

# TableTalk: Question Answering on Tabular Data

## T<sup>3</sup> : The Table Turners

Anurag Joardar  
anuragjoardar10@example.com

Biswajit Rana  
ranabiswajit911@gmail.com

Rakesh Nemu  
rakeshnemu237@gmail.com

November 24, 2024

### Abstract

In an era where data is central to decision-making, the ability to interact with structured data seamlessly and intuitively is more critical than ever. The project " **TableTalk : Question Answering on Tabular Data**" addresses the challenge of enabling non-technical users to query structured data stored in tabular formats like databases and spreadsheets. Traditional methods require expertise in languages like SQL, which poses a barrier for quick and effective interaction with data. This project develops a model using a fine-tuned Small-T5 large language model (LLM) to translate natural language queries into SQL. By leveraging a full fine-tuning approach and training on three text-to-SQL datasets, the model generates SQL queries that interact with MySQL databases, providing precise results. We have also created an end-to-end application using Streamlit, enabling users to interact seamlessly with the system in a user-friendly interface. This method empowers users to query and understand data intuitively.

## 1 Introduction

Pre-trained language models like BERT and BART have achieved remarkable success in understanding free-form natural language (NL) tasks by learning from vast unstructured textual data. Inspired by this, researchers have focused on structured tabular data. However, unlike unstructured NL data, tabular data carries significant structural information, which poses challenges for applying existing methods designed for unstructured data. These efforts aim to adapt techniques to effectively handle the unique characteristics of tabular data.

In this paper, **TableTalk : Question Answering on Tabular Data** focuses on developing a system that enables non-technical users to interact with structured data using natural language queries. The system translates these queries into SQL, retrieves the requested information from a MySQL database, and presents the results in an accessible format. Users provide a natural language query and the name of the relevant table, while the system leverages a fine-tuned Small-T5 large language model (LLM) to generate SQL queries. Additionally, an end-to-end application built with Streamlit ensures that the interaction is intuitive and seamless.

We know that most of Organizations rely heavily on structured data stored in databases and spreadsheets to drive critical decisions. However, accessing this data typically requires technical expertise in querying languages like SQL, which can exclude non-technical users and slow down workflows. This project addresses these challenges by bridging the gap between natural language input and structured data retrieval, making data access more democratic and efficient.

A key component of this project is the full fine-tuning of the Small-T5 model for text-to-SQL tasks. By updating all the parameters of the pre-trained model during fine-tuning, the system achieves a higher degree of specialization for this specific task. This method, while computationally intensive, ensures robust adaptation to the nuances of text-to-SQL conversion. The model was trained on three prominent text-to-SQL datasets, ensuring adaptability across diverse query types and database schemas.

By combining cutting-edge machine learning techniques with a user-friendly interface, this project empowers users from all backgrounds to interact effectively with structured data, driving better decision-making and operational efficiency.

## 2 Literature review

Several advancements have been made in natural language-to-SQL (text-to-SQL) systems:

- **Seq2SQL**, Victor Zhong et al.(2017):[8] A sequence-to-sequence model designed for text-to-SQL tasks, with a focus on generating structured queries from natural language inputs.
- **SQLNet**, Xiaojun Xu et al.(2018):[5] This paper addresses the "order-matters" problem in SQL query synthesis by replacing sequence-to-sequence models with a sketch-based approach where the sketch contains a dependency graph so that one prediction can be done by taking into consideration only the previous predictions that it depends on and a sequence-to-set model, enhancing accuracy in converting natural language to SQL for tabular data.
- **Spider Dataset**, Tao Yu et al.(2018):[6] A complex dataset designed to evaluate cross-domain text-to-SQL performance, promoting the development of models like

RAT-SQL(Relation-Aware Transformer for SQL),a model designed to improve text-to-SQL tasks by incorporating relational awareness. It enhances the ability of models to understand the relationships between tables and columns in a database when translating natural language queries into SQL.

- **T5, Colin Raffel et al.(2019):[4]** Due to its text-to-text framework, pretraining on diverse text tasks, and sequence-to-sequence architecture, which enable it to effectively convert natural language into structured SQL queries.
- **Instruction Tuning for LLM,Shengyu Zhang et al.(2023):[7]** Instruction tuning (IT) is a method for enhancing the performance of large language models (LLMs) by fine-tuning them on datasets consisting of (instruction, output) pairs, bridging the gap between LLMs' next-word prediction objectives and users' task-specific goals. It has been applied across various domains, improving model controllability and alignment with human intent. However, challenges such as dataset size, task diversity, and generalization remain, with ongoing research focusing on addressing these issues. Critics highlight the risk of overfitting and the need for more robust and diverse training strategies to improve IT outcomes.

The main distinction of this project lies in the use of the full fine tuning technique to fine-tune a lightweight Small-T5 model on 3 datasets, optimizing it for text-to-SQL conversion while maintaining computational efficiency. Unlike prior methods, this approach emphasizes adaptability to various schemas, empowering users without requiring technical expertise.

### 3 Proposed methodology

This project develops a pipeline for converting natural language queries into SQL to retrieve data from table.

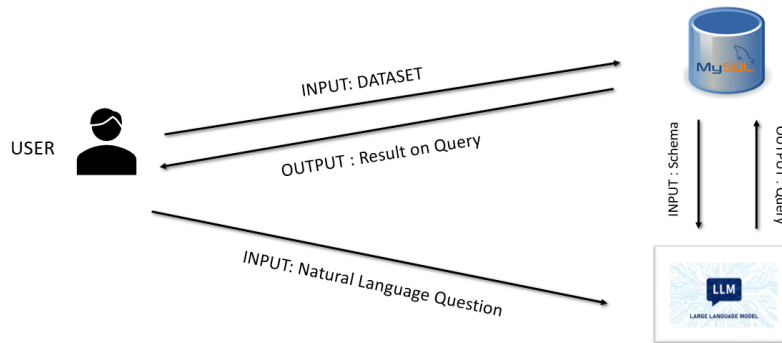


Figure 1: Pipeline Overview

The process includes the following steps:

**Input:**

- **Natural Language Query (NLQ):** A user-friendly question or request.
- **Table Name:** Identifies the relevant table within the database.

**Processing:**

- The NLQ is fed into a fine-tuned Small-T5 LLM.
- Schema information of the target table (columns, data types, etc.) is retrieved from the MySQL database and fed to the LLM alongside the NLQ.
- The LLM generates a corresponding SQL query.

**Execution:**

- The generated SQL query is executed against the MySQL database to fetch results.
- The results are returned to the user in a structured format.

**Fine-Tuning:**

The pretrained LM, **Small-T5** model is fine-tuned using the **full fine tune** technique on 3 standard text-to-SQL datasets named **sql-create-context**, **know\_sql**, **Text-to-sql-v1** from Hugging Face. This approach leverages the pretrained model's generalized knowledge and adapts it to a specific task in a supervised way, using a smaller dataset compared to the one required for pretraining.

For training, we combined these three datasets and applied an instruction-based fine-tuning method, where the model is guided by specific task-oriented instructions. The fine-tuning process was run about 1 epoch with a learning rate of 0.005 and a batch size of 16.

**Application Development:**

An end-to-end application was developed using **Streamlit** to provide a user-friendly interface. The application allows users to input their natural language queries and table names, view the corresponding SQL query generated, and access the results retrieved from the MySQL database seamlessly.

This comprehensive pipeline, combined with a user-facing application, makes the system intuitive and accessible, ensuring effective interaction with tabular data for non-technical users.

## 4 Experimental Setup and Results

### 4.1 Datasets

We utilized four distinct datasets for fine-tuning the T5-small model from Hugging Face repositories:

- **SCC (SQL Create Context):** [2] Dataset focused on SQL query generation from given contexts.
- **TTS (Text-to-SQL):** [3] Dataset providing SQL queries generated from natural language instructions, preprocessed to retain essential columns (question, context, and answer).
- **KS (Know-SQL):**[1] Validation data split into training, validation, and testing subsets.

The final dataset was formed by merging the training, validation, and testing splits from these datasets, resulting in:

- **Train Set:** Combined from 80% of each dataset's training data.
- **Validation Set:** Combined from 10% of each dataset's training data.
- **Test Set:** Combined from the remaining 10%.

### 4.2 Experimental Settings

Fine-tuning was conducted on the **T5-small** model using a Kaggle notebook equipped with a Tesla P100 GPU. The training process lasted for **12 hours**, running for nearly **1 epoch** due to resource constraints.

**Hyperparameters:**

- Model: t5-small
- Optimizer: AdamW (Adam with Weight Decay Regularization: 0.01)
- Batch Size: 16.
- Learning Rate: 0.005.
- Evaluation Steps: 500.



Figure 2: Training Loss Curve

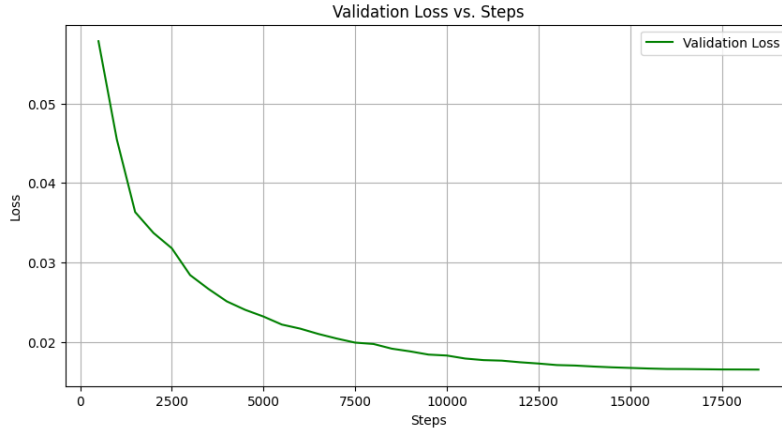


Figure 3: Validation Loss Curve

Further evaluation metrics and qualitative results will be elaborated in the subsequent sections.

### 4.3 Results

The model's performance was evaluated using ROUGE scores, which are widely used for text generation tasks. The results are as follows:

- **ROUGE-1:** 0.9964
- **ROUGE-2:** 0.9961

- **ROUGE-L:** 0.9964
- **ROUGE-Lsum:** 0.9964

The high ROUGE scores indicate the model's ability to generate SQL queries with a high degree of accuracy and semantic relevance. Specifically, the ROUGE-1 and ROUGE-2 scores reflect strong unigram and bigram overlap, while the ROUGE-L and ROUGE-Lsum scores highlight the model's ability to generate syntactically correct and contextually meaningful SQL queries. The model was tested on a dataset of 39,023 instances, ensuring statistically significant results.

Overall, the model demonstrates excellent performance in generating accurate SQL queries from natural language inputs, making it suitable for real-world applications.

#### 4.4 Time complexity

The time complexity of the model during inference depends on the size of the input and the maximum number of tokens generated. The main factors influencing the time complexity are:

- **Tokenization:** Converting the input text into token IDs has a linear time complexity,  $\mathcal{O}(n)$ , where  $n$  is the length of the input text.
- **Model Inference:** Generating tokens is influenced by the number of tokens in the input sequence ( $n$ ) and the maximum number of tokens to be generated ( $m$ ). Transformer-based models, like T5-small, have a time complexity of  $\mathcal{O}(n^2d + mnd)$  per layer, where  $d$  is the model's embedding size. The quadratic dependence on  $n$  arises from the self-attention mechanism.
- **Decoding:** The output is generated one token at a time, resulting in an additional  $\mathcal{O}(m)$  complexity.

Inference for a single input, including context and a question, took approximately 2.3 seconds on an NVIDIA GTX 1650 GPU. This time accounts for tokenization, model inference, and decoding steps. The performance aligns with transformer-based architectures when handling long context sequences and moderate output lengths. Optimization strategies such as mixed-precision inference and beam search adjustments could further reduce latency.

## 5 Summary

**TableTalk : Question Answering on Tabular Data** project bridges the gap between non-technical users and structured data by leveraging a fine-tuned Small-T5 LLM for

natural language to SQL conversion. The project achieves significant results on multiple datasets while maintaining computational efficiency through the full fine-tuning technique. By enabling intuitive and accessible interaction with tabular data, this system has the potential to democratize data querying, empowering users across various domains. We have also created an end-to-end application using Streamlit, providing a user-friendly interface for seamless interaction with the system.

## 6 References

Some references related to your project. I am attaching a *nlp\_reference.bib* file and you can use that. You have to add your *bibtex* according to your references like conference, journal, book, etc.. Your reference section should be some like bellow:

### References

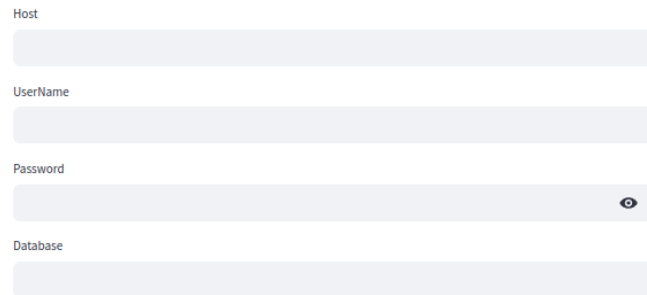
- [1] Hugging Face. Know sql dataset, 2024.
- [2] Hugging Face. Sql create context dataset, 2024.
- [3] Hugging Face. Text-to-sql v1 dataset, 2024.
- [4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.
- [5] Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning, 2017.
- [6] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *CoRR*, abs/1809.08887, 2018.
- [7] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey, 2024.
- [8] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.



## 7 Inference

A short demonstration of the project on a real environment.

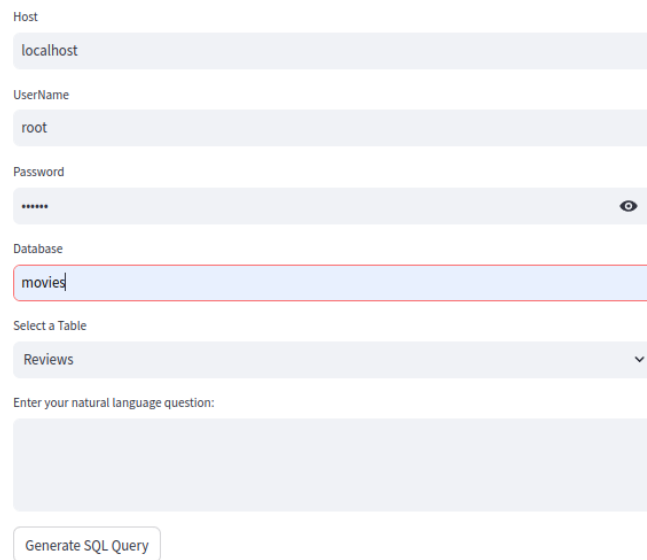
### SQL Query Generator and Executor



A form titled "SQL Query Generator and Executor" with four input fields: "Host", "UserName", "Password", and "Database". Each field is a light blue rounded rectangle. The "Password" field has a small eye icon on the right side.

Figure 4: User Info Input

### SQL Query Generator and Executor



A form titled "SQL Query Generator and Executor" with several input fields and a button. The fields are: "Host" (with "localhost" entered), "UserName" (with "root" entered), "Password" (with "\*\*\*\*\*" entered and an eye icon), "Database" (with "movies" entered and highlighted with a red border), "Select a Table" (a dropdown menu with "Reviews" selected), and "Enter your natural language question:" (a large text area). At the bottom is a button labeled "Generate SQL Query".

Figure 5: Database Selection

## SQL Query Generator and Executor

Host  
localhost

UserName  
root

Password  
\*\*\*\*\*

Database  
movies

Select a Table  
Reviews

- Reviews
- award\_roles
- awards
- movies
- stardetails
- starphoto
- starsin
- temp1

Figure 6: Table Selection

Select a Table

awards

Enter your natural language question:

show all data from the table where award year is 2002

Generate SQL Query

Schema

"Reviews" "review" varchar, "title" varchar, "user" varchar primary key: "title","user" [SEP]

"award\_roles" "award\_details" varchar, "award\_name" varchar, "role" varchar primary key:

"award\_name" [SEP] "awards" "award\_name" varchar, "award\_year" int, "awarded\_to\_movie"

varchar, "starname" varchar primary key: "award\_name","award\_year" [SEP] "movies" "genre"

varchar, "length" int, "title" varchar primary key: "title" [SEP] "stardetails" "dob" date, "gender"

char, "starname" varchar primary key: "starname" [SEP] "starphoto" "Photo" longblob, "starname"

varchar primary key: "starname" [SEP] "starsin" "role" varchar, "starname" varchar, "title" varchar

primary key: "role","starname","title" [SEP] "temp1" "genre" varchar, "starname" varchar [SEP]

Generated SQL Query

SELECT FROM awards WHERE award\_year = 2002;

Query Results

	award_name	award_year	awarded_to_movie	starname
0	BEST ACTOR	2,002	LAGAAN	AMIR KHAN
1	BEST ACTRESS	2,002	LAGAAN	GRACY SINGH
2	BEST DIRECTOR	2,002	LAGAAN	ASHUTOSH GOWARIKER
3	BEST FILM	2,002	LAGAAN	None
4	BEST LYRICIST	2,002	LAGAAN	JAVED AKHTAR
5	BEST MUSIC DIRECTOR	2,002	LAGAAN	A.R. RAHMAN
6	BEST PLAYBACK SINGER FEMALE	2,002	LAGAAN	ALKA YAGNIK
7	BEST PLAYBACK SINGER MALE	2,002	LAGAAN	UDIT NARAYAN

Figure 7: SQL Query: 1

Select a Table

awards

Enter your natural language question:

show all data from the table where award year is 2002 and starname is amir khan

1

Generate SQL Query

Schema

"Reviews" "review" varchar, "title" varchar, "user" varchar primary key: "title","user" [SEP]  
"award\_roles" "award\_details" varchar, "award\_name" varchar, "role" varchar primary key:  
"award\_name" [SEP] "awards" "award\_name" varchar, "award\_year" int, "awarded\_to\_movie"  
varchar, "starname" varchar primary key: "award\_name","award\_year" [SEP] "movies" "genre"  
varchar, "length" int, "title" varchar primary key: "title" [SEP] "stardetails" "dob" date, "gender"  
char, "starname" varchar primary key: "starname" [SEP] "starphoto" "Photo" longblob, "starname"  
varchar primary key: "starname" [SEP] "starsin" "role" varchar, "starname" varchar, "title" varchar  
primary key: "role","starname","title" [SEP] "temp1" "genre" varchar, "starname" varchar [SEP]

Generated SQL Query

SELECT \* FROM awards WHERE award\_year = 2002 AND starname = 'Amir Khan';

Query Results

	award_name	award_year	awarded_to_movie	starname
0	BEST ACTOR	2,002	LAGAAN	AMIR KHAN

Figure 8: SQL Query: 2