TabSQLify: Enhancing Reasoning Capabilities of LLMs Through Table Decomposition

Md Mahadi Hasan Nahid
University of Alberta
mnahid@ualberta.ca

Davood Rafiei
University of Alberta
drafiei@ualberta.ca

Abstract

Table reasoning is a challenging task that requires understanding both natural language questions and structured tabular data. Large language models (LLMs) have shown impressive capabilities in natural language understanding and generation, but they often struggle with large tables due to their limited input length. In this paper, we propose **TabSQLify**, a novel method that leverages text-to-SQL generation to decompose tables into smaller and relevant sub-tables, containing only essential information for answering questions or verifying statements, before performing the reasoning task. In our comprehensive evaluation on four challenging datasets, our approach demonstrates comparable or superior performance compared to prevailing methods reliant on full tables as input. Moreover, our method can reduce the input context length significantly, making it more scalable and efficient for large-scale table reasoning applications. Our method performs remarkably well on the WikiTQ benchmark, achieving an accuracy of 64.7%. Additionally, on the TabFact benchmark, it achieves a high accuracy of 79.5%. These results surpass other LLM-based baseline models on gpt-3.5-turbo (chatgpt). TabSQLify can reduce the table size significantly alleviating the computational load on LLMs when handling large tables without compromising performance.

1 Introduction

Tables serve as the most prevalent forms of structured information across diverse domains, ranging from databases and spreadsheets to open data repositories, web pages and document collections. Developing natural language interfaces for tabular data poses a significant challenge, primarily in terms of effectively interpreting the semantics of table cells and understanding the relationships between cell values in response to a user query. This challenge is accentuated when tables are enveloped in text, such as titles, captions, and contextual text

Table: Figure skating at the Asian Winter Games

Rank	Nation	Gold	Silver	Bronze	Total
1	China	13	9	13	35
2	Japan	7	10	7	24
3	Uzbekistan	1	2	3	6
4	Kazakhstan	2	2	0	4
5	North Korea	1	0	1	2
6	South Korea	0	0	2	2
Total		24	23	26	73

Q: who received more bronze medals: japan or south korea?
A: **Japan**

Figure 1: An example of table-based question answering.

within a document. In these instances, the scope of reasoning expands beyond the confines of table cells to incorporate the surrounding natural language text. This reasoning is essential for many downstream tasks such as table-based fact verification and table-based question answering (TableQA). As depicted in Figure 1, table-based reasoning is intricate, demanding sophisticated textual, numerical, and logical reasoning across both unstructured text and (semi-)structured tables.

Recent studies highlight the impressive capability of LLMs in reasoning over both text and tabular data. However, these works typically utilize the full table as context for reasoning (Chen, 2023), limiting their ability to large tables. In particular, LLMs operate under a maximum token limit, and when processing a large table, there is a risk of potential truncation of the input or hallucination in the output (Chen, 2023; Ji et al., 2023). This limitation poses difficulties in handling large tables, making it impractical to encompass the entire table within the maximum token boundary of a prompt. Chen
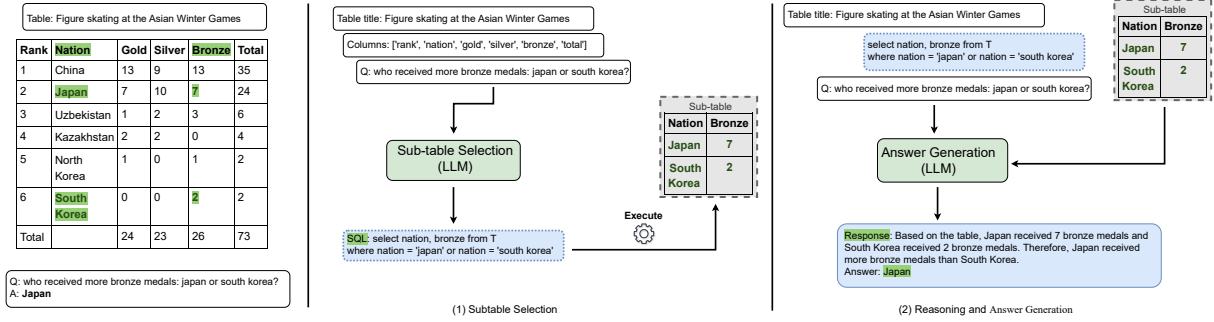
Figure 2: Overview of **TabSQLify**, consisting of two steps: **(1)** generating SQL queries from natural language questions or statements and executing the SQL queries on the original tables to obtain sub-tables containing only essential information, and **(2)** using LLMs with the sub-table and the question or claim to generate the answer.

(2023) highlights that LLMs struggle to generalize when confronted with "large" tables containing 30 or more rows, leading to a decline in accuracy as the table size increases. While there have been works to decompose both questions and tables using LLMs (Ye et al., 2023), this line of work still requires providing the full table to the LLM and cannot scale to large tables. The question studied in this work is if the size of a table can be reduced before passing it to the language model without impacting its performance.

In this work, our aim is to leverage the symbolic representation capabilities of LLMs to reduce table size and their robustness to natural language variations for addressing formatting differences. Symbolic models, such as text-to-SQL, are not affected by table size and can reliably scale to large tables. However, for reliable storage and querying in a relational database, tables are expected to adhere to a more rigorous formatting. Tables in the wild, such as those found on the web, often lack this formatting, necessitating substantial preprocessing and normalization efforts to convert the content (Cheng et al., 2023). LLMs are well-suited for resolving potential differences in the formating of rows and cell values. This work aims to strike a balance between table reasoning and table decomposition. Our approach involves using symbolic methods to narrow down the task to a targeted region in a table and then utilizes LLMs to reason over the limited relevant information.

We propose **TabSQLify**, a novel approach that integrates symbolic methods with the reasoning power of LLMs. TabSQLify leverages text-to-SQL generation to decompose large tables into smaller and relevant sub-tables for table reasoning tasks. The method involves two key steps: **(1)** generating SQL queries from natural language questions or statements using LLMs under few-shot prompting, then executing the SQL queries on the original tables to obtain sub-tables containing only essential information for answering questions or verifying statements, and **(2)** using LLMs with the sub-table and the question or claim to generate the answer. The core concept of the approach is to utilize the natural language understanding and generation strengths of LLMs while reducing their burden in table encoding and reasoning (see Figure 2). Decomposing tables into sub-tables offers several advantages, including **(1)** reducing input length for improved scalability and efficiency in reasoning tasks involving large tables, **(2)** filtering out irrelevant and redundant information that do not contribute to the reasoning process, hence making the reasoning more focused, and **(3)** providing an intermediate representation (in this case, SQL queries and sub-tables) that is more interpretable and explainable for tracing and verification purposes.

We evaluate our method on four challenging table reasoning datasets: WikiTQ (Pasupat and Liang, 2015), FeTaQA (Nan et al., 2022), TabFact (Chen et al., 2020) and WikiSQL (Zhong et al., 2017). Our evaluation on table-based question answering and fact verification tasks show that our method outperforms other LLM-based baselines, with gpt-3.5-turbo (chatgpt) as the LLM. Moreover, our method can significantly reduce the input length, making it more scalable and efficient for large-scale table reasoning applications than existing methods that require the full table context as input.

The contributions of this paper are as follows:

1. We present a novel approach that utilizes text-

to-SQL generation to decompose tables into smaller, contextually relevant sub-tables, particularly designed for table reasoning tasks. This method offers a substantial reduction in table size, proving particularly advantageous for large tables that exceed the maximum allowable context window of LLMs.

2. Our model outperforms some of the leading models that employ multiple responses and self-consistency. Clearly using those techniques can further boost the performance of our method.

3. Our evaluation on challenging table reasoning datasets demonstrates the remarkable performance of our method compared to existing methods that rely on full tables as input. A comprehensive evaluation across various tasks is conducted to elucidate both the advantages and constraints of our approach.

2 Related Work

Our work is closely intertwined with the literature on semantic parsing of questions and table schema (also known as text to data) as well as the reasoning applied to semi-structured tables (alternatively known as data to text).

2.1 Semantic Parsing: Text to Data

Table-based reasoning conventionally involves semantic parsing of questions and subsequent execution of the generated queries on tables. Traditional models in this domain were often domain-specific, supporting controlled natural language (Popescu et al., 2003; Li et al., 2007), and posed challenges in adaptation to new domains or datasets. However, recent models leveraging machine learning techniques or large language models are trained on extensive datasets and query repositories, supporting a shift towards greater domain-independence. In particular, LLMs, when used with few-shot prompting, serve as powerful code generators, and techniques such as controlled decoding further improves the reliability of code generation (Brown et al., 2020; Rajkumar et al., 2022; Pourreza and Rafiei, 2023; Chang and Fosler-Lussier, 2023; Ni et al., 2023).

Cross-domain benchmarks such as WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2018), CoSQL (Yu et al., 2019a), SParC (Yu et al., 2019b), and BIRD (Li et al., 2023) have played a pivotal

role in advancing this field, offering diverse examples of natural language queries paired with formal query language counterparts, such as SQL.

Glass et al. (Glass et al., 2021) innovatively explores methods to capture both row and column semantics, improving the model's query comprehension. Inner Table Retrieval (ITR) (Lin et al., 2023) employs a similarity-based approach for locating sub-tables. These approaches involve pre-training and fine-tuning, which heavily rely on specific datasets. This reliance makes them inapplicable without access to a corresponding training dataset, while the need for optimal hyperparameters further limits their generalization.

In this line of work, the reasoning is generally done on questions and table schemata, with the expectation that the data in a table strictly adheres to the table schema (e.g., all values in a column having the same data type).

2.2 Table Reasoning: Data to Text

The relevant models can be categorized into more traditional models and recent LLM-based models. Many early models undergo pre-training on both tables and text to acquire a joint representation, utilizing this representation for reasoning without relying on symbolic execution. Notably, TaPas (Herzig et al., 2020) retrieves masked information from tables, TAPEX (Liu et al., 2022b) employs the BART model to emulate an SQL executor, ReasTAP (Zhao et al., 2022) instills reasoning skills via pre-training, TABERT (Yin et al., 2020) encodes a subset of table content most pertinent to the input, and PASTA (Gu et al., 2022) pretrains language models to be cognizant of common table-based operations. All these models have contributed to the progress on table-based reasoning. Despite achieving commendable results through pre-training on substantial datasets, these models still necessitate fine-tuning on task-specific datasets (Chen, 2023).

Large language models have become competitive models in many domains and tasks including table reasoning, with their reasoning capabilities covering math, common sense, and symbolic reasoning. This is often done using few-shot prompts without fine-tuning (Brown et al., 2020). It has been shown that the reasoning capabilities of these models can be further improved by breaking more complex tasks into steps, using methods such as chain-of-thought (CoT) (Wei et al., 2023) and Ze-

roCoT (Kojima et al., 2023) or more carefully selecting examples in the prompt (Liu et al., 2022a). The Table-CoT Model (Chen, 2023) generates the final answer to a question by employing in-context learning and chain-of-thought prompting to table-based tasks. In contrast, the BINDER (Cheng et al., 2023) model generates programs in a programming language, extending its capabilities to solve commonsense problems. The DATER (Ye et al., 2023) approach uses LLMs to decompose tables and questions for solving table-based QA and fact verification tasks.

ReAcTable (Zhang et al., 2023) adopts the ReAct paradigm, encompassing step-by-step reasoning, code execution through external tools, intermediate table generation, and a majority voting mechanism. This method leverages LLMs to decompose the problem into multiple steps, each consisting of logical operations in the form of code to process tabular data as required. In a more recent model, LEVER (Ni et al., 2023) presents a method to enhance language-to-code generation by training to validate the generated programs based on their execution results.

StructGPT (Jiang et al., 2023) enhances LLM reasoning for structured data using an Iterative Reading-then-Reasoning approach. However, the complexity and cost of StructGPT are exacerbated by the practice of passing entire tables to LLM in the reading phase, thus limiting the model's scalability to large tables due to token limits. Chain-of-Table (Wang et al., 2024) extends Chain-of-Thought to tables, improving accuracy by transforming input tables and guiding LLM with intermediate tables during reasoning. However, it requires multiple intermediate steps and LLM calls.

Table reasoning approaches typically operate under the assumption that the tables in question are sufficiently small to be directly input into the model. This specific issue is the focus of our investigation in this paper.

3 Methodology

Our approach capitalizes on the proficiency of LLMs in parsing natural language text and generating SQL to enhance their capabilities in table reasoning. Large language models face challenges in accommodating extensive contextual information, especially when dealing with large tables that exceed their token limits. Increasing this size for large tables is unrealistic due to the quadratic time

and memory complexities of self-attention mechanism in input length. Furthermore, LLMs are more likely to produce errors when handling lengthy contexts (Chen, 2023; Ji et al., 2023). To overcome these challenges, our work efficiently identifies and extracts relevant table parts, optimizing the input prompt size without sacrificing performance.

3.1 Table Preprocessing

Although tabular data is typically stored in a relational database and queried using SQL, many tables collected from web sources lack the rigorous structure and consistency that is needed for SQL queries to retrieve correct answers. It is generally a challenge to fully clean data from different sources or with no clean lineage records. Our hypothesis is that applying some general table cleaning and relaxing the granularity of retrievals to relevant rows and columns that have the answers, instead of the exact answers, makes the SQL engine more reliable. Of course, the exact answer must be extracted at the end. This is done in our reasoning phase (§ 3.3) where an LLM is used, and it is better equipped to handle formatting differences.

For our table cleaning, we normalized numerical values and date fields. In particular, numerical values frequently feature commas, necessitating preprocessing to ensure consistency. To address this, we uniformly removed commas from all numerical entries. Additionally, the diverse date formats within the tables posed a challenge in generating accurate conditions for SQL queries. To address this, we standardized all date formats to the YYYY-MM-DD format. As an example, we converted numbers like "360,000" to "360000," and different date formats such as "31 October 2008," "31 Oct 2008" and "October 31, 2008" to the standardized "2008-10-31".

3.2 Subtable Selection

The subtable selection can be done by three strategies: (1) selecting essential columns, (2) selecting essential rows, and (3) selecting both essential columns and rows.

In this step, instead of feeding the entire table to the LLM, we provide essential table information such as the title, column names, and three example rows alongside the question. We utilize few-shot learning for this step, where we provide the LLM with a few examples. Subsequently, the LLM generates an SQL query to select the subtable based on this provided information. Selecting essential

rows may require performing grouping and aggregation, and our generated SQL queries can include GROUP BY clauses and aggregation functions.

By selecting the essential columns and rows, we are reducing the context size while optimizing the relevance of information for subsequent reasoning tasks. When employing strategies (2) or (3), sometimes essential rows may not be safely extracted, for example returning an empty table due to noisy input. In those cases, we opt for the column selection strategy. The format of the prompt used for selecting necessary columns and row is described in Figure 3.

```
Generate SQL for selecting the required rows and columns, given the question and table to answer the
question correctly.
..... ..... ..... .....
SQLite table properties:

Table: 2012–13 Exeter City F.C. season(row_number,name,league,fa_cup,league_cup,jp_trophy,total)

3 example rows:
select * from T limit 3;
row_number | name | league | fa_cup | league_cup | jp_trophy | total
0 | scot bennett | 5 | 0 | 0 | 0 | 5
1 | danny coles | 3 | 0 | 0 | 0 | 3
2 | liam sercombe | 1 | 0 | 0 | 0 | 1

Q: does pat or john have the highest total?
SQL: select name, total from T where name like '%pat%' or name like '%john%'
..... ..... ..... .....
SQLite table properties:

Table: Figure skating at the Asian Winter Games (row_number, rank, nation, gold, silver, bronze, total)

3 example rows:
select * from T limit 3;
row_number | rank | nation | gold | silver | bronze | total
0 | 1 | china | 13 | 9 | 13 | 35
1 | 2 | japan | 7 | 10 | 7 | 24
2 | 3 | uzbekistan | 1 | 2 | 3 | 6

Q: who received more bronze medals: japan or south korea?
SQL:
----- ------ ------ ------
response:  select nation, bronze from T where nation = 'japan' or nation = 'south korea'
```

Figure 3: Prompt used for the subtable selection step of TabSQLify$_{col+row}$.

It is still conceivable that the output of the subtable selection step remains large, for example when finding the top-k most popular products for large values of k. We consider this limitation as inherent to the nature of the task and not specific to our approach since the sub-table containing all the items of the top-k is necessary to answer this question.

3.3 Reasoning and Answer Generation

In this step, an LLM is employed, wherein we input the SQL derived from the previous step, the subtable obtained by executing the SQL query and the question. Depending on the domain, additional contextual information, such as the surrounding text, may also be incorporated. This approach is adopted to help the model focus on the relevant parts for understanding the context and answering the question. Moreover, we utilize few-shot learning techniques while adhering to the Chain-of-Thought prompting style. The format of the answer

generation prompt is described in Figure 4.

```
Based on the table title and execution result of the sql query bellow, find the answer to the given question correctly.
..... ..... .....
Table_title: Playa de Oro International Airport
SQL: select City, Passengers from T;

City | Passengers
United States, Los Angeles | 14,749
United States, Houston | 5,465
Canada, Calgary | 3,761
Canada, Saskatoon | 2,282
Canada, Vancouver | 2,103
United States, Phoenix | 1,829
Canada, Toronto | 1,202
Canada, Edmonton | 110
United States, Oakland | 107

Question: how many more passengers flew to los angeles than to saskatoon from manzanillo airport in 2013?
A: To find the answer to this question, let's think step by step. Based on the table, in 2013, the number of
passengers who flew to Los Angeles from Manzanillo Airport was 14,749, while the number of passengers who flew
to Saskatoon was 2,282. So, the difference in the number of passengers between Los Angeles and Saskatoon is
14,749 - 2,282 = 12,467. Therefore, the answer is 12,467.
Answer: 12,467
.... .... ....
Table_title: Figure skating at the Asian Winter Games
SQL: select nation, bronze from T where nation = 'japan' or nation = 'south korea'

nation | bronze
japan | 7
south korea | 2

Question: who received more bronze medals: japan or south korea?
A: To find the answer to this question, let's think step by step.
----- ------ ------ ------
response:  Based on the table, Japan received 7 bronze medals and South Korea received 2 bronze
medals. Therefore, Japan received more bronze medals than South Korea.
Answer: Japan
```

Figure 4: Prompts used for the answer generation step.

4 Experimental Setup

4.1 Dataset

We assess our proposed approach across four datasets centered on reasoning with tables. Given our constraints on using LLMs, in terms of the number of requests and associated costs, our method is exclusively evaluated on the test sets of these datasets, with no fine-tuning on the training sets.

WikiTQ WikiTableQuestions (WikiTQ) contains complex questions annotated by crowd workers based on Wikipedia tables. These questions involve multiple complex operations, such as comparison, aggregation, and arithmetic operations, which require reasoning over multiple entries in a table. The standard test set contains 4,344 samples (Pasupat and Liang, 2015).

FetaQA Free-form Table Question Answering (FeTaQA) contains free-form table questions that require deep reasoning and understanding. These questions are usually hard because it requires processing information from different parts of the table. Unlike WikiTQ, this dataset annotates long free-form answers. Our approach is evaluated on the test set that contains 2,003 samples (Nan et al., 2022).

TabFact Table-Fact-Checking (TabFact) is a benchmark for verifying facts based on tables, which includes statements created by crowd workers using tables from Wikipedia. For example, a statement must be judged as either "True" or

"False" based on the information in a given table. The accuracy is reported on the test-small set, which contains 2,024 statements and 298 tables (Chen et al., 2020).

WikiSQL WikiSQL is a simpler TableQA dataset, necessitating the filtering and aggregation of information from the table content. Each question in WikiSQL is associated with a ground truth SQL query, from which we extract the gold answer and compare it with our results. We present the accuracy achieved on the test set of WikiSQL (Zhong et al., 2017).

4.2 Implementation Details

In the experiments, we use gpt-3.5-turbo (chatgpt) as our language model. The prompt format mainly follows Chang and Fosler-Lussier (2023) and Tai et al. (2023), which inputs the table schema and the first three table rows. The detail LLM hyper-parameters are provided in Appendix A, and all our code and prompts are available at `https://github.com/mahadi-nahid/TabSQLify`.

4.3 Baselines

We compare our approach with several strong baseline methods. These methods can be split into two groups.

Pre-training and fine-tuning based models Our evaluation involves comparing our work with different models ranging from pre-training to fine-tuning. These models, pretrained on a large table corpus, aim to encode a given table as a plain sequence into an encoder and subsequently employ a decoder to generate an answer. As our baselines, we consider Table-BERT (Chen et al., 2020), LogicFactChecker (Zhong et al., 2020), TaPas (Herzig et al., 2020), SAT (Zhang et al., 2020), TAPEX (Liu et al., 2022b), GraPPa (Yu et al., 2020), PASTA (Gu et al., 2022) as our baslines. For FeTaQA evaluation, we compare our results against T5 (Raffel et al., 2020; Nan et al., 2022).

LLM based models For the LLM based methods with in-context learning, we compare against TableCoT (Chen, 2023), BINDER (Cheng et al., 2023), DATER (Ye et al., 2023), StructGPT (Jiang et al., 2023), ReAcTable (Zhang et al., 2023), ITR (Lin et al., 2023), LEVER (Ni et al., 2023) and Chain-of-Table (Wang et al., 2024) as our baselines.

4.4 Evaluation metrics

For the WikiTQ and WikiSQL dataset, exact match (EM) accuracy was used to check if the predicted answers were the same as the correct ones. To account for different formatting of date and number fields, we added a pre-mactching check (Cheng et al., 2023), consistent with preprocessing (§ 3.1). The accuracy of TabFact was determined using binary classification accuracy. To evaluate FeTaQA, metrics such as ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) were used. However, ROUGE score lacks the ability to gauge the faithfulness and correctness of model-generated content. In line with Chen (2023), a human evaluation was conducted across four aspects: fluency (assessing linguistic errors), correctness (ensuring accurate answers to questions), faithfulness (verifying grounding on the input table), and adequacy (evaluating the comprehensiveness of the generated sentence in covering all answers) (Nan et al., 2022).

5 Results

5.1 Model accuracy

As shown on Table 1, TabSQLify achieves an accuracy of 62.0% and 63.7% on the more challenging WikiTA dataset when reasoning is performed solely using the extracted columns and extracted rows, respectively. By extracting both the necessary columns and rows, we achieve an accuracy of 64.7%. Our model outperforms all pretrained models and LLM-based baselines, with chatgpt used as the LLM, on WikiTQ dataset [1]. It surpasses BINDER-Codex and achieves accuracy very close to the state-of-the-art model DATER. It is worth noting that, unlike our model, which considers only one response, both BINDER and DATER utilize 20 responses for the WikiTQ dataset to obtain the final answer.

For the TabFact dataset, as illustrated in Table 2, TabSQLify outperforms all LLM-based state-of-the-art approaches, with ChatGPT as the LLM. We achieve an accuracy of 79.5% when we extract the required sub-table by applying both column and row filtering. It is important to highlight that BINDER and DATER employ multiple responses and self-consistency to obtain the final answer. The reported results on TabFact are based on 50 responses for BINDER, 20 responses for DATER,

---

[1] Codex was not available at the time of running our experiments, and the reported results are from the respective papers of our baselines.

Models	Accuracy
Agarwal et al., 2019	44.1
Wang et al., 2019	44.5
TaPas	48.8
GraPPa	52.7
LEVER	62.9
ITR	63.4
GPT-3 CoT	45.7
TableCoT-Codex	48.8
DATER-Codex	65.9
BINDER-Codex	61.9
ReAcTable-Codex	65.8
SQL-Codex	61.1
BINDER-chatgpt	55.4
DATER-chatgpt	52.8
ReAcTable-chatgpt	52.5
SQL-chatgpt	54.1
TableCoT-chatgpt	52.4
StructGPT	52.2
Chain-of-Table	59.9
TabSQLify$_{col}$	62.0
TabSQLify$_{row}$	63.7
TabSQLify$_{col+row}$	**64.7**

Table 1: Accuracy compared to the baselines on WikiTQ with the official evaluator.

and only one response for our model, hence it gives a lower bound of our model performance.

Model	Accuracy
Table-BERT	68.1
LogicFactChecker	74.3
SAT	75.5
TaPas	83.9
TAPEX	85.9
SaMoE	86.7
PASTA	90.8
Human	92.1
TableCoT-Codex	72.6
DATER-Codex	85.6
BINDER-Codex	85.1
ReAcTable-Codex	83.1
ReAcTable-chatgpt	73.1
TableCoT-chatgpt	73.1
BINDER-chatgpt	79.1
DATER-chatgpt	78.0
Chain-of-Table	80.2
TabSQLify$_{col}$	77.0
TabSQLify$_{row}$	78.5
TabSQLify$_{col+row}$	**79.5**

Table 2: Experimental results on TabFact. Here, "Human" indicates the human performance (Ye et al., 2023)

For FeTaQA dataset, we achive a performance comparable to the baselines. As ROUGE metrics do not reflect the actual correctness of the model's responses, we manually evaluated 100 randomly chosen sample and quantified their performance in terms of fluency, correctness, adequacy and faithfulness. The performance is summarized in Tables 3 and 4. TabSQLify outperforms models based

on fine-tuning and pre-training, such as T5-large. The evaluation suggests that the model's output closely aligns with average human performance in terms of fluency, adequacy, and faithfulness. The correctness is notably impressive, although it falls behind human-level performance. This indicates that, utilizing TabSQLify results in high accuracy without the need for the entire table, showcasing the model's high level of precision in retrieving the relevant sub-table. We additionally assess the results using RAGAS (Gradients, 2023). The evaluation outcomes obtained from RAGAS are detailed in Appendix D.

Model	R-1	R-2	R-L
T5-small	0.55	0.33	0.47
T5-base	0.61	0.39	0.51
T5-large	0.63	0.41	0.53
TableCoT-Codex	0.62	0.40	0.52
DATER-Codex	0.66	0.45	0.56
ReAcTable	0.71	0.46	0.61
TableCoT-chatgpt	0.62	0.39	0.51
TabSQLify$_{col}$	0.57	0.34	0.47
TabSQLify$_{row}$	0.60	0.37	0.49
TabSQLify$_{col+row}$	0.58	0.35	0.48

Table 3: Experimental results on FeTaQA.

Model	Fluency	Correct	Adequate	Faithful
T5-large	94.6	54.8	50.4	50.4
Human (Chen, 2023)	95	92.4	95.6	95.6
TableCoT-chatgpt	96	82	75	87
TabSQLify$_{col}$	98	83	79	85
TabSQLify$_{row}$	96	80	77	89
TabSQLify$_{col+row}$	97	88	84	93

Table 4: Human evaluation results on FeTaQA.

TabSQLify shows an outstanding performance on the WikiSQL dataset, as demonstrated in Table 5. This dataset appears to be easier compared to the WikiTQ test dataset, with our approach achieving 76.7% accuracy. In 70% of cases, it can produce the answer in the first step, eliminating the need to pass the sub-table and question for the second step.

Model	Accuracy
SEQ2SQL	59.4%
StructGPT	65.6%
RCI (Glass et al., 2021)	89.8%
TabSQLify$_{col+row}$	**76.7%**

Table 5: Experimental results on WikiSQL. RCI is a fine tuning based model, and its results may not be directly comparable due to the model's high reliance on the training set.

5.2 Scalability and robustness

We assessed the scalability and robustness of our model by imposing a token limit on each table across three datasets: WikiTQ, FeTaQA and Tab-Fact. To accomplish this, we established cutoff thresholds to discard tokens exceeding these limits. Subsequently, we evaluated the model's performance within these constrained token boundaries. For the WikiTQ dataset, we set the cutoff threshold at 2000, while for both the TabFact and FeTaQA datasets, it was set to 600. Table 6 summarizes the distribution across different classes, illustrating the categories based on the percentage of discarded table tokens (see Appendix B for more detail).

Cut-off (%)	WikiTQ	FeTaQA	TabFact
0 - 10%	76	81	91
10 - 25%	89	143	141
25 - 50%	116	202	260
50% +	128	69	81

Table 6: The distribution of samples across various classes as a function of the percentage cut-off of table tokens.

The evaluation results for the WikiTQ dataset are presented in Table 7. Our model consistently performs well within the specified token boundary. In contrast, the performance of TableCoT is subpar. We have observed a similar trend in the other two datasets (see Tables 8 and 9).

Cut-off (%)	TableCoT	TabSQLify$_{col+row}$
0 - 10%	40.7	64.4
10 - 25%	49.4	60.6
25 - 50%	46.5	66.3
50% +	33.3	56.2

Table 7: Performance across different classes based on the percentage cut-off of table tokens in the WikiTQ dataset.

Cut-off (%)	TableCoT	TabSQLify$_{col+row}$
0 - 10%	76.9	79.1
10 - 25%	67.3	80.8
25 - 50%	63.0	70.0
50% +	55.5	72.8

Table 8: Performance across different classes based on the percentage cut-off of table tokens in the TabFact dataset

TableCoT			
Cut-off (%)	R-1	R-2	R-L
0 -10%	0.58	0.35	0.45
10-25%	0.60	0.37	0.50
25-50%	0.53	0.30	0.43
50% +	0.49	0.28	0.40
TabSQLify$_{col+row}$			
Cut-off (%)	R-1	R-2	R-L
0 -10%	0.62	0.39	0.50
10-25%	0.64	0.42	0.53
25-50%	0.55	0.32	0.44
50% +	0.51	0.31	0.41

Table 9: Performance across different classes based on the percentage cut-off of table tokens in the FeTaQA dataset

In the WikiTQ dataset, 128 tables contain >4000 tokens exceeding chatGPT's maximum token limit (4096 tokens including table and question). Table 10 reports the performance on these instances. These results reveal that both BINDER (Cheng et al., 2023) and DATER (Ye et al., 2023) face challenges when dealing with large tables. Specifically, BINDER-Codex achieves only 29.6% accuracy, while DATER achieves an accuracy of 34.6%. BINDER-chatgpt fails to produce any correct answers for these large tables. On the other hand, Chain-of-Table (Wang et al., 2024) achieves an accuracy of 44.8%.

In contrast, our model outperforms these baselines significantly. It is crucial to note that our Table-CoT achieves this accuracy because the answers for questions about those large tables are typically in the upper part, fitting within the LLM's context boundary. If the answer is elsewhere, all models fail. On the other hand, our model has no issue with the answer's position in a table, making it scalable for large tables.

Model	Acc (Large)
BINDER-Codex	29.6
BINDER-chatgpt	0.0
DATER-chatgpt	34.6
Table-CoT-chatgpt	35.1
Chain-of-Table (Wang et al., 2024)	44.8
TabSQLify$_{col}$	50.0
TabSQLify$_{row}$	**57.0**
TabSQLify$_{col+row}$	52.3

Table 10: Experimental results on Large (>4000 tokens) tables from WikiTQ. As the input tables grow larger, we observe a decline in performance for strong baseline models.
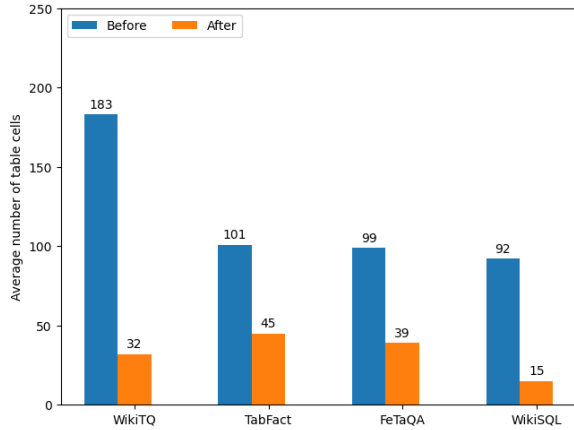
Figure 5: Reduction in table size using our row-col filtering across four datasets, showing a significant reduction of the table size.

5.3 Table size reduction

Figure 5 demonstrates the average reduction in the number of table cells before and after employing TabSQLify$_{col+row}$ across three datasets. This reduction, from 183 to 32 cells in WikiTQ, indicates a substantial decrease in sub-table size while maintaining a strong performance. Likewise, similar trends can be observed in the TabFact, FetaQA and WikiSQL datasets. When utilizing both column and row filters to extract the required subtable, direct answers to questions may be found. Specifically, in the WikiTQ dataset, TabSQLify$_{col+row}$ successfully retrieves answers in 58% of cases by executing the generated query, requiring the answer generation step only for the remaining 42% of cases.

5.4 Error Analysis

An important advantage of TabSQLify is its ability to provide the intermediate stages of reasoning path, including SQL queries and sub-tables. To conduct our error analysis, we randomly selected 100 responses generated by TabSQLify$_{row+col}$ from the WikiTQ and TabFact test sets. The identified errors are categorized into incorrect columns, incorrect conditions, incorrect reasoning, and false negatives, as listed in Table 11.

In this context, a "missing column" refers to instances where TabSQLify either selects incorrect columns or omits necessary columns to answer the question. "missing rows" denotes situations where the generated SQL query contains an erroneous condition within the WHERE clause. Cases where the extracted sub-table is adequate to answer the question, but the LLM fails to provide a correct

response, are labeled as "incorrect reasoning". Additionally, within the dataset, there are instances where the gold answer is incorrect or misjudged by the evaluator, which we classify as "incorrect annotation".

From the table, we observe that out of 100 error cases from WikiTQ, 6% involve the generated SQL query missing columns, while 56% miss required rows. The irregular format of the text in the table is identified as the primary cause. Additionally, in 29% of cases, the reasoning is found to be incorrect, while 9% exhibit incorrect annotation. In the TabFact dataset, 10% of the time, the subtable selection query misses required columns, and in 32% of cases, it misses required rows. The main source of errors is incorrect reasoning, accounting for 50% of cases, while 8% involve incorrect annotations.

Error Type	WikiTQ	TabFact
Missing Columns	6%	10%
Missing Rows	56%	32%
Incorrect Reasoning	29%	50%
Incorrect Annotation	9%	8%

Table 11: Error types of 100 samples from WikiTQ and TabFact of TabSQLify$_{col+row}$

6 Conclusion

Our proposed decomposition approach has shown promise across different table reasoning tasks, achieving remarkable performance compared to models that require the use of a full table. Our method is novel in leveraging text-to-SQL generation to decompose tables into smaller and relevant sub-tables tailored for table reasoning tasks. This approach provides a new perspective and direction for table reasoning research, and we hope it will inspire more future work on combining natural language understanding and structured data processing.

Limitations

Our approach is not without its limitations. While it shows promise in reducing table size and maintaining a strong performance, for large tables, the size of a column can exceed the context window size, and the approach may not be applicable. Also, after our preprocessing, the tables are stored in a relational tables. For less regular tables, more preprocessing may be needed.

Ethical Considerations

The datasets utilized in this study are accessible through peer-reviewed articles, as specified in the references. Our source code is made openly available for future research under the MIT License. It's important to note that since our framework relies on gpt-3.5-turbo, it may inherit ethical concerns associated with gpt models, such as potential responses to toxic content or displaying a biased behavior.

References

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *International conference on machine learning*, pages 130–140. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings.

Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages.

Michael Glass, Mustafa Canim, Alfio Gliozzo, Saneem Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing row and column semantics in transformer based question answering over tables. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1212–1224, Online. Association for Computational Linguistics.

Exploding Gradients. 2023. Ragas: Evaluation framework for retrieval augmented generation. https://github.com/explodinggradients/ragas.

Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. 2022. PASTA: Table-operations aware fact verification via sentence-table cloze pre-training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4971–4983, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, 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, Online. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Chenhao Ma, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Yunyao Li, Huahai Yang, and HV Jagadish. 2007. Nalix: A generic natural language search environment for xml data. *ACM Transactions on database systems (TODS)*, 32(4):30–es.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. An inner table retriever for robust table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022a. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022b. Tapex: Table pre-training via learning a neural sql executor.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I Wang, and Xi Victoria Lin. 2023. Lever: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*.

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, Beijing, China. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *arXiv preprint arXiv:2304.11015*.

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.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models.

Chang-You Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain-of-thought style prompting for text-to-sql.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3774–3785, Hong Kong, China. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA. Association for Computing Machinery.

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, Online. Association for Computational Linguistics.

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

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 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, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, Online. Association for Computational Linguistics.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2023. Reactable: Enhancing react for table question answering.

Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. 2022. ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9006–9018, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Wanjun Zhong, Duyu Tang, Zhangyin Feng, Nan Duan, Ming Zhou, Ming Gong, Linjun Shou, Daxin Jiang, Jiahai Wang, and Jian Yin. 2020. Logical-FactChecker: Leveraging logical operations for fact checking with graph module network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6065, Online. Association for Computational Linguistics.

A LLM Hyper-parameters

We configured the in-context learning hyperparameters for gpt-3.5-turbo according to the specifications outlined in Table 12 and Table 13.

Sub table selection using Text-to-SQL			
Parameter	WTQA	FeTaQA	TabFact
temperature	0.3	0.3	0.3
top_p	1	1	1
sample_n	1	1	1
max_tokens	100	100	100
num_shots	10	6	8

Table 12: Our hyper-parameter setting of LLM for selecting required column/row

Answer Generation			
Parameter	WTQA	FeTaQA	TabFact
temperature	0.7	0.7	0.6
top_p	1	1	1
sample_n	1	1	1
max_tokens	200	64	100
num_shots	2	6	4

Table 13: Our hyper-parameters setting of LLM for the answer generation

B Scalability and Robustness Experiment

LLMs function within a restricted token boundary, allowing us to provide only a limited number of tokens as a prompt to the LLM. We report the impact of the cutoff threshold where tokens beyond the cutoff points are discarded (§ 5.2).

The cutoff percentage denotes the percentage of tokens that are truncated when the threshold is applied. For example, if a table has 4500 tokens and we set the threshold at 2000, then 2500 tokens of the original table are truncated, and the percentage is 2500/4500 = 55.56%. We separated the number of samples in different cutoff ranges (see table 5) and compared the results of those samples from different cutoff ranges in table 6 and 7.

For the WikiTQ dataset, we set the threshold at 2000 tokens. In this case, out of 4,344 samples, there are 128 samples where more than 50% of the tokens of the original table are truncated if we want

to pass the original table to the LLM with a maximum token boundary of 2000. In our approach, TabSQLify selects the relevant limited subtable from the original table for a given question. This allows us to fit the subtable within the maximum token boundary when passing it to the LLM, resulting in improved performance. The aim of this experiment is to demonstrate that TabSQLify can be useful under limited token (context) boundary conditions.

C Comparison with other models

In this section, we conduct a comparative analysis of our model against two strong baselines, DATER (Ye et al., 2023) and BINDER (Cheng et al., 2023). DATER utilizes Large Language Models (LLMs) for decomposing both questions and tables. On the other hand, BINDER stands out by offering an Application Programming Interface (API) that extends language model (LM) functionalities to programming languages such as SQL and Python. This extension broadens its grammar coverage, enabling the model to address a more diverse range of questions. However, a drawback is that both DATER and BINDER necessitates sending the entire table to the LLM and face challenges when dealing with large tables. Both DATER and BINDER leverage self-consistency (Wang et al., 2023) strategies to bolster their performance, ensuring a higher level of consistency in their responses.

In our experiment we did not consider using self-consistence decoding strategy. Using self-consistency we can push the performance even higher. Our implementation does not require any additional processing on the SQL code, unlike BINDER, which necessitates a complex re-implementation of the SQL executor (Ni et al., 2023). BINDER generates a total of 50 samples for a given table and question in the intermediate stages (Generate Neural-SQL: 50); while DATER generates 100 samples in its intermediate stages (table decomposition: 40; Generate Cloze: 20; Generate SQL: 20; reasoning: 20). In contrast, TabSQLify generates only two samples in total, making it simpler and more cost-effective. We summarize the comparison with DATER and BINDER in Table 14.

Compared to the other LLM-based approach, our approach has several benefits: (1) Unlike other models our approach do not need to provide the table data to LLM to select the target portion of

-	DATER	BINDER	TabSQLify
# stage	4	2	2
Max context size	8000	8000	4096
# of generated samples	100	50	2
sampling_n	20-50	20	1
Self Consistency	yes	yes	no
Table required	full	full	partial
Cost	high	high	low

Table 14: Comparison with the other LLM based models. TabSQLify is much simpler than the other approach.

the table. Instead we utilize text-to-sql capability of LLMs. (2) Our approach requires partial table, not full table. (3) Our model can be applied in tight token boundary (4) Considering only one response our model can achive comparable performance while other top performing model uses more than 20 responses (5) Our approach is less costly and it requires less LLM calls which can be vital factor to reduce the cost.

D RAGAS Evaluation for FeTaQA

Apart from human evaluation, we analyze 100 sample outputs from the FeTaQA dataset using the RAGAS evaluator (Gradients, 2023), a framework specifically designed for evaluating Retrieval Augmented Generation (RAG) pipelines. The RAGAS evaluation results is listed in Table 15. RAGAS assesses several key aspects: (1) Faithfulness: Evaluates the factual consistency of the answer concerning the context based on the question, (2) Context Precision: Measures the relevance of the retrieved context to the question, reflecting the quality of the retrieval pipeline, (3) Answer Relevancy: Assesses the relevance of the answer to the question, and (4) Context Recall: Measures the retriever's capability to retrieve all essential information required to answer the question. The performance of TabSQLify is comparable to that of Table-CoT-chatgpt, which utilized the full table context. Additionally, the RAGAS evaluation shows a similar trend to our human evaluation.

Model	Precision	Recall	Relevancy	Faithfulness
TableCoT-chatgpt	0.44	0.94	0.94	0.73
TabSQLify$_{col}$	0.42	0.92	0.93	0.67
TabSQLify$_{row}$	0.45	0.97	0.94	0.73
TabSQLify$_{col+row}$	0.44	0.94	0.94	0.72

Table 15: RAGAS evaluation results on FeTaQA.