Nevertheless, such pre-computed results can't cover answers requiring non-aggregation operations, e.g., question asking the difference(`diff`) of two cell values or asking the summation result(`add`) of certain cells with specific filtering conditions. In fact, the proportion of non-aggregation operations is even higher in some of the numerical reasoning datasets (Zhu et al., 2021; Zhao et al., 2022b; Chen et al., 2021b). Based on such observation, We extend the operation types for pre-computed results, making it cover more operation types.

We denote the total extended pre-computed results(including **cube**) as $\textbf{cube}_{\texttt{ext}}$. The newly introduced operators contain non-aggregation computation, such as `add`, `subtraction`, `division` and `change ratio`. Notably, `add` and `sum` are in different groups of operators. We use `sum` to represent an aggregation operator which sums up all the numeric values under one or many dimensions, and use `add` to represent an extended operator which

Operator	aggr / ext	Calculation
count	aggr	$\lvert\text{cell}_{\text{selected}}\rvert$
sum	aggr	$\sum \text{cell}_{\text{selected}}$
average	aggr	$1/\lvert\text{cell}_{\text{selected}}\rvert \cdot \sum \text{cell}_{\text{selected}}$
add	ext	$\text{cell}_1 + \cdots + \text{cell}_k$
diff	ext	$\text{cell}_1 - \text{cell}_2$
div	ext	$\text{cell}_1/\text{cell}_2$
change ratio	ext	$(\text{cell}_1 - \text{cell}_2)/\text{cell}_2$

(a) Operators for cube generation

Pattern	Selected operand cells
Same column	cells in one candidate column and all candidate rows
Same row	cells in one candidate row and all candidate columns
All row	all cells in one candidate column and all rows
All column	all cells in one candidate row and all columns
Top-k row	top-k rows' cells in one candidate column

(b) Computing patterns for cube generation

Table 1: Operator types and computing patterns to generate $\texttt{cube}_{\texttt{ext}}$. We divide operators into two groups: **aggr** stands for aggregation (types) and **ext** stands for extended (types).

adds up candidate numeric values after filtering. As Figure 2 shows, in this paper, we only extend to **first-order** cube, i.e., we do not further perform an operation over the generated cube items, which may include more compositional computation cases. We also conclude some **computing pattern** templates, e.g., `same row` pattern which indicates the operands are all in the same row. Such templates further shrink the generation space. The extended operator types and the computing patterns are listed in Table 1. The computing patterns are concluded based on observations of numerical reasoning cases and can help select operands for pre-computation in a restricted manner.

We will show that $\texttt{cube}_{\texttt{ext}}$ can cover a significant portion of the numerical reasoning cases in the following sections.

3.2 Cube Generation

We aim to provide pre-computed results in the model input phase so that the model can generate answers based on table cells as well as augmented cube information, bridging the gap in numerical reasoning. Also, to reduce the cube size, the cube generation is question-sensitive, i.e., for a given question and table pair, we generate the corresponding cube where cube items are most related to the question. For future work, we may try question-insensitive cube generation if the length of the cube is not a burden to the sequence length.

Brute Force Generation It is straightforward to use the brute force method to traverse the whole search space and generate all the candidate results for the $\texttt{cube}_{\texttt{ext}}$ of a given table. If the operation type and computing pattern to generate the expected answer are included in Table 1, then the correct answer must exist in the generated $\texttt{cube}_{\texttt{ext}}$.

But brute force leads to two problems in the generation phase and model training phase. First, the explosive time consumption and space consumption for a large table. Suppose a table with m rows and n columns is given and we understand little about its structure, the theoretical search space will be up to $O(n \cdot 2^m + m \cdot 2^n)$ for an aggregation operation, and $O(n \cdot m^2 + m \cdot n^2)$ for a 2-operand operation. Even if we rigorously follow the definition to generate the cube, we can still see from the Figure 2 that the cube size will easily exceed the size of the table itself. Note this is just the situation where the simple matrix table contains only two dimensions(time dimension and country dimension), five rows, and four columns. For a multi-dimension table with much more columns and rows, the time consumption and the search space will be intolerably large. Secondly, too many candidates can be a burden on the model input length and make it difficult for the model to leverage.

Rule-based Generation Due to the high expense and potential problems of brute force generation, it is worth exploring the possibility of efficiently generating pre-computed results for the Question-Table pair. We want the cube to be streamlined so that it can save the input sequence's length. Meanwhile, it should cover most of the common numerical reasoning cases in the tasks.

First, We restrict the cube to be computed using **same-row** or **same-column** operands following Table 1b. Secondly, we require the generated cube to be question-sensitive, i.e., to generate specific cube items closely related to the question. For instance, given a question answering "what is the difference in passengers between Los Angles and Toronto", the cube should better not to conclude items with irrelevant operators, such as `count` and `sum`. We decide the candidate operators, headers, and cells based on an aligned mention in the question. We conclude the template for computing patterns of our rule-based generation in Table 1b. The template for computing patterns also reduces the search space of cube generation.

The rule-based generation process can be concluded in three steps in general: (i) Decide operator type(s) by textual mention in questions. (ii) Find

candidate columns and candidate rows by matching column headers, row headers, and cell values with aligned mention in questions. (iii) Try all the combinations using (1) operator; (2) k_{row} candidate rows and k_{column} candidate columns; (3) computing patterns. Each computation result serves an item in the cube.

Such heuristic methods significantly shrink the time complexity to polynomial while sacrificing the coverage within a tolerable range. If a pre-computed cube for a given question and table pair contains one item that generates the correct numeric value to the answer, we treat it as a correct cube. The coverage is the correct cubes' proportion of total extracted cubes. (i) On the TAT-QA dev set, we achieve coverage of approximately 70% over all cases involving aggregation/arithmetic operation; (ii) On WikiTQ, we achieve coverage of 68% over all arithmetic-involved cases on the dev set and 62% on the train set.

Cube Serialization The pre-computed and ranked cube will be flattened as a sequence in the end and this leads to designing cube serialization. As shown in Figure 1, we present one item in the cube with its operator, the row header, the column header, the selected cell value, and the pre-computed answer. We design a naive cube linearization similar to the table linearization. And a flattened cube sequence is denoted as C^*.

$$\begin{aligned} C^* = & \texttt{[CUBE]}, \texttt{OPERATOR}, \\ & \texttt{CH}_1, \cdots, \texttt{CH}_{kc}, \texttt{RH}_1, \cdots, \texttt{RH}_{kr}, \\ & \texttt{op}_1, \texttt{op}_2, \cdots, \texttt{op}_m, \texttt{[ANSWER]} : \texttt{answer} \end{aligned} \tag{2}$$

Here `[CUBE]` and `[ANSWER]` are special tokens indicating the start of the TACUBE and the answer; `CH`, `RH` and `op` stands for **C**olumn **H**eader, **R**ow **H**eader and **Op**erand; kc, kr are numbers of the headers of all the operands.

Notably, in Table 1 we concluded all the computing patterns for a pre-computed result. So all the operands in one TACUBE must share either the same column headers or same row headers.

3.3 Cube Ranking

Although the extracted pre-computed cube results' size is within a polynomial level, it still can be non-trivial and causes the same dilemma as the brute force generation. To overcome the problem, we propose two methods for ranking the cube results, including a heuristic method and a neural-based method. We denote the top k picked pre-computed cube items as $\textbf{cube}_\textbf{k}$.

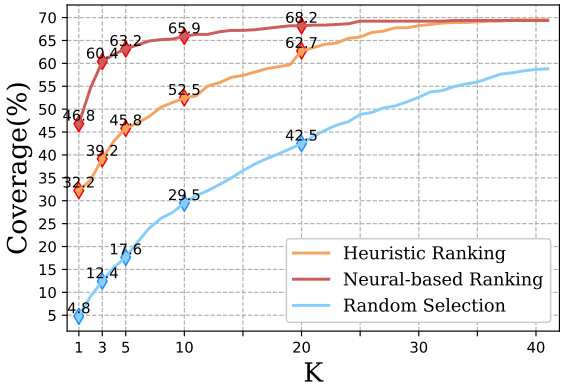

Figure 3: Coverage on validation data of TAT-QA, where **K** stands for picking top k pre-computed items as input.

Heuristic Ranking We use the proposed cube linearization method to generate cube sequences and calculate the text similarity (Ramos et al., 2003) between the question and the flattened cube sequences. The ranks of pre-computed results are decided by the similarity, i.e., the result with a higher similarity will have a higher rank.

Neural-based Ranking We first train a binary classifier following table fact verification tasks. Given a question, a cube item in the cube, and a table, the classifier needs to predict whether the pre-computed result is the correct answer. We employ a BART-base architecture and use the similar configuration of TAPEX's experiments on TabFact (Chen et al., 2019). We use the output logits as the confidence score to rank the candidates.

Both heuristic ranking and neural-based ranking method use $\texttt{cube}_\texttt{ext}$ as input and more details are provided in Appendix C.1.

3.4 Coverage Discussion

To prove that our method covers most of the reasoning cases, we test the coverage of extracted cubes on validation data of TAT-QA(Zhu et al., 2021).

As is shown in Figure 3, increasing the number of pre-computed cube items will result in a greater coverage, eventually reaching about 70%. Additionally, with a neural ranker, the coverage under the same number k for $\texttt{cube}_\texttt{k}$ has increased significantly.

We also analyze cases that our cube fails to cover. We randomly pick 100 samples of WikiTQ dev set. We observe that the upper bound of cube generation does not outperform the current rule-based cube generation method by a large margin: only 15% of the fail-to-cover cubes are caused by current rules. For more details, please check Appendix A.

Models	EM	F1
TagOP	55.2	62.7
BART-Large	38.8	46.7
w. TACUBE + HR	55.2	63.7
w. TACUBE + NR	**57.1**	**65.6**
TAPEX-Large	41.5	49.6
w. TACUBE + HR	56.9	65.8
w. TACUBE + NR	**57.7**	**66.2**

Table 2: Exact match and F1 scores on TAT-QA dev set.

Models	Dev	Test
BART-Large	37.2	38.0
w. TACUBE + HR	42.1	40.0
w. TACUBE + NR	**42.9**	**40.3**
TAPEX-Large	58.9*	57.5
w. TACUBE + HR	**59.7**	**59.6**
w. TACUBE + NR	59.3	59.2

Table 3: Denotation accuracies on WikiTQ dev set and test set. We reimplement the TAPEX on WikiTQ and surpass the reported results on the dev set in TAPEX paper(57.0% $\rightarrow$ 58.9%).

4 Experimental Results and Analysis

In this section, we describe the details of TACUBE and evaluate the effectiveness of TACUBE on table-related questions answering benchmarks.

4.1 Experimental Setup

Datasets We evaluate TACUBE effectiveness on TAT-QA (Zhu et al., 2021) and WikiTQ (Pasupat and Liang, 2015). On TAT-QA, We apply the official evaluation metrics: exact match(EM) and F1 score. On WikiTQ, we choose denotation accuracy as the evaluation metric which is used in TAPEX and UNIFIEDSKG.

Baselines We mainly adopt TAPEX (Liu et al., 2021) and BART (Lewis et al., 2020) as our baselines. On TAT-QA, we also compare with state-of-the-art logical-form-based method TagOP (Zhu et al., 2021). TagOP uses Roberta (Liu et al., 2019) as its encoder and designs multiple classification heads to generate the answer.

Serialization We adopt the same serialization in TAPEX, which concatenates the input questions and the linearized table. For extracted and ranked cube items, we use cube serialization discussed in Section 3.2. Like TAPEX, we also separate different cube items using a vertical bar "|". Appendix C.3 presents examples for our serialization.

Implementation Details For WikiTQ, the fine-tuning is set up to 50 epochs with batch size of 128. For TAT-QA, we set 50 epochs and batch size equal as 24. The beam size is 4 for both datasets. We set the initial learning rate as 3×10^{-5} on WikiTQ and 5×10^{-5} on TAT-QA for TAPEX models, and the learning rate as 5×10^{-5} for BART models.

Selection of k For pre-computed cube item number k for $\texttt{cube}_\texttt{k}$, we mainly select it according to cube-generation results(basically from Figure 3). When cube's coverage of current k is high, we decide such k a proper choice. Moreover, in Appendix C.3, we provide an ablation study on k and results show a large k does not significantly perturb the model's performance, i.e., the model is insensitive with increasing k. We choose $k = 5$ for WikiTQ, and $k = 10$ for TAT-QA. Appendix C.2 presents more details about our experimental configuration, including an ablation study on different choices of cube item number k. In all, we think TACUBE is simple but effective.

4.2 Main Results

Table 2 and Table 3 present the evaluation results of various models' performance, where **HR** and **NR** are abbreviations for **H**euristic **R**anking and **N**eural-based **R**anking. Across all instances, we observe a marginal increase in performance. (i) On the dev set of TAT-QA, TACUBE registers the EM of 57.7% and the F1 score of 66.2%, surpassing the baseline TAPEX by 16.6% and TagOP by 3.5%. (ii) On both dev and test set of WikiTQ, TACUBE also achieves new state-of-the-art denotation accuracy of 59.7%(+0.8%) and 59.6%(+2.1%).

On each benchmark, we also study TACUBE's effect over arithmetic/aggregation cases. For TAT-QA, we test over all the annotated arithmetic cases Moreover, on TAT-QA, the F1 score is always equal to EM on arithmetic/aggregation involved cases, thus we only report EM in the table. For WikiTQ, we extract a subset of the original dataset using annotations in the Squall (Shi et al., 2020) dataset, which manually annotates SQL query corresponding to the question and table, covering 80% examples for train and dev set of the WikiTQ. Please check Appendix B for more details.

The results are presented in Table 4 and Table 5. The models with TACUBE substantially outperform on each operator. Again, TAPEX shows its power in aggregation operator: among extracted `count` operations, TAPEX without TACUBE achieves denotation accuracy of 64.0%, which outperforms the BART baseline by over 30%.

Further, because the proportion and the abso-

Models	CR.	Avg	Sum	Diff	Div	Arith.
BART-Large	0.0	2.8	1.8	4.3	5.6	2.6
w. TACUBE + HR	69.7	**62.4**	26.3	38.7	5.6	44.0
w. TACUBE + NR	**78.7**	55.3	**40.4**	**40.9**	**27.8**	**47.5**
TAPEX-Large	0.6	3.5	1.8	6.0	5.6	3.5
w. TACUBE + HR	78.1	53.2	**33.3**	**51.9**	**33.3**	**50.0**
w. TACUBE + NR	**79.4**	**53.9**	**33.3**	48.5	27.8	49.6

Table 4: Exact match on arithmetic/aggregation involved cases of TAT-QA dev set. **Arith.** represents all cases involving arithmetic operations. **CR.** stands for the **c**hange-**r**atio operator.

Models	Diff	Sum	Avg	Count	Total
BART-Large	2.4	2.5	10.0	33.6	30.2
w. TACUBE + HR	21.4	**20.0**	**40.0**	**46.1**	**43.5**
w. TACUBE + NR	**33.3**	15.0	**40.0**	45.7	**43.5**
TAPEX-Large	19.0	10.0	10.0	64.0	58.4
w. TACUBE + HR	**33.3**	**22.5**	**20.0**	**65.4**	**61.0**
w. TACUBE + NR	**33.3**	17.5	10.0	65.3	60.6

Table 5: Denotation accuracy on different operators of WikiTQ dev set.

lute number of WikiTQ samples requiring `sum`, `average`, `diff` and other operations are quite small, it is not conducive to TACUBE's training. Despite such negative factors, using TACUBE still brings performance boost on each operator. On TAT-QA that contains more arithmetic cases, TACUBE outperforms baselines by a larger margin.

4.3 Further Exploration

Experiments on Higher-order Operations Although current heuristic cube generation is effective for the two datasets, it may result in combination explosion for higher-order operations, making the generated cubes too heavy to be fed into the input sequence. Thus, we attempt to generate cubes using a neural-based model. We test on FinQA (Chen et al., 2021b) which contains high-order operations. We simply use the retrieval results of FinQANet (Chen et al., 2021b) and use a BART-large model to generate Top-5 programs. We augment TAPEX with the generated programs as cubes. The Top-1 cube execution accuracy is 65.9%(on dev set); the execution accuracy of TACUBE is 66.9%(on dev set). TACUBE outperforms FinQANet by about 2%(compared to results on FinQA leaderboard after official bug fix).

5 Related Work

Table QA with PLMs There are a variety of works applying PLMs on table QA. To perform question answering over semi-structured tabular data, prior work formulated the problem into a semantic parsing task and adopted semantic parsers to operate over tables (Yin et al., 2020; Shi et al., 2018; Liang et al., 2018; Cheng et al., 2021a). To better encode and represent the tabular data, work also focused on: (i) table input featurization, which may conclude extra information about tabular data, e.g., column/row embeddings based on cell location (Wang et al., 2021; Herzig et al., 2020; Eisenschlos et al., 2021); (ii) structural-aware encoding, which designs visualization matrix for structure-pruned attention of transformers (Wang et al., 2021; Eisenschlos et al., 2021) or produce row-wise/column-wise embedding (Yin et al., 2020); (iii) table pre-training using collected tabular data corpus (Yin et al., 2020; Liu et al., 2021; Herzig et al., 2020; Eisenschlos et al., 2021). Recent work has shown the potential of directly using auto-regressive PLMs for table QA. The decoders of PLMs are expected to autoregressively generate the answers without extra architecture design (Liu et al., 2021; Xie et al., 2022) , and such practice achieves competable performance on multiple table QA benchmarks. There are also works focusing on table augmentation such as column expansion, to provide extra information at the input phase (Suadaa et al., 2021; Zhao et al., 2022a) and thus to enhance the semantic parsing or table-to-text task. In this paper, we introduce the pre-computed data cube to provide information for numerical reasoning in a systematic way, and we adopt auto-regressive PLMs to decode the answer.

Numerical Reasoning over tabular data Numerical reasoning is important in different NL tasks (Dua et al., 2019; Zhu et al., 2021; Zhao et al., 2022b). For tables, they usually contain well-organized numeric values. Moreover, in several end-user tools, the spreadsheet formulas in cells imply the numerical relationships among the table cells (Dong et al., 2022). Therefore, a wide range of tasks require numerical reasoning over tabular data, such as table-to-text (Suadaa et al., 2021; Cheng et al., 2021b), formula prediction (Cheng et al., 2021a; Chen et al., 2021a) and table fact verification (Chen et al., 2019; Aly et al., 2021) and table question answering (Pasupat and Liang, 2015). Recently, various benchmarks are proposed to solve table QA problems and contain a large proportion of numerical reasoning examples (Chen et al., 2021b; Zhu et al., 2021; Zhao et al., 2022b). Moreover, datasets like CubeQA (Höffner et al.,

2016) directly collects QA samples on generic RDF data which require aggregation operations.

Retrieval-based ODQA Most existing methods deals with open-domain QA(ODQA) by retrieving evidence over documents (Chen et al., 2017), knowledge graph triples (Verga et al., 2021) and collection of question-answer pairs (Chen et al., 2022; Xiao et al., 2021) to aggregate the answer using retrieved evidence. Different from these works, our method does not directly transform the table QA into a retrieval problem; instead, the model needs to leverage the pre-computed cube and may perform post-processing during answer decoding, such as adding a scale string and further computation based on the cube items.

6 Conclusion

In this paper, we focus on numerical reasoning problems over tabular question answering. We propose our method, namely TACUBE, which automatically extracts pre-computed results for a given question and a table following the designed cube generation rules. The pre-computed cube is serialized and fed to the model in the input phase, mitigating the gap in numerical reasoning skills for PLMs. TACUBE is tested over multiple table QA benchmarks using an encoder-decoder architecture and achieves new SOTA performance on each of them. Further analysis shows the performance boost mainly comes from the numerical reasoning examples in the benchmarks.

Limitations

The performance of TACUBE on downstream tasks is constrained by the following issues: (i) the coverage of TACUBE, i.e., the pre-computed results should be accurate enough (ii) the ability of the model, i.e., the model should be capable to decide which numeric value provided in the input sequence is the answer to the question.

For (i), although we propose a general method for TACUBE generation, the coverage can still be improved through other designs, e.g., to include more operators, and to support higher-order data cubes. Moreover, we believe the generation can also be realized through a neural-based method, which may result in higher accuracy and produce an automatic generation pipeline. For (ii), we think it can be improved in two directions: First, to better represent a cube item, e.g., to use more accurate textual information in the table to describe each operand; Secondly, to leverage more unsupervised data in the pre-training phase to learn to use TACUBE to improving numerical reasoning skills.

Ethics Statement

In this work, we propose a pre-computing method to generate candidate answers for numerical reasoning questions over tabular data. We do not collect additional tabular data for this work. The datasets we use are all collected from Wikipedia or publicly available websites in English. WikiTQ (Pasupat and Liang, 2015) is available under the CC BY-SA 4.0 license and TAT-QA (Zhu et al., 2021) is under the MIT license. Both datasets permit us to modify and publish.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. Feverous: Fact extraction and verification over unstructured and structured information. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Wenhu Chen, Pat Verga, Michiel de Jong, John Wieting, and William Cohen. 2022. Augmenting pre-trained language models with qa-memory for open-domain question answering. *arXiv preprint arXiv:2204.04581*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint:1909.02164*.

Xinyun Chen, Petros Maniatis, Rishabh Singh, Charles Sutton, Hanjun Dai, Max Lin, and Denny Zhou. 2021a. Spreadsheetcoder: Formula prediction from semi-structured context. In *International Conference on Machine Learning*, pages 1661–1672. PMLR.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. 2021b. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

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

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

Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. 2022. Table pretraining: A survey on model architectures, pretraining objectives, and downstream tasks. *arXiv preprint arXiv:2201.09745*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. 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.

Julian Eisenschlos, Maharshi Gor, Thomas Mueller, and William Cohen. 2021. Mate: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619.

Yan Gao, Jian-Guang Lou, and Dongmei Zhang. 2019. A hybrid semantic parsing approach for tabular data analysis. *arXiv preprint arXiv:1910.10363*.

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. 1997. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535.

Alon Y Halevy. 2001. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294.

Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques*. Elsevier.

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

Konrad Höffner, Jens Lehmann, and Ricardo Usbeck. 2016. Cubeqa—question answering on rdf data cubes. In *International semantic web conference*, pages 325–340. Springer.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. *Advances in Neural Information Processing Systems*, 31.

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

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

Panupong Pasupat and Percy Liang. 2015. 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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of EMNLP*.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. Towards table-to-text generation with numerical reasoning. In *Proceedings of the 59th Annual*

Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1451–1465.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William Cohen. 2021. Adaptable and interpretable neural memoryover symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3678–3691.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. 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.

Jinfeng Xiao, Lidan Wang, Franck Dernoncourt, Trung Bui, Tong Sun, and Jiawei Han. 2021. Open-domain question answering with pre-constructed question spaces. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 61–67.

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

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. 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.

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

Chen Zhao, Yu Su, Adam Pauls, and Emmanouil Antonios Platanios. 2022a. Bridging the generalization gap in text-to-sql parsing with schema expansion. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5568–5578.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022b. Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv:1709.00103*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287.

A Failed-to-extract cases

We categorize cases of failed extraction into four categories: (i) **Outside Knowledge**: We need to have both the knowledge provided by the form as well as external knowledge in order to answer such questions. Take nt-7969 as an instance, the question answering the times of an athlete compete in the game. While in the original table(which is shown in Figure 4), one row is denoted as "DNF" which means "do not finish". Such cases need outside knowledge in sports terminology and are hard to generate the answer. (ii) **Non-number Pattern**: The answers to these questions contain non-number patterns, while the pre-computed cube results are limited to numerical data only. (iii) **Rule-uncovered Cases**: As stated in Section 3.1, current designed rule for cube generation only considers naive first-order data cubes. Thus, the extracted cubes can not include compositional computation. Moreover, for unusual numeric formats, such dates, length in ft., or magnitude, it is non-trivial to design a general rule and currently such cases are not covered. (iv) **Other Cases**: Answers are either incorrectly annotated or the reasoning process is unclear.

A.1 Outside Knowledge

Question: how many times did imma clopes compete?
Table: See Figure 4
Answer: 5
Analysis: The question answering the times of an athlete compete in the game. While in the original table(which is shown in Figure 4), one row is denoted as "DNF" which means "do not finish". Such cases need outside knowledge in sports terminology and are hard to generate the answer.

A.2 Non-number Pattern

Question: what is the difference between the time air uganda commenced operations and skyjet airlines commenced operations?
Table: See Figure 5
Answer: 4 years
Analysis: The answer to the question includes a non-number pattern. Thus, the value "4" is included in our precomputed cube results, however, the answer "4 years" is not.

A.3 Rule-uncovered Case

Question: what was the average time for the americans?
Table: See Figure 6

Outside Knowledge (9%)	e.g., nt-7969 **question**: how many times did imma clopes compete? **answer**: 5
Non-number Pattern (1%)	e.g., nt-6701 **question**: what is the difference between the time air uganda commenced operations and skyjet airlines commenced operations? **answer**: 4 years
Rule-uncovered Case (15%)	e.g., nt-2401 **question**: what was the average time for the americans? **answer**: 4:19:41
Other Case (3%)	e.g. nt-10206 **question**: how many years ago did ne-yo play as mixx? **answer**: 8
Correct Case (72%)	-

Table 6: Detailed analysis on TACUBE fail-to-cover cases. We randomly pick 100 samples in WikiTQ dev set and manually check the extraction results.

Year	Competition	Venue	Position	Notes
1995	World Indoor Championships	Barcelona, Spain	11th	Pentathlon
1996	Olympic Games	Atlanta, Georgia, USA	24th	Heptathlon
1997	World Championships	Athens, Greece	16th	Heptathlon
1998	European Indoor Championships	Valencia, Spain	7th	Pentathlon
1998	European Championships	Budapest, Hungary	14th	Heptathlon
2000	Olympic Games	Sydney, Australia	DNF	Heptathlon

Figure 4: outside knowledge

AIRLINE	ICAO	IATA	CALLSIGN	COMMENCED OPERATIONS
Air Uganda	UGA	U7	UGANDA	2007
Eagle Air (Uganda)	EGU	H7	AFRICAN EAGLE	1994
Fly540 Uganda	FUL		ORANGE CRANE	2008
Pearl Air Services	PBY		PEARL SERVICES	
Royal Daisy Airlines	KDR	6D	DARLINES	2005
Skyjet Airlines	SJU	UQ	SKYJET	2003
Africa Safari Air	ASA	AS	ASA	2013
Uganda Air Cargo	UCC		UGANDA CARGO	1994
United Airlines Limited				

Figure 5: non-number pattern

Rank	Name	Nationality	Time	Notes
1	Janelle Atkinson	Jamaica	4:16.89	Q
2	-	-	-	Q
3	Kaitlin Sandeno	United States	4:18.97	Q
4	Julia Stowers	United States	4:19.84	Q
5	-	-	-	Q
6	-	-	-	Q
7	-	-	-	Q
8	-	-	-	Q

Figure 6: rule-uncovered case

Answer: 4:19.41
Analysis: Due to the fact that the data format in the table is a date, the calculation of the result requires parsing complex data.

A.4 Other Case

Question: how many years ago did ne-yo play as mixx? **Table:** See Figure 7 **Answer:** 8

Year	Title	Role	Notes
2006	Save the Last Dance 2	Mixx	Direct to video
2007	Nick Cannon/Wild 'N Out	Himself	Improv Comedy
2007	Stomp the Yard	Rich Brown	Film
2011	CSI: NY	The hitman	Episode 7.14 "Smooth Criminal"
2011	The Fresh Beat Band	Himself	Special episode "Band in a Jam"
2011	Battle: Los Angeles	Specks	Film
2012	Empire Girls: Julissa & Adrienne	Himself	Reality-Show
2012	Red Tails	Andrew 'Smoky' Salem	Film
2012	I Heart Tuesdays	None	(TV), creator
2012	The X Factor	Guest Mentor	
2012	Never Mind the Buzzcocks	Guest Host	
2012	90210	Guest star	

Figure 7: other cases

Analysis: The question answering "how many years ago" and the answer is 8, which implies such sample is annotated in the year 2014. However, it is too hard to know such information, making reasoning almost impossible.

B Numerical Reasoning Case Study Details

TAT-QA TAT-QA annotates whether a question needs arithmetic operations. But TAT-QA does not directly annotate the operation. We extract the operator following the baseline method TagOP's practice, which conclude common operators from the **derivation**, **facts** and **answer** annotations in TAT-QA including *sum*, *count*, *average*, *multiplication*, *division*, *difference* and *change ratio*.

WikiTQ For WikiTQ, the original dataset contains few available information about numerical reasoning cases. Meanwhile, the Squall dataset (Shi et al., 2020) manually annotates SQL query corresponding to the question and table, covering 80% examples for train set and dev set of WikiTQ. With Squall annotations, We determine the operator by the aggregation/arithmetic keywords appearing in the SQL query and get the performance over each operator.

Moreover, for operator `sum`, we find that the answer can be already in the original table without additional need to compute. Take the cases in Figure 8 and Figure 9 for example: Both questions ask the total / overall number, but the answers are already concluded in the original table under "**Total**" row / column. It is ambiguous to determine for such cases whether cube information is well leveraged, and we remove such cases manually.

Finally we get the number of each operator. We get 41 cases of difference, 40 of summation, 10 of average and 709 of count respectively. We report the corresponding denotation accuracy on these extracted cases in Table 5.

Season	Team	Country	Competition	Matches	Goals
1999	Djurgårdens IF	Sweden	Allsvenskan	15	1
2000	Djurgårdens IF	Sweden	Superettan	15	3
2001	Djurgårdens IF	Sweden	Allsvenskan	22	7
2002–2003	Grazer AK	Austria	Bundesliga	24	6
2003	Denizlispor	Turkey	Süper Lig	3	0
2003	Landskrona BoIS	Sweden	Allsvenskan	11	3
2004	Landskrona BoIS	Sweden	Allsvenskan	22	4
2005	Djurgårdens IF	Sweden	Allsvenskan	24	12
2006	Djurgårdens IF	Sweden	Allsvenskan	17	6
2007	Djurgårdens IF	Sweden	Allsvenskan	23	4
2008	Djurgårdens IF	Sweden	Allsvenskan	29	6
2008–09	Esbjerg fB	Denmark	Superliga	6	0
2010	AaB	Denmark	Superliga	3	1
2011	Assyriska FF	Sweden	Superettan	19	5
Total	Total	Total	Total	233	58

nt-3135

how many matches overall were there?

233

Figure 8: Manually deleted `sum` case 1

Rank	Nation	Gold	Silver	Bronze	Total
1	Cuba	4	3	2	9
2	Canada	4	2	1	7
3	United States	2	0	2	4
4	Mexico	1	1	0	2
5	Ecuador	1	0	0	1
6	Argentina	0	4	3	7
7	Brazil	0	2	2	4
8	Chile	0	0	1	1
8	Venezuela	0	0	1	1
Total	Total	12	12	12	36

nt-7291

how many total medals were there all together?

36

Figure 9: Manually deleted `sum` case 2

C Implementation Details

C.1 Ranking Methods

The neural ranking implementation is similar to the BART's practice on sequence classification tasks (Lewis et al., 2020). We adopt TAPEX-base, and feed the same input in both encoder and decoder. We use the last token's hidden state of the decoder, and feed it into a binary classifier. We rank all the cube item according to the output logits of the binary classifier, no matter whether the prediction label is positive or negative.

C.2 More on Experimental Configuration

We mainly follow the TAPEX and UNIFIEDSKG for hyperparameter selection, including batch size and initial learning rate. The main result is achieved using BART/TAPEX large-sized model, which contains 406M parameters. Moreover, We use Adafactor optimizer and linear learning rate decay on all

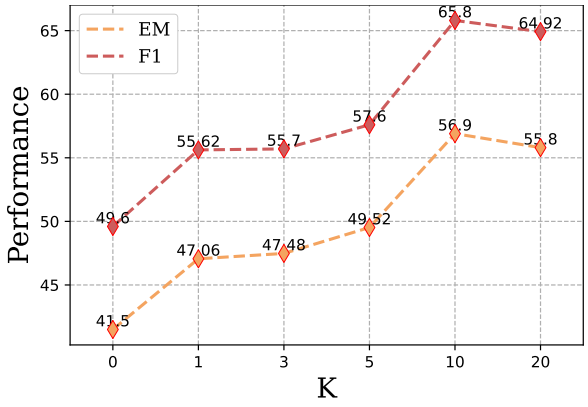

Figure 10: Ablation study on different choices of cube item number k. We use TAPEX as the base model, and evaluate EM and F1 score on TAT-QA dev set.

fine-tuning tasks. All fine-tuning experiments is done on 4 NVIDIA Tesla V100 GPUs. The re-implemented denotation accuracy of TAPEX on WikiTQ surpasses the officially reported results and we present the higher score in Table 3.

We also made an ablation study on the choice of number k, which decides the maximum number of cube items in one cube fed to the model input. Figure 10 presents the EM and F1 performance on TAT-QA dev set.

C.3 Serialization Example

As discussed in Section 4.1, we will concatenate the cube sequence with the question sequence and tabular sequence. Here we present some real cases in TAT-QA dataset and WikiTQ dataset.

TAT-QA The input contains both table and text sequence. We append the cube sequence after the table sequence. The example sequence is shown as below, which contains a question sequence, a table sequence, a cube sequence and a textual sequence.

```
what was the change in the amount for appliances
 in 2019 from 2018? table: col : | | fiscal |
row 1 : | 2019 | 2018 | 2017 row 2 : | | (in
millions) | row 3 : transportation solutions: |
| | row 4 : automotive | $ 5,686 | $ 6,092 | $
5,228 row 5 : commercial transportation | 1,221
| 1,280 | 997 row 6 : sensors | 914 | 918 | 814
row 7 : total transportation solutions | 7,821 |
 8,290 | 7,039 row 8 : industrial solutions: | |
 | row 9 : industrial equipment | 1,949 | 1,987
| 1,747 row 10 : aerospace, defense, oil, and
gas | 1,306 | 1,157 | 1,075 row 11 : energy |
699 | 712 | 685 row 12 : total industrial
solutions | 3,954 | 3,856 | 3,507 row 13 :
communications solutions: | | | row 14 : data
and devices | 993 | 1,068 | 963 row 15 :
appliances | 680 | 774 | 676 row 16 : total
communications solutions | 1,673 | 1,842 | 1,639
 row 17 : total | $ 13,448 | $ 13,988 | $ 12,185
cube: change Appliances Fiscal 2018 (in millions
) $ 6,092 2019 $ 5,686 answer : 94 | change
Appliances 2019 $ 5,686 Fiscal 2018 (in millions
) $ 6,092 answer : -94 | change Appliances 2017
$ 5,228 Fiscal 2018 (in millions) $ 6,092 answer
 : -98 | percentage change Appliances Fiscal
2018 (in millions) $ 6,092 2019 $ 5,686 answer :
 -12.14 | percentage change Appliances 2019 $
5,686 Fiscal 2018 (in millions) $ 6,092 answer :
 13.82 | percentage change Appliances 2017 $
5,228 Fiscal 2018 (in millions) $ 6,092 answer :
 14.5 | passage: (1) Industry end market
information is presented consistently with our
internal management reporting and may be revised
 periodically as management deems necessary. |
Net sales by segment and industry end market(1)
were as follows:
```

WikiTQ We append the cube sequence after the table sequence for WikiTQ. The input sequence contains three parts: a question sequence, a table sequence and a cube sequence.

```
what is the difference in runners-up from
coleraine academical institution and royal
school dungannon? table: col : school | location
 | outright titles | shared titles | runners-up
| total finals | last title | last final row 1 :
 methodist college belfast | belfast | 35 | 2 |
25 | 62 | 2014 | 2014 row 2 : royal belfast
academical institution | belfast | 29 | 4 | 21 |
 54 | 2007 | 2013 row 3 : campbell college |
belfast | 23 | 4 | 12 | 39 | 2011 | 2011 row 4 :
 coleraine academical institution | coleraine |
9 | 0 | 24 | 33 | 1992 | 1998 row 5 : the royal
school, armagh | armagh | 9 | 0 | 3 | 12 | 2004
| 2004 row 6 : portora royal school |
enniskillen | 6 | 1 | 5 | 12 | 1942 | 1942 row 7
 : bangor grammar school | bangor | 5 | 0 | 4 |
9 | 1988 | 1995 row 8 : ballymena academy |
ballymena | 3 | 0 | 6 | 9 | 2010 | 2010 row 9 :
rainey endowed school | magherafelt | 2 | 1 | 2
| 5 | 1982 | 1982 row 10 : foyle college |
londonderry | 2 | 0 | 4 | 6 | 1915 | 1915 row 11
 : belfast royal academy | belfast | 1 | 3 | 5 |
 9 | 1997 | 2010 row 12 : regent house grammar
school | newtownards | 1 | 1 | 2 | 4 | 1996 |
2008 row 13 : royal school dungannon | dungannon
 | 1 | 0 | 4 | 5 | 1907 | 1975 row 14 : annadale
 grammar school (now wellington college) |
belfast | 1 | 0 | 1 | 2 | 1958 | 1978 row 15 :
ballyclare high school | ballyclare | 1 | 0 | 1
| 2 | 1973 | 2012 row 16 : belfast boys' model
school | belfast | 1 | 0 | 0 | 1 | 1971 | 1971
row 17 : grosvenor high school | belfast | 1 | 0
 | 0 | 1 | 1983 | 1983 row 18 : wallace high
school | lisburn | 0 | 0 | 4 | 4 | n/a | 2007
row 19 : derry academy | derry | 0 | 0 | 2 | 2 |
 n/a | 1896 row 20 : dalriada school |
ballymoney | 0 | 0 | 1 | 1 | n/a | 1993 row 21 :
 galway grammar school | galway | 0 | 0 | 1 | 1
| n/a | 1887 row 22 : lurgan college | lurgan |
0 | 0 | 1 | 1 | n/a | 1934 row 23 : omagh
academy | omagh | 0 | 0 | 1 | 1 | n/a | 1985 row
 24 : sullivan upper school | holywood | 0 | 0 |
 1 | 1 | n/a | 2014 ; cube : difference school
runners-up royal school dungannon coleraine
academical institution answer : 20 | difference
location runners-up dungannon coleraine answer :
 20 ;
```