

# Evaluating the Text-to-SQL Capabilities of Large Language Models

**Akshita Pachauri**

Dept. of Computer Science  
University of Massachusetts  
Lowell, MA 01854, USA

akshita\_pachauri@student.uml.edu

**Sonia Singh**

Dept. of Computer Science  
University of Massachusetts  
Lowell, MA 01854, USA

Sonia\_Singh@student.uml.edu

## Abstract

Generating SQL queries from natural language has long been a popular and useful task. Most of the existing work shows sequence-to-sequence has been widely adapted to directly generate the SQL queries from natural language. Furthermore, Google's T5 model has been proved to be efficient for text to Sql conversion task. In this project, we aim to explore different existing checkpoints of T5 models and analyze on them, as well as implement our model using pre-trained T5-base, T5-large, T5-small, T5-3b. We draw a performance comparison over them. Furthermore, we also make use of evaluation metrics like Bleu score, Exact Match, Component Matching, Evaluation Accuracy, Human Evaluation. We experiment with various state-of-the-art models and the best model achieves only 12.2% exact matching accuracy on a database split setting. This shows that Spider presents a strong challenge for future research. Our dataset and task are publicly available at <https://yale-lily.github.io/spider>.

## 1 Introduction

Generating SQL queries from questions in natural languages helps bridge the gap between a non-technical user and a database systems, as users do not require to understand the database schema and query language syntax. Text to SQL is a system that converts natural language statements to SQL queries. This can help in retrieving information stored in a database by expressing commands in natural language. Text2SQL offers applications in various domains like healthcare, customer support and search engines, which requires elaborating of structured data having information on the text. With the emergence of various popular "natural language to SQL" datasets, such as WikiSQL [1] and Spider [2], many teams have contributed to solving this task. Seq2SQL [1] builds the WikiSQL dataset and proposes a sequence-to-sequence deep learning model on this dataset. They use reinforce-

ment learning method to resolve order matters in SQL conditions. SQLNet [3] eliminates reinforcement learning by introducing dependencies on SQL structures and reflecting the dependencies via a specific attention mechanism. The state-of-the-art model, SyntaxSQLNet [4] is based on SQLNet but generates a more complex syntax tree which even supports recursive SQL queries. In this project we first explored Google's T5 model and further make use of it four checkpoints namely T5-base, T5-large, T5-small, T5-3B as our pre-trained model for the project.

## 2 Related Work

With the burgeoning of Transfer Learning, Deep Learning has achieved many wonders. More specifically, in NLP, with the rise of the Transformer (Vaswani et. al.), various approaches for 'Language Modeling' have arisen wherein we leverage transfer learning by pre-training the model for a very generic task and then fine-tuning it on specific downstream problems.

Seq2SQL [1] has been the first deep neural network based approach to solve this problem. They constructed WikiSQL, one of the largest CS224N Natural Language Processing with Deep Learning, Winter 2019, Stanford natural language query to SQL dataset and developed a deep neural network for translation. They translated SQL based on SQL structures and utilized reinforcement method to predict SQL conditions. They are the first to decompose SQL queries into different parts and predict each part to narrow the output space. Our model also utilizes this and predicts each part of SQL independently. Instead of "sequence-to-sequence" model, SQLNet [3] proposes a "sequence-to-set" model and eliminates reinforcement learning by constructing dependency relationship on SQL queries and developed a specific attention mechanism based on this dependency. They improved the prior state-of-the-art

by 9 points to 13 points. Our model refers to their work on how they encode natural language questions together with column names in the WikiSQL dataset. While most of the models for this task use LSTM network, nowadays, alternative networks are proposed to be effective and efficient in some specific tasks. Typically, Transformer [6], the first sequence transduction model based entirely on attention achieves state-of-the-art in both English- to-German and English-to-French problems as well as shows time efficiency. CNN [7] has been proved to be effective in text classification tasks. In our work, we introduce these two models on SQL translation problem. Our baseline model uses sequence-to-sequence neural machine translation with attention [8]. We simply view SQL as another language and translate English into SQL without any SQL specific optimization.

In 2021 Scholak et al., propose PICARD1[link], a method for constraining auto-regressive decoders of language models through incremental parsing. PICARD helps to find valid output sequences by rejecting inadmissible tokens at each decoding step. On the challenging Spider and CoSQL text-to-SQL translation tasks, we show that PICARD transforms fine-tuned T5 models with passable performance into state-of-the-art solutions. When combined with a large language model called T5-3B, PICARD produced a test accuracy of 75.1

### 3 Approach

#### 3.1 Model

We perform Text 2 SQL wherein we leverage transfer learning by making use of pre-trained model T5, which is trained on many datasets and language translation tasks, and then we fine-tuned it on our specific downstream problems i.e Text to SQL conversion on T5-base, T5-Large, T5-small, and T5-3b.

As Figure 1 illustrates, given a database with multiple tables including foreign keys, our corpus creates and annotates complex questions and SQL queries including different SQL clauses such as joining and nested query. In order to generate the SQL query given the input question, models need to understand both the natural language question and relationships between tables and columns in the database schema.

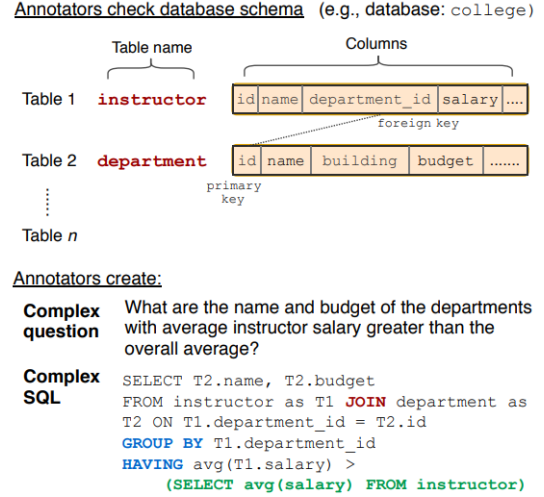


Figure 1: Our corpus annotates complex questions and SQLs. The example contains joining of multiple tables, a GROUP BY component, and a nested query.

## 4 Experiments

### 4.1 Data

Our experiments are mainly focused on Spider (Yu et al., 2018), a large multi-domain and cross-database dataset for text-to-SQL parsing. We train on the 7,000 examples in the Spider training set and evaluate on Spider’s around 1,000 examples development set. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. The dataset comprises of four levels of complex SQL Queries namely easy, medium, hard, very hard. As shown in Figure 2

### 4.2 Experiment Details

We use PyTorch to implement this project Lists are easy to create:

- Data Split : train: 7000 questions and SQL query pairs. dev: 1034 question and SQL query pairs
- Learning rate: We used  $5e^{-5}$  for T5-small, T5-base and  $1e^{-5}$  for T5-large and  $1e^{-4}$ .
- Number of epoch: We set the maximum training epoch as 100.
- Optimization and regularization: We used ADAM to train to regularize.
- Used AutoTokenizer and Accelerate which provides an easy API to make our scripts run with mixed precision on any kind of distributed setting multi-GPUs,TPUs etc.

#### Easy

What is the number of cars with more than 4 cylinders?

```
SELECT COUNT(*)  
FROM cars_data  
WHERE cylinders > 4
```

#### Meidum

For each stadium, how many concerts are there?

```
SELECT T2.name, COUNT(*)  
FROM concert AS T1 JOIN stadium AS T2  
ON T1.stadium_id = T2.stadium_id  
GROUP BY T1.stadium_id
```

#### Hard

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name  
FROM countries AS T1 JOIN continents  
AS T2 ON T1.continent = T2.cont_id  
JOIN car_makers AS T3 ON  
T1.country_id = T3.country  
WHERE T2.continent = 'Europe'  
GROUP BY T1.country_name  
HAVING COUNT(*) >= 3
```

#### Extra Hard

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)  
FROM country  
WHERE name NOT IN  
(SELECT T1.name  
FROM country AS T1 JOIN  
country_language AS T2  
ON T1.code = T2.country_code  
WHERE T2.language = "English"  
AND T2.is_official = "T")
```

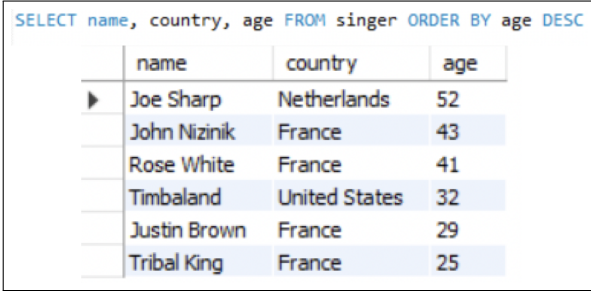
Figure 2: SQL query examples in 4 hardness levels.

### 4.3 Evaluation Method

The metrics used to determine model performance: Bleu Score, Component Matching, Exact Match, and test-suite execution accuracy (Zhong et al., 2020), Human Evaluation. Exact-set-match accuracy compares the predicted and the ground-truth SQL query by parsing both into a normalized data structure. This comparison is not sensitive to literal query values and can decrease under semantic-preserving SQL query rewriting. Execution accuracy compares the results of executing the predicted and ground-truth SQL queries on the database contents shipped with the Spider dataset. This metric is sensitive to literal query values, but suffers from a high false positive rate (Zhong et al., 2020). Lastly, test-suite execution accuracy extends execution to

multiple database instances per SQL schema. The contents of these instances are optimized to lower the number of false positives and to provide the best approximation of semantic accuracy.

Furthermore for Human Evaluation, we not only did human evaluation by just comparing our gold files with our predicted query files but we further created a database using the Database-schemas provided by Spider. And we ran our predicted queries over those databases in order to evaluate if they generate any results. Fig3 shows our results when queried over a singer table.



	name	country	age
▶	Joe Sharp	Netherlands	52
	John Niznik	France	43
	Rose White	France	41
	Timbaland	United States	32
	Justin Brown	France	29
	Tribal King	France	25

Figure 3: SQL predicted query evaluation on "Singer" database

### 4.4 Results

Our findings on the Spider dataset are summarized the performance of all models on our test set includes the accuracy of exact matching in Table 5 and F1 scores of component matching.

As Component Matching results in Figure4 shows an evaluation results run on model T5-large with 770M parameters performs brilliantly with token prediction.

To conduct a detailed analysis of model performance, we measure the average exact match between the prediction and ground truth on different SQL components like SELECT, WHERE, GROUP BY, ORDER BY, KEYWORDS (including all SQL keywords without column names and operators). we decompose each component in the prediction and the ground truth as bags of several , and check whether or not these two sets of components match exactly. To evaluate each SELECT component, for example, consider SELECT avg(col1), max(col2), min(col1), we first parse and decompose into a set (avg, min, col1), (max, col2), and see if the gold and predicted sets are the same.

In general, the overall performances of all models are low, indicating that our task is challenging and there is still a large room for improvement.

	easy	medium	hard	extra	all
count	248	446	174	166	1034
===== EXECUTION ACCURACY =====					
execution	0.242	0.105	0.098	0.024	0.124
===== EXACT MATCHING ACCURACY =====					
exact match	0.238	0.103	0.098	0.024	0.122
----- PARTIAL MATCHING ACCURACY -----					
select	0.984	0.947	1.000	0.857	0.965
select(no AGG)	0.984	0.965	1.000	0.857	0.972
where	0.875	0.933	1.000	1.000	0.923
where(no OP)	0.875	0.933	1.000	1.000	0.923
group(no Having)	1.000	0.938	1.000	1.000	0.974
group	1.000	0.812	1.000	1.000	0.923
order	1.000	0.929	1.000	1.000	0.974
and/or	1.000	0.915	0.897	0.880	0.926
IJUN	0.000	0.000	1.000	0.000	1.000
keywords	1.000	0.953	1.000	1.000	0.980
----- PARTIAL MATCHING RECALL -----					
select	0.250	0.121	0.098	0.036	0.134
select(no AGG)	0.250	0.123	0.098	0.036	0.135
where	0.130	0.077	0.043	0.043	0.075
where(no OP)	0.130	0.077	0.043	0.043	0.075
group(no Having)	0.300	0.113	0.308	0.063	0.140
group	0.300	0.098	0.308	0.063	0.133
order	0.409	0.173	0.218	0.038	0.160
and/or	1.000	1.000	1.000	1.000	1.000
IJUN	0.000	0.000	0.071	0.000	0.039
keywords	0.207	0.108	0.098	0.042	0.111
----- PARTIAL MATCHING F1 -----					
select	0.399	0.215	0.178	0.069	0.236
select(no AGG)	0.399	0.219	0.178	0.069	0.238
where	0.226	0.142	0.082	0.082	0.139
where(no OP)	0.226	0.142	0.082	0.082	0.139
group(no Having)	0.462	0.201	0.471	0.119	0.245
group	0.462	0.174	0.471	0.119	0.232
order	0.581	0.292	0.358	0.073	0.275
and/or	1.000	0.956	0.945	0.936	0.962
IJUN	1.000	1.000	0.133	1.000	0.076
keywords	0.343	0.195	0.178	0.081	0.199

Figure 4: Accuracy of Exact Matching, Execution Match on SQL queries with different hardness levels and Accuracy, F1, and Recall scores of Component Matching on SQL queries with different hardness levels.

MODEL NAME	BATCH SIZE	EPOCHS	#PARAMS	BLEU SCORES	EM	EA	LR
T5-SMALL	32	15	60M	25.80	4.4	4.1	5e <sup>-5</sup>
T5-BASE	32	100	220M	36.78	7.3	7.1	5e <sup>-5</sup>
T5-LARGE	32	100	770M	37.58	12.2	12.4	1e <sup>-5</sup>
T5-3B	4	3	3B	34.594	15.4	15.1	1e <sup>-4</sup>

Figure 5: Spider evaluation results on different T5 models.

## 4.5 Conclusion

In this paper we used huggingface Spider dataset, a large, complex and cross-domain semantic parsing and text-to-SQL dataset, which directly benefits

both NLP and DB communities. Based on Spider, we define a new challenging and realistic semantic parsing task. We demonstrated that generative language models trained on T5 provide a strong baseline for Text-to-SQL. Experimental results on several state-of-the-art models on this task suggest plenty space for improvement.

## References

- Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.
- Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.
- Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. *Yara parser: A fast and accurate dependency parser*. *Computing Research Repository*, arXiv:1503.06733. Version 2.