Query Plan Generation for Table Question Answering

Ivan A. Poddubnyi
Matrosov Institute for System Dynamics and Control Theory of SB RAS
Irkutsk, Russia
ISP RAS Research Center for Trusted Artificial Intelligence
Moscow, Russia
poddubnyyiv@yandex.ru

Nikita O. Dorodnykh
Matrosov Institute for System Dynamics and Control Theory of SB RAS
Irkutsk, Russia
ISP RAS Research Center for Trusted Artificial Intelligence
Moscow, Russia
nikidorny@icc.ru

ABSTRACT

Tabular data are widely used in various fields, but their automatic interpretation remains a challenge due to their structural heterogeneity and lack of explicit semantics. Solving the task of automatic Table Question Answering requires overcoming a number of challenges, in particular, related to the low accuracy of current methods and errors in numerical reasoning. This paper proposes a new approach based on pre-training a BART language model to generate computational graphs. The graphs are analogous to the execution plans of SQL queries in relational database management systems. The approach replaces the direct execution of SQL queries by query plan generation, which reduces computational complexity and minimizes errors associated with implicit computations. Experiments are performed on a WikiSQL set containing 80 thousand examples. Pre-training of the model is performed on 3.8 million pairs of SQL queries and linearized tables, followed by fine-tuning. The results demonstrate that the model achieves 95.1% denotation accuracy on the test sample, outperforming the baseline TAPEX solution. This opens up new opportunities for building table-based question-answering intelligent systems combining high performance and semantic consistency.

VLDB Workshop Reference Format:
Ivan A. Poddubnyi and Nikita O. Dorodnykh. Query Plan Generation for Table Question Answering. VLDB 2025 Workshop: Tabular Data Analysis (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/YRL-AIDA/QueryPlanGenerator.git.

1 INTRODUCTION

Tabular data remains one of the key means for structured presentation of information in various fields such as finance, science, and management. Their versatility and compactness make them indispensable for data storage and analysis [2]. However, the heterogeneous structure of the tables, the lack of explicit semantics, and the difficulties in automatic interpretation limit their practical use in different intelligent systems [7, 22].

Proceedings of the VLDB Endowment. ISSN 2150-8097.

One way to utilize the valuable information in the tables is to develop ideas in the direction known as Table Question Answering (TQA) [6, 13] - automatic generation of answers to questions based on data presented in tabular form. TQA addresses the user's question in natural or logical language and aims to provide accurate and correct answers through understanding and reasoning of tabular data. Current TQA methods based on semantic parsing [4, 5, 8, 9, 13–16, 21] or generative models [3, 11, 12, 17, 24] show progress but face a number of difficulties even in test problems due to errors in numerical reasoning, context dependencies, and query case.

In this paper, we propose a new TQA approach based on pre-training a language model on computational graphs. Unlike existing solutions, the proposed approach replaces direct query execution by generating a query plan graph similar to the execution plan in relational database management systems. This reduces computational complexity, improves accuracy, and minimizes errors associated with implicit computations. Experiments conducted on the WikiSQL test dataset [19] confirm the effectiveness of our approach: After fine-tuning, the model achieves 95.1% denotation accuracy, demonstrating case-resilience of SQL commands. Thus, the main contributions include the following:

- We proposed a novel way to generate answers through computational graphs, providing high accuracy and interpretability.
- We prepared a large-scale pre-training dataset customized for SQL query syntax.
- Experiments on the WikiSQL dataset show that the proposed approach outperforms the baseline solution, which confirms its effectiveness.

2 RELATED WORK

One of the directions in the field of automatic table understanding is Table Question Answering (TQA) [6, 13]. TQA is at the intersection of areas such as natural language processing and machine learning and is a specialization of the scientific direction on the study of question-answering systems.

We can distinguish the following models and methods to solve the TQA problem [6]:

(1) **Semantic parsing methods** convert natural language questions into logical forms (e.g., SQL) [20] , which are then executed on source tables to obtain answers. These approaches can be weakly supervised [9, 10, 14, 16, 23] or fully supervised [4, 13, 15, 21], depending on how the logical forms are constructed. TAPEX [8] has become a state-of-the-art (SOTA) solution by pre-training language models on

tabular data and addressing data scarcity with a large-scale, high-quality synthetic SQL corpus. Consequently, TAPEX is widely used as a baseline for further fine-tuning, including in the OmniTab approach [5]. OmniTab uses natural (Wikipedia, WikiTableQuestion (WTQ) dataset[1]) and synthetic (SQL2NL, SQuALL datasets[2]) data for training, using an end-to-end response generation framework.

(2) **Non-semantic-parsing-based methods** generate answers directly without constructing intermediate logical forms. Generative models, such as FeTaQA [12], use end-to-end pre-trained language models to encode questions and linearized tables, producing free-form answers. A graph-based generative approach [11] represents tables as graphs (with columns, rows, and cells as nodes), encodes them via graph neural networks, and generates responses using a transformer-based decoder. Alternatively, some methods extract the answer spans directly from linearized tables, as in TaPaS [3] and TAT-QA [24].

Over the past year, significant progress has been made in solving the TQA problem, which is associated with the use of large language models. In particular, the Chain-of-Table method [17] adapts the Chain-of-Thought reasoning to tabular data by decomposing questions and iteratively transforming tables using large language models to derive answers. The Tool-Augmented Reasoning framework for Tables (TART) approach [18] integrates LLM (CodeLlama) with specialized tools to apply precise numerical reasoning.

Our approach uses an intermediate representation in the logical form between the question and the answer, like classical semantic parsing methods (e.g., TAPEX or OmniTab). However, instead of direct SQL, it is a computational graph (query plan) generated by the language model. Direct SQL execution is prone to implementation errors, command case sensitivity, and the complexities of implicit computations (especially numerical ones). Query plan generation replaces SQL execution.

3 TABLE PRE-TRAINING VIA PLAN GENERATION

3.1 Problem Statement

The TQA aims to generate an answer A for a given question Q on a table T.

The source (input) tables for the proposed approach are relational tables in the third normal form (3NF). This table represents a set of similar entities in the form of a relation (a subset of the Cartesian product of N data domains), where:

- **Attribute (column name)** is a name of the data domain in the relation schema.
- **Metadata (schema)** is an ordered set of N attributes of a relation table.
- **Tuple (record)** is an ordered set of N atomic values, one for each attribute of a relation.
- **Data (record set)** is a set of tuples of a relation table.

In this case, the first row of the source table is a header containing the names of attributes (columns), and the values of the column cells have the same entity types and data types.

Questions are usually formulated in natural language or can be presented using intermediate forms, for example, in the form of an SQL query. Thus, the input data for our approach is a linearized table T', obtained on the basis of expanding a source table T into a sequence of tokens, as well as a question Q, presented as an SQL query. Then let's formalize the problem as follows:

$$f(T', Q) \rightarrow A \tag{1}$$

where A is an answer that satisfies the following conditions:

- compliance with the question's intent (semantic consistency);
- logical consistency with tabular data;
- the answer can be obtained through operations acceptable for the structure and data types of T' (e.g., summing numeric columns, filtering by condition).

Following the assumption made in [8], the core idea of our approach is that if a language model can be trained to accurately "execute" SQL queries and produce the correct results, then it should have a deep understanding of tables. In this study, we propose the approach for pre-training a language model to generate a linearized computational graph. Such a graph resembles the query plan produced by a relational DBMS when executing SQL queries over a table. In this case, an accurately generated query plan is equivalent to the result obtained from its execution in the computational module. This plan generation does not require the model to perform table operations to derive the result, as is done, for example, in the Chain-of-Table method [17]. This fact conceptually simplifies the generation task, thereby improving model performance, since all transformations over the table can be performed by the software module, especially mathematical operations. Thus, the answer A is a query plan. Next, let's take a closer look at the dataset preparation and the specifics of the language model's pre-training.

3.2 Pre-training Task

Figure 1 shows the general outline of the proposed language model pre-training approach for generating a linearized computational graph (query plan), consists of the following main steps:

(1) An executable SQL query Q is concatenated with a linearized table T' and passed to the input of the model.
(2) An SQL query Q' is executed on the source table T, which is an SQL query Q with the addition of the analysis prefix "EXPLAIN". The result is a plan P for executing an SQL query Q.
(3) The resulting plan P is transformed into a linearized graph plan P'.
(4) Training is performed, comparing the model-generated plan MP with the resulting linearized graph plan P'.

A large language model - BART-large[3] with an appropriate tokenizer is used for pre-training. The BART architecture was chosen

[1] https://github.com/ppasupat/WikiTableQuestions
[2] https://github.com/tzshi/squall
[3] https://huggingface.co/facebook/bart-large

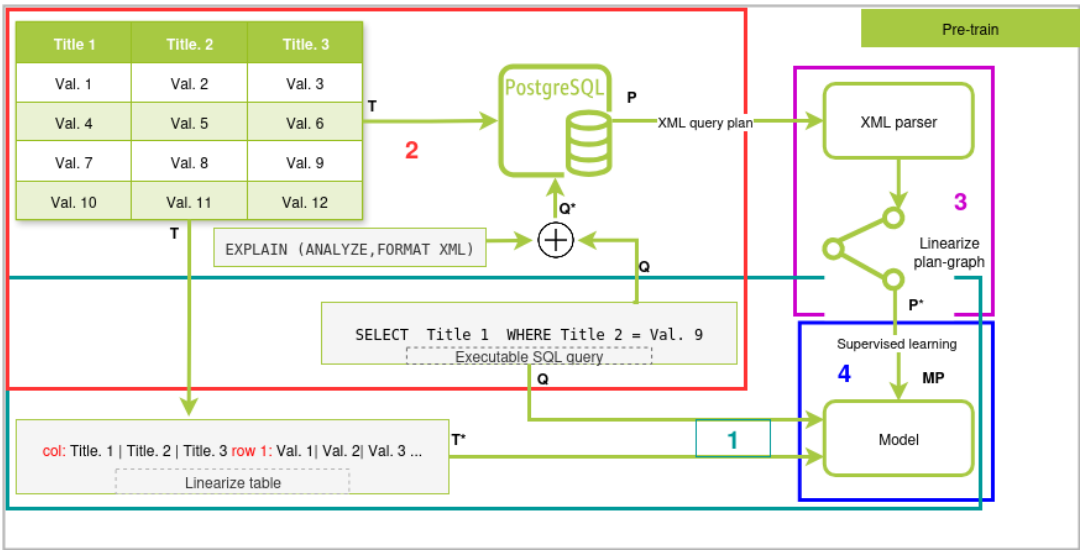

Figure 1: The general outline of the proposed approach pre-trains a language model to generate a linearized computational graph (query plan). It consists of: (1) An executable SQL query (Q) is concatenated with a linearized table (T') and passed to the input of the model. (2) An SQL query (Q') is executed on the source table (T), which is a SQL query (Q) with the addition of the analysis prefix "EXPLAIN". The result is a plan (P) for executing SQL query (Q). (3) The resulting plan (P) is converted into a linearized graph plan (P'). (4) Training is performed to compare the model-generated plan (MP) with the obtained linearized graph plan (P').

as the foundation for our approach due to its unique hybrid design, which optimally aligns with the demands of computational graph (query plan) generation. Moreover, BART underpins SOTA TQA models like TAPEX, demonstrating superior table reasoning capabilities. Our approach extends this strength to plan generation. During training, the cross-entropy function is used as the loss function together with L2 regularization.

A concatenation of an SQL query Q and a linearized table T' is fed to the input of the model. During supervised training, the model learns to generate a linearized query plan MP that was obtained by executing Q over the table T'.

After pre-training, the resulting model can be fine-tuned to fit different TQA datasets (e.g., WTQ, SQuALL, WikiSQL[4], FeTaQA [12], TabFact [1]).

3.3 Pre-training Dataset

A large-scale TAPEX dataset [8] was taken as a basis to perform model pre-training. This dataset consists of five million pairs representing a question and an answer. The question consists of an SQL query and a linearized table in the format: "***col:*** *col1 col2 ...* ***row1:*** *r1 r2 ...* ***row2:*** *...*". The response can be single or contain a serialized array.

At the stage of data pre-processing, the corresponding SQL query and table are extracted from the question, which then underwent the delinearization procedure. At the same time, the table column names and their mentions in the SQL query are converted to conventional notations of the following type: "**col_i**", where i is a column index in a table for further parsing through the SQLGlot library[5]. The question text is hashed using the hashlib.md5 library[6], and from the obtained hash by adding a special character "f" to the beginning of the sequence. Thus, a unique name is formed to save the table to the database for further processing. This approach avoids conflicts and delays during parallel processing of the dataset. The data types for all columns of the table are determined, and a reference to the obtained table name is inserted into the SQL query via SQLGlot, after which error correction is performed. The obtained table is loaded into the PostgreSQL DBMS. A corrected SQL query with the "EXPLAIN (ANALYZE, FORMAT XML)" prefix are executed on the loaded table, which allows analyzing how exactly the query is executed. The obtained XML string with the query plan is converted into a simplified query graph plan, where the optimization information, which is excessive for PostgreSQL, is filtered out. Such a graph can consist of one or several vertices (nodes). In this case, each node has the prefix "NODE" and the following attributes:

- *Type* is a type of operation such as scanning, grouping, etc.
- *i_width(i-w)* is a node width index.
- *i_depth(i-d)* is an inverted depth index (leaf to root).
- *Parent* is a parent node name.
- *Args* is the operation arguments in the format: "*argument name: value*", separated by symbol "|".

To ensure better sequential generation by the model, the graph vertices are arranged in *i_width* order sequentially from leaf to root. Figure 2 shows an example fragment of the trained dataset. In the

[4]https://github.com/salesforce/WikiSQL

[5]https://github.com/tobymao/sqlglot

[6]https://docs.python.org/3/library/hashlib.html

end, the updated table is linearized similarly to the original table and merged with the linearized graph into the input question.

Before feeding the model, the data are filtered by exceeding the maximum input sequence length in tokens for the target model. In particular, a large BART model is used for this approach, where the maximum sequence length is 1024 tokens. Thus, 24% of the original amount of data is lost after pre-processing and filtering. Figure 3 shows the percentage distribution of pairs in terms of response complexity (query plan). Figure 4 shows the occurrence of different SQL commands in the set.

4 EXPERIMENTS

4.1 Implementation Details

The pre-training on the query plan generation task was performed on the compute cluster "Akademik V.M. Matrosov"[7] on the basis of the Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of the Russian Academy of Sciences (ISDCT SB RAS). We used two NVIDIA A100 80 GB PCIe GPUs with batch of 16 per card and a gradient accumulation factor of 8, giving a total batch of 256. Thus, seed = 544, learning rate = $3e^{-5}$, gradient decay = $1e^{-4}$. The checkpoints were stored at the end of each epoch. Training in the trained TAPEX dataset lasted 3 epochs and took 15 days. This training resulted in a Query Plan Generator model that is capable of generating a query plan.

The fine-tuning on the query plan generation task on the WikiSQL and SQUALL datasets was also performed on the two NVIDIA A100 80 GB PCIe GPUs with a batch of 16 per card and a gradient accumulation factor of 4, giving a total batch of 128. Training lasted 20 epochs, seed = 544, learning rate = $3e^{-5}$, gradient decay = $1e^{-3}$. The checkpoints were stored at the end of each epoch. This fine-tuning of the model took 24 hours for WikiSQL and 1.8 hours for SQUALL.

4.2 Evaluation Datasets

WikiSQL is chosen as the main dataset to test the performance of the proposed approach. SQL queries in this dataset include simple queries such as SELECT, FROM, WHERE. The statistics on the SQL commands used for the WikiSQL dataset is presented in Figure 5.

WikiSQL includes training (*train*), validation (*dev*) and test samples (*test*). After data pre-processing and filtering similar to that described above, the following was lost 19%, 29%, 30% for train, dev, and test, respectively.

SQUALL is selected as an additional dataset in evaluating the performance of the proposed approach. After data pre-processing, the dataset was randomly divided into train and test sets, where the test set constituted 20% of the dataset. The resulting set, as can be seen from the Figure 6, has a similar structure to the original set.

4.3 Evaluation Metrics

Denotation Accuracy (DA) [8] is used as the main evaluation metric, which means that the answer provided by the system is considered accurate only when it completely matches the reference content without errors and assumptions. The metric is defined using the following formula:

$$DA = \frac{P}{N} \tag{2}$$

where P is the number of exactly matching question-answer pairs; N is the total number of question-answer pairs.

4.4 Main Results and Discussion

In this paper, the TAPEX approach [8] is chosen as a baseline solution for the performance comparison. In this paper, the proposed approach focuses only on SQL query execution as a first step in developing a pre-training method similar to TAPEX. The comparison is made with the TAPEX SQL Executor model. This version of TAPEX is trained on SQL query execution without fine-tuning to answer questions asked in natural language. An experiment is conducted to show how the basic pre-trained language models (Query Plan Generator and TAPEX sql executor) are able to handle SQL query answering without any pre-training. These models were then fine-tuned on the WikiSQL and SQUALL datasets. In addition, the experiments investigated the effect of SQL commands written in uppercase letters on the test results (the base models of the query plan generator and SQL TAPEX executor were trained on sequences in which SQL commands are represented only by lowercase letters). For this purpose, the base models of the query plan generator and TAPEX were fine-tuned on WiKiSQL in two modes: with commands written in lowercase and with commands written in uppercase. The results of the experimental evaluation are presented in Table 1 and Table 2.

Table 1: Comparative analysis of the results of the experimental evaluation on the WikiSQL dataset for the Query Plan Generator against the TAPEX SQL executor across pre-training and fine-tuning stages. In our case, "*up*" means the allocation of SQL commands to uppercase during testing; "*tr*" means that the model was fine-tuned on data with commands corrected to uppercase. Critical findings: (1) After fine-tuning, our model achieves SOTA performance (95.1% on DA test); (2) Superior case-robustness (DA test on pre-training stage: about 5% with uppercase SQL vs TAPEX's 20% sensitivity).

Model	*DA dev*	*DA test*
Pre-training stage		
TAPEX sql-executor	40.4	41.0
TAPEX sql-executor (*up*)	54.8	61.1
Query Plan Generator	37.4	68.1
Query Plan Generator (*up*)	37.1	63.4
Fine-tuning stage		
TAPEX sql-executor	73.8	76.0
TAPEX sql-executor (*up*)	78.5	80.0
TAPEX sql-executor (*tr*)	81.9	83.5
TAPEX sql-executor (*tr+up*)	81.5	83.2
Query Plan Generator (*tr*)	**95.0**	**95.1**
Query Plan Generator (*tr+up*)	94.1	94.2

[7] https://hpc.icc.ru

NODE	Seq Scan	0|	0	Query	Relation-Name : f649a345f49651bbefbd5c8e555b812cd	Alias : f649a345f49651bbefbd5c8e555b812cd	Filter : (col_2 = 'cork'::text)
NODE	Type	i-w	i-d	Parent	Arg1	Arg2	Arg3

Figure 2: An example of the structured representation of a linearized generated query graph as answer, showcasing node attributes (Type, width, depth, Parent, Args) in a sequential format optimized for language model decoding. This human-readable yet machine-parsable format enables precise computational equivalence to SQL execution while enhancing interpretability as a critical advantage over opaque SQL execution methods.

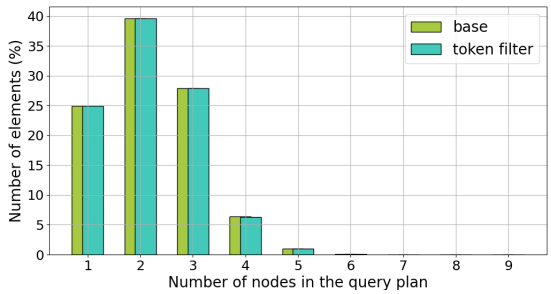

Figure 3: Statistics on the number of nodes in the query plan for the original TAPEX dataset (base) and pre-processed TAPEX dataset after filtering by exceeding the length of the tokenized input sequence (token filter). The node count distribution across query plans in the processed TAPEX dataset and after sequence-length filtering is the same. The right-skewed distribution confirms the prevalence of moderately complex operations (four or more nodes).

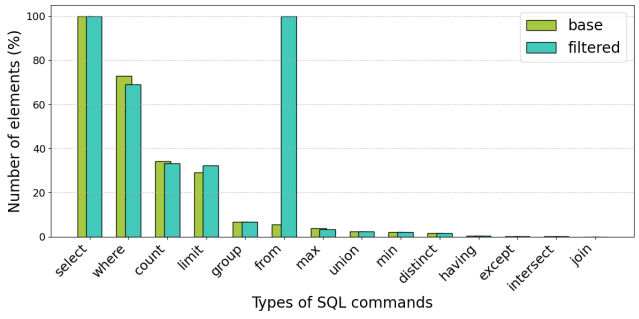

Figure 4: Statistics on SQL commands used in the original TAPEX dataset (base) and pre-processed TAPEX dataset after filtering by exceeding the length of the tokenized input se- quence (filtered). SELECT/FROM/WHERE commands dominate (about 80% occurrence), while complex operations (JOIN/GROUP BY/ORDER BY) maintain proportional representation post-filtering validating the dataset's suitability for query plan generation training.

As shown in Table 1, the Query Plan Generator model is inferior to the TAPEX model on the validation set by 3%, but outperforms it on the test set by 27.1%. After fine-tuning the Query Plan Generator model and TAPEX sql-executor on WikiSQL still on the SQL query execution task, but with half the batch size, the model results have

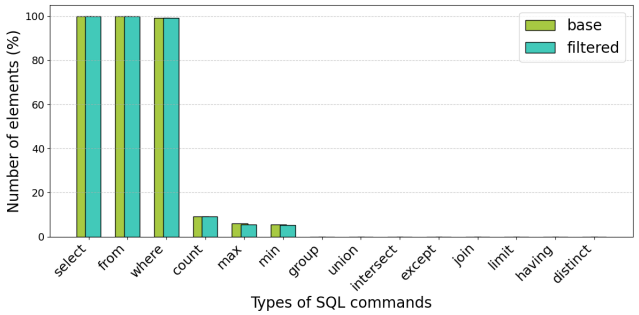

Figure 5: Statistics on SQL commands used in the original WikiSQL dataset (base) and pre-processed WikiSQL dataset after filtering by exceeding the length of the tokenized input sequence (filtered). We highlights the composition of SQL operations in the WikiSQL evaluation set, dominated by fundamental commands (about 97-100% for SELECT/WHERE/FROM). The limited occurrence of aggregations (about 9-10% for COUNT/MAX/MIN) establishes a rigorous testbed for core TQA capabilities while aligning with real-world query patterns.

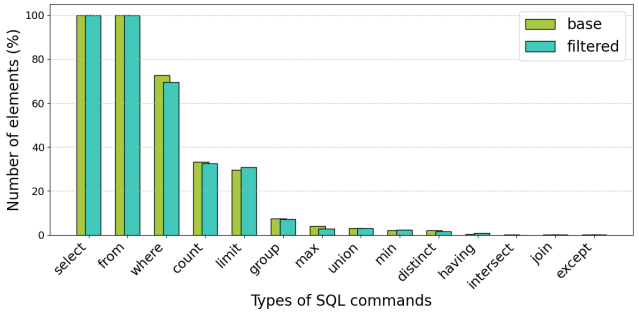

Figure 6: Statistics on SQL commands used in the SQUALL dataset (base) and pre-processed SQUALL dataset after filtering by exceeding the length of the tokenized input se- quence (filtered). We illustrates the retention of SQL operation diversity in the SQuALL dataset after sequence-length filtering. Essential SQL commands (SELECT/FROM/GROUP BY) maintain near-complete representation (about 98%). Other operations (e.g., WHERE, COUNT, MAX) show minimal attrition, preserving SQuALL's signature challenge of nested queries and multi-step reasoning. This robustness validates the dataset's utility for testing advanced TQA capabilities under realistic token constraints.

Table 2: Comparative analysis of the results of the experimental evaluation on the SQuALL dataset for the Query Plan Generator against the TAPEX SQL executor across pre-training and fine-tuning stages. In this case, contains an SQL commands without taking into account the uppercase. Results confirm: (1) Pre-trained TAPEX excels in zero-shot settings (8.2% vs TAPEX's 32.3%); (2) After fine-tuning, our model achieves near-parity (72.8% vs TAPEX's 73.9%), demonstrating its rapid adaptability to diverse SQL schemas despite architectural differences in plan generation.

Model	*DA test*
TAPEX sql-executor	32.3
Query Plan Generator	8.2
Fine-tuned TAPEX sql-executor	**73.9**
Fine-tuned Query Plan Generator	72.8

improved significantly and now the gain relative to TAPEX is +13.5% on the validation set and +11.9% on the test set. Also, various modes of fine-tuning and evaluating have shown that the allocation of SQL commands leads to an improvement in the accuracy of the model by an average of 3%. The results of the experiments for the base models in Table 1 have shown that the sensitivity to the case of SQL commands for the Query Plan Generator without fine-tuning is more than 3 times lower than that of TAPEX. At the same time, further fine-tuning on the allocated SQL commands leads to better and less sensitive results to the spelling of SQL commands.

Additionally, we fine-tuned and tested the models on the SQUALL dataset. As can be seen in the Table 2, the pre-trained TAPEX outperforms in the zero-shot setting (8.2% vs. 32.3% for TAPEX), but after fine-tuning, our model reaches near parity (72.8% vs. 73.9% for TAPEX), demonstrating its fast adaptability to different SQL schemas despite architectural differences in plan generation.

5 CONCLUSION

In this paper, we propose a novel approach for automatic question answering on tabular data based on pre-training a BART language model for generating computational graphs of SQL queries. Experiments show that the model achieves 95.1% denotation accuracy on a WikiSQL test sample after fine-tuning, outperforming baseline solutions (TAPEX) and achieving parity results on the SQUALL dataset. The key advantage of the approach is the reduced dependence on the SQL command case and the ability to integrate program modules for accurate computations, which is critical for numeric data. The developed approach opens new opportunities for creating intelligent table analysis systems combining high accuracy and interpretability of results.

In the future, we plan to compare the Query Plan Generator against SQL based benchmark systems. We also plan to implement a computational software module for the query plan graph and fine-tuning on well-known datasets such as WTQ and TabFact with natural language questions and comparison with advanced models in this context. In addition, we plan to include heterogeneous tables with different structural layouts in the processing.

ACKNOWLEDGMENTS

This work was supported by a grant, provided by the Ministry of Economic Development of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000C313925-P4G0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated June 20, 2025 No. 139-15-2025-011.

REFERENCES

[1] 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. In *International Conference on Learning Representations (ICLR)*. Addis Ababa, Ethiopia.

[2] Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. 2022. Table Pre-training: A Survey on Model Architectures, Pre-training Objectives, and Downstream Tasks. In *Proceedings of the 31th International Joint Conference on Artificial Intelligence*. 5426–5435.

[3] 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*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[4] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069* (2019).

[5] Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. OmniTab: Pretraining with Natural and Synthetic Data for Few-shot Table-based Question Answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 932–942. https://doi.org/10.18653/v1/2022.naacl-main.68

[6] Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*. Springer, 174–186.

[7] Jixiong Liu, Yoan Chabot, Raphaël Troncy, Viet-Phi Huynh, Thomas Labbé, and Pierre Monnin. 2023. From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods. *Journal of Web Semantics* 76 (2023), 100761. https://doi.org/10.1016/j.websem.2022.100761

[8] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. TAPEX: Table pre-training via learning a neural SQL executor. *arXiv preprint arXiv:2107.07653* (2021).

[9] Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A Discrete Hard EM Approach for Weakly Supervised Question Answering. 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)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 2851–2864. https://doi.org/10.18653/v1/D19-1284

[10] Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen-tau Yih. 2018. Policy Shaping and Generalized Update Equations for Semantic Parsing from Denotations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 2442–2452. https://doi.org/10.18653/v1/D18-1266

[11] Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia, and Yasemin Altun. 2019. Answering Conversational Questions on Structured Data without Logical Forms. 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)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 5902–5910. https://doi.org/10.18653/v1/D19-1603

[12] 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 (2022), 35–49. https://doi.org/10.1162/tacl_a_00446

[13] Abhyankar Nikhil, Gupta Vivek, Roth Dan, and K. Reddy Chandan. 2025. H-STAR: LLM-driven Hybrid SQL-Text Adaptive Reasoning on Tables. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*. 8841–8863.

[14] Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic Parsing with Syntax- and Table-Aware SQL Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, Melbourne, Australia, 361–372. https://doi.org/10.18653/v1/P18-1034

[15] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 7567–7578. https://doi.org/10.18653/v1/2020.acl-main.677

[16] 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)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3774–3785. https://doi.org/10.18653/v1/D19-1391

[17] 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* (2024).

[18] Lu Xinyuan, Pan Liangming, Ma Yubo, Nakov Preslavn, and Kan Min-Yen. 2025. TART: An Open-Source Tool-Augmented Framework for Explainable Table-based Reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, Chiruzzo Luis, Ritter Alan, and Wang Lu (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico, 4323–4339. https://doi.org/10.18653/v1/2025.findings-naacl.244

[19] Semih Yavuz, Izzeddin Gür, Yu Su, and Xifeng Yan. 2018. What it takes to achieve 100% condition accuracy on WikiSQL. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1702–1711.

[20] Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables with natural language. *arXiv preprint arXiv:1512.00965* (2015).

[21] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 588–594. https://doi.org/10.18653/v1/N18-2093

[22] Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 2 (2020), 1–35.

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

[24] 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)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 3277–3287. https://doi.org/10.18653/v1/2021.acl-long.254