# Question2SQL: A Basic Implementation and Analysis of Methods in Natural Language to SQL Problems

**Student**
Ilmar Uduste
xudust00@fit.vut.cz

**Supervisor**
Martin Fajci
ifajcik@fit.vut.cz

## Abstract

This work aimed to solve natural language to SQL problems using a Seq2Seq, a model typically used for natural language translation. Concatenating table and column names and GloVe embeddings were also used. The translation models that were trained on WikiSQL and Spider NL2SQL datasets were also then tested and performance was expectedly lackluster on both of them, but decidedly better on the simpler WikiSQL dataset, achieving a 12% exact matching accuracy while having many other output queries barely miss the exact matching mark. These are not novel results, rather, it is very far away from top-of-the-line performance, but building the pipeline and training the models was a learning experience in and of itself. The project and tool is available for use here: (https://github.com/ilmaruduste/Question2SQL).

## 1 Introduction

So much of the data generated in today's world lies in relational database systems. Usually, to access this kind of info, someone has to interface the database and write Syntax Query Language (SQL) queries to be able to access the data in the right format, from the right table and with the right filters, but writing these queries is a technical skill that not a lot of people possess. An increasing number of people also use mobile phones for their day-to-day doings, not traditional keyboard and mouse computers, making it near-impossible to do any technical tasks. Nevertheless, technology users still want access to the information hidden away in databases. One solution to this problem is to build a natural language interface that could automatically translate the question of the user into machine-readable SQL.

## 2 Task Definition

Text-to-SQL is a task to translate a user's query spoken in natural language into SQL automatically

**Question:** Who is the player that wears number 42?
**SQL:** SELECT player WHERE no. = 42 **Result:** Art Long

| Player | No. | Nationality | Position | Team |
|---|---|---|---|---|
| Antonio Lang | 21 | United States | Guard-Forward | Duke |
| Voshon Lenard | 2 | United States | Guard | Minnesota |
| Martin Lewis | 32,44 | United States | Guard-Forward | Butler CC (KS) |
| Brad Lohaus | 33 | United States | Forward-Center | Iowa |
| Art Long | 42 | United States | Forward-Center | Cincinnati |

Figure 1: A tabular representation of a NL2SQL problem.

(Figure 1), considering the context that the user is asking the question in. It is also a semantic parsing problem, since you have to recognise what word in the input query could point to a table, what word could be a filter etc. This project focuses on implementing and testing a general NLP model, Seq2Seq, to translate natural language into SQL.

## 3 Methodology

The NL2SQL model that is implemented in this project is mostly based off of a Seq2Seq model that's used to complete NL2NL tasks. SQL is not a natural language, so most methods in NL2SQL use some kind of constraints for SQL output, but in this project, no constraints are used on the output.

In addition to the relatively barebones Seq2Seq model, concatenating column and table names to the input question and GloVe embeddings were used. Concatenation is a method that is used in many top-performing NL2SQL models, like SQLova, to add context into input queries and to connect database terms with natural language words, by giving the input query the columns and tables that it can query in the respective database (Hwang et al. (2019); Xuan et al. (2021); Ma et al. (2020)). It has been said to increase performance by a significant margin, although at the cost of training intensity.

The previously mentioned Seq2Seq model and its variations are then trained on either WikiSQL

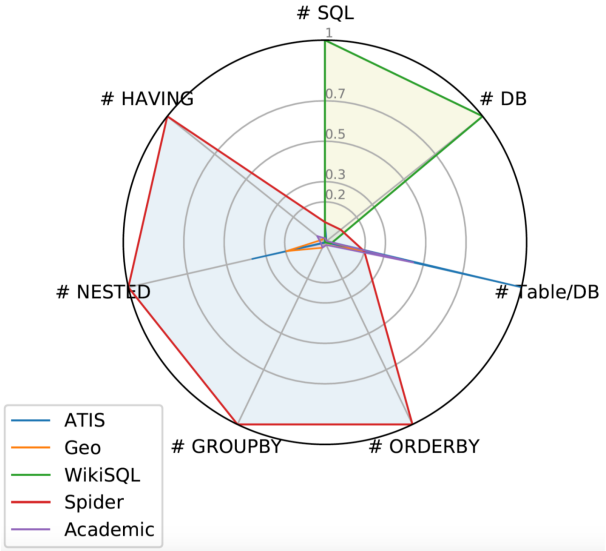


Figure 2: Spider chart of Text-to-SQL datasets (Yu et al. (2018)).

or Spider datasets and finally tested for accuracy metrics. The main evaluation metric for WikiSQL and Spider datasets is 'exact matching accuracy', which measures the number of SQL queries that the model could word-for-word predict to be the same as the original output. This metric is used in measuring the success of the models built in this project.

There is a problem however, that the correct result of a person's query could be gotten via many different SQL queries. This is usually solved by executing the model's queries against a database holding the text-to-SQL data and seeing whether executing the model's query gives the same result as executing the original query. This is called 'execution accuracy' (Yu et al. (2018); Zhong et al. (2017)), but it is not used in the current project, since the expected performance is too low to be able to actually measure execution accuracy.

## 4 Experimental Setup

### 4.1 Datasets

Choosing the correct dataset for testing NL2SQL models can be quite tricky. Many datasets exist (Figure 2), but the main choice is between the simple but non-practical WikiSQL and complex but varied Spider SQL datasets (Table 1) which are both commonly used to benchmark NL2SQL models by the academic community.

**WikiSQL:** The number of SQL queries and tables present in WikiSQL is significantly large, but all the SQL queries are simple and they only cover SELECT and WHERE clauses. Also, each database is only a simple table without any foreign key. Models trained on WikiSQL still work when tested on a different new database. However, the model cannot handle complex SQL (e.g. with GROUP BY, ORDER BY, or nested queries) and databases that have multiple tables and foreign keys (Yu et al. (2018)).

**Spider:** It spans the largest area in the spider chart of text-to-SQL datasets (Figure 2), making it the most complex and cross-domain text-to-SQL dataset. It has over 10,000 questions and about 6,000 corresponding and unique SQL queries. Most of the SQL queries cover almost all the important SQL components, including GROUP BY, ORDER BY, HAVING and nested queries. Also, all 200 databases have multiple tables linked by some foreign keys, making the Spider dataset more akin to a real-life relational database system (Yu et al. (2018)).

Both datasets were used in training and testing the models built in this project. It is worth noting that achieving a decent result on the WikiSQL dataset is significantly easier than doing so on the Spider dataset, as the methods that were had great performance and were considered top-of-the-line on WikiSQL, having up to 80% exact matching accuracy on WikiSQL test datasets, only had up to 33% exact matching accuracy on Spider datasets (Yu et al., 2018). As Spider is more akin to real-life systems, the ultimate goal of a NL2SQL model would be to get good performance on Spider, but as a proof-of-concept, this project aims to achieve results in WikiSQL that are in the same ballpark as a baseline Seq2Seq model, meaning about 20-30% exact matching accuracy (Zhong et al. (2017)).

It is worth noting that the WikiSQL dataset used in this project is an older version of the WikiSQL dataset and its format is a bit different than today's WikiSQL. The main difference is that the version in use here has NL questions and corresponding SQL queries, while the latest versions don't have the corresponding SQL queries, but rather the indices of columns that should be selected for SELECT, FROM or WHERE clauses. The changes make it difficult to train translation models, so the older WikiSQL version with approximately 56,000 SQL queries was chosen (with 15,000 test questions and queries).

| Dataset | Original Input | Original Output |
|---|---|---|
| **WikiSQL** | What song has a length of 3:05? | `SELECT Song FROM table WHERE Length = 3:05` |
| **Spider** | Give the name of each department and the number of employees in each. | `SELECT T2.department_name, COUNT(*) FROM employees AS T1 JOIN departments AS T2 ON T1.department_id = T2.department_id GROUP BY T2.department_name` |

Table 1: Examples from two of the most prominent Text-to-SQL datasets.

### 4.2 Project implementation

The implementation of the Seq2Seq model is mostly based off of a TensorFlow tutorial on neural machine translation with attention (TensorFlow, 2019), although the tutorial there is meant for running in a Jupyter notebook and the project here is an object-oriented solution in a cohesive package that can be deployed easily on any system.

The project itself is in a GitHub repository (https://github.com/ilmaruduste/Question2SQL), giving it easy access and great deployability, which played a large role in training the models, since the suitable hardware for training neural networks is difficult to come by nowadays, therefore the project was deployed onto a Google Colab environment where there were sufficient resources (e.g. a GPU with tensor cores) to train a neural network. The reference data (e.g. Spider and WikiSQL datasets, pre-trained GloVe embeddings) could be downloaded within a couple of minutes via bash scripts included with the project and the models could be created and trained using the `run_basic_seq2seq.py` script, then later tested via `test_basic_seq2seq.py`.

The whole process flow of Question2SQL can be seen in Figure 3 and it consists of many steps, the most time-consuming of which is definitely training the translation models, although preprocessing the queries and preparing the GloVe embedding matrix for use also took its time.

On a typical Google Colab system with a GPU, training the model takes about 5 minutes per epoch for Spider (8 minutes when using concatenated inputs) and approx. 10 minutes per epoch for WikiSQL. No notable gains in performance were noted for Spider beyond 3 epochs, but WikiSQL seemed to lessen batch loss and get better performance with every epoch, so that was trained for 7 epochs to get better accuracy but to keep training time low at the same time, since Google Colab will disconnect

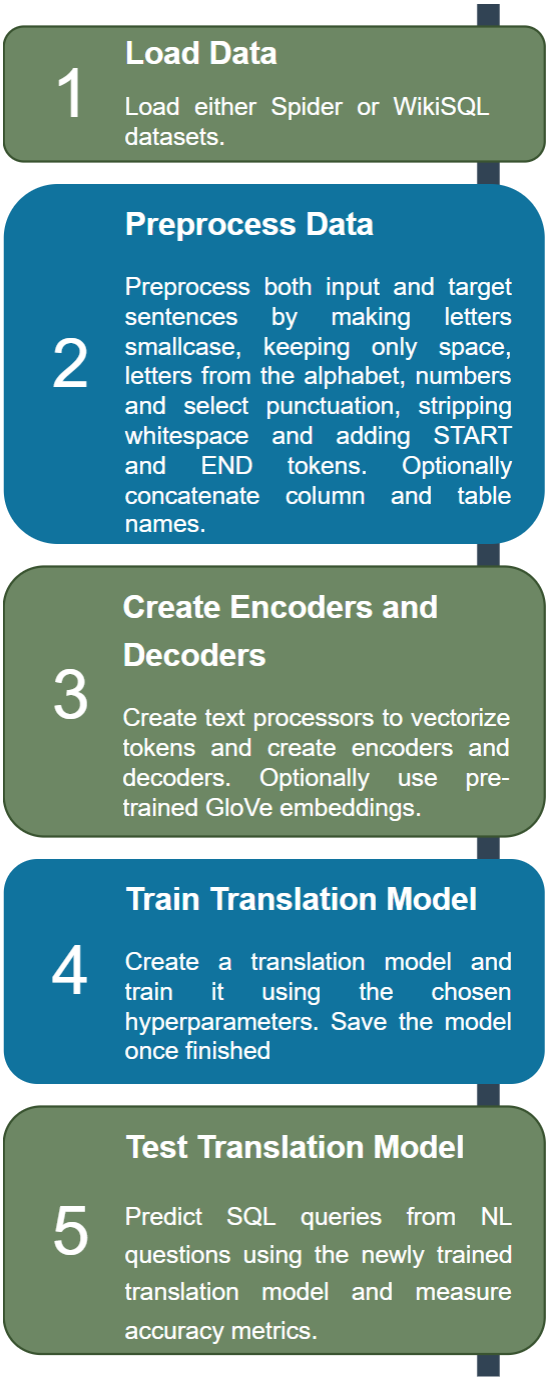


Figure 3: The process flow of Question2SQL.

| Model | Query |
|---|---|
| Original Input | Find the number of distinct type of pets. |
| Original Output | `SELECT COUNT(DISTINCT pettype) FROM pets` |
| Basic Seq2Seq Model Output | `SELECT languages FROM parties ORDER BY emailaddress DESC LIMIT` |
| Concatenated Seq2Seq Model Output | `SELECT DISTINCT invoicedetails FROM bookclub WHERE year INTERSECT SELECT artistname FROM bookclub WHERE year` |
| Basic Seq2Seq Model with Glove Embeddings Output | `SELECT count FROM faculty AS t JOIN facultyparticipatesin AS t ON t.facid t. facid WHERE t.budgettypedescription government` |

Table 2: Seq2Seq prediction examples on the Spider dataset.

if the user is away for too long. The total running time for the script was therefore about 20-30 minutes for Spider and 50-70 minutes for WikiSQL. This might be due to the smaller amount of data present in Spider compared to WikiSQL.

There were some issues with training on Google Colab, however. It seemed that using the default hyperparameters to train a Seq2Seq model on WikiSQL lead to an 'out-of-memory' issue, which was solved by training the model on `embedding units = 128` and `encoding units = 512` instead of 256 and 1024 respectively. It was also noted that Google Colab actually has runtime usage limits, meaning that if you use Google Colab too much for long-term calculations over the course of many days, it can cancel access and prompt you to purchase a plan for extra Google Colab usage (Pro or Pro+). This problem could be circumvented by just using a different gmail, though.

## 5 Results and Analysis

For Spider, 3 models were tested out: basic Seq2Seq, basic Seq2Seq with column and table concatenation, and basic Seq2Seq with GloVe embeddings. Having experimented with different approaches and translation models with different hyperparameters, **the results were on par with the very low expectations that had been set by the performance of similar models on WikiSQL and Spider. Performance on Spider was non-sensical regardless of concatenation or GloVe embeddings**, as seen in Table 2, meaning that the model outputs quite often completely ignored the input query and selected columns or features that were completely out of context. This is hypothesized to happen because of 'mode collapse', where the decoder of the Seq2Seq model learns to ignore the encoder before the encoder learns to produce anything sensible, leading to optimization being stuck in 'local mode', never achieving good performance, because it never tries to consider the encoder outputs again.

The possible solutions for this anomaly were to tune the hyperparameters (e.g. lower learning rates), which turned out not to matter in this case, or to use pre-trained sentence embeddings, which also didn't save the model on Spider. Needless to say, the exact matching accuracy of the 3 models turned out all to be 0% on Spider, which is a bit of a disappointment, but not surprising, considering that the Spider dataset has so many joins and other SQL operations which assume a knowledge of the data schema that isn't present in these naive approaches. What can be said, however, is that the model outputs queries that follow SQL syntax, meaning that it has fundamentally understood the rules of the language, e.g. there are SELECT and FROM clauses, the order of these clauses is important etc. Even if exact matching doesn't happen, outputting any kind of rule-based SQL is note-worthy in a sense.

**Results for WikiSQL, however, were more promising.** Training Seq2Seq models with GloVe embeddings and 7 epochs yielded an exact matching on 1,905 queries out of 15,878 test queries, leading to an exact matching accuracy of approx. 12%, which is below the result of a similar model in Seq2SQL (Zhong et al. (2017)), but it is at least not 0%. Many of the queries that did not exactly match were off by just one keyword and could be dependant on the table structure, since the meaning of the query is understood (e.g. SELECT nation vs. SELECT nationality). Sometimes, though, the model didn't understand the significance some number

| Original Input | Original Output | Model Output | Comment |
|---|---|---|---|
| What chassis had 39 points? | SELECT chassis FROM table WHERE points 39 | SELECT chassis FROM table WHERE points 39 | Correct |
| What was the loss that had a score of 91? | SELECT loss FROM table WHERE score 91 | SELECT loss FROM table WHERE score 107 | Wrong number |
| Which nation had 14 goals? | SELECT nationality FROM table WHERE goals 14 | SELECT nation FROM table WHERE goals 14 | Nationality vs nation |
| What was the record at week 7? | SELECT record FROM table WHERE week 7 | SELECT record FROM table WHERE week 7 | Correct |
| What song has a length of 305? | SELECT song FROM table WHERE length 305 | SELECT song order FROM table WHERE 2013 916 | Out of context prompts |
| What is the highest track number? | SELECT max track FROM table | SELECT max japanese FROM table WHERE country denmark | Non-sensical |
| What is the tournament on jul 11? | SELECT tournament FROM table WHERE date jul 11 | SELECT tournament FROM table WHERE date june 11 | Wrong month |

Table 3: Some examples of Seq2Seq prediction on the WikiSQL dataset.

in the input query and outputted a completely out-of-context number for comparison (Table 3). It could be that execution accuracy on WikiSQL in this case could be significant, but unfortunately, the reference data that came with readable and usable queries did not come with database information. Nevertheless, having any kind of positive results on WikiSQL showed that the model wasn't buggy, it was just a particular case with Spider data.

The Seq2Seq models obviously have their limitations, but they could be fine-tuned for a better performance on the WikiSQL dataset. For Spider, however, drastic changes to the input queries and model architecture should be made to achieve any kind of results.

## 6 Conclusion

This project sought out to solve NL2SQL problems using Seq2Seq, a model typically used for NL2NL problems. To give the models a better chance of predicting the correct SQL queries, concatenating table and column names to the input query and using GloVe embeddings were also used. Having built a deployable tool for creating and training these translation models ((https://github.com/ilmaruduste/Question2SQL)), the performance of these translation models were tested on two NL2SQL datasets: the simpler WikiSQL and the more complicated Spider. Performance on Spider was expectedly lackluster, as the model gave an output that mostly followed the rules of SQL, but was completely context-free and seemed to ignore the original prompt. Results on WikiSQL, however, were more promising with a 12% exact matching accuracy with many other queries being incorrect by just one or two elements. However, context-free outputs were also present, just like in Spider. In the end, these results are not anything novel, rather, it's very far away from top-of-the-line performance, but building an easily deployable tool to create neural networks for a specific task is itself a good experience and certainly a stepping stone for the future.

## References

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization.

Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and

Jianping Shen. 2020. Mention extraction and linking for sql query generation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

TensorFlow. 2019. Neural machine translation with attention. https://www.tensorflow.org/text/tutorials/nmt_with_attention. Accessed: 2021-12-05.

Kuan Xuan, Yongbo Wang, Yongliang Wang, Zujie Wen, and Yang Dong. 2021. Sead: End-to-end text-to-sql generation with schema-aware denoising.

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.

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