NL2SQL: Extension of the Seq2SQL Model

Shanelle Roman[1]

[1]Department of Computer Science, Yale University, New Haven, CT

LILY Lab

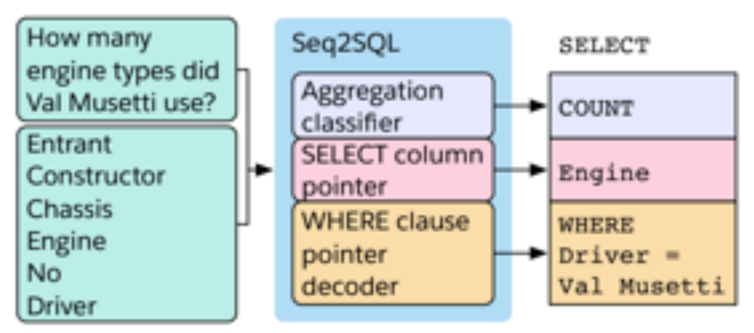
Introduction

The overall goal of this project was to translate natural language questions into executable SQL queries. This is a nascent area of research within the field of Natural Language Processing, with many exciting business applications. Previous research has mostly been done using Zhong 2017's WikiSQL dataset. I mostly worked with Tao Yu and James Ma on constructing a new dataset that more accurately reflects real-world, complex questions and on creating a seq2SQL baseline model that reflects this more difficult problem. This research project drew heavily on the approach implemented in two previous papers. The first paper, "Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning" by Zhong et.al is the one that I based most of my final project on. The second paper is "SQLNet: Generating Structured Queries from Natural Language without Reinforcement Learning" by Xu and Song. Both of these papers, however, used the WikiSQL dataset, which is extremely simplistic and not real-world. As a result, my work was primarily extending the seq2SQL model to new data.

The Dataset: Beyond WikiSQL

The WikiSQL dataset is a large crowd-sourced dataset for developing natural language interfaces for relational databases that was created by the authors of the Zhong 2017 paper. It consists of natural language questions that can be translated into simple SQL queries that contain the following basic components. SELECT AGG? COL_NAME WHERE COND1 AND COND2. As such, the SQL query structure was extremely limiting – there was only room for aggregator and column selection, and the structure of the SQL query itself was pre-determined. Moreover, there was only one table, which meant that the algorithm merely had to determine the column name. In order to define a more real-world problem, we as a team constructed a new dataset. We found approximately 200 populated databases online, and we created questions with their corresponding SQL translations. Unlike WikiSQL, we include a much wider variety of components. For example, we introduce GROUPBY, HAVING, UNION, INTERSECTION, LIMIT, SORT, ASC and DESC. In addition, this new dataset also involves nested SQL queries inside the WHERE clause.

Figure 1. Original Seq2SQL Model



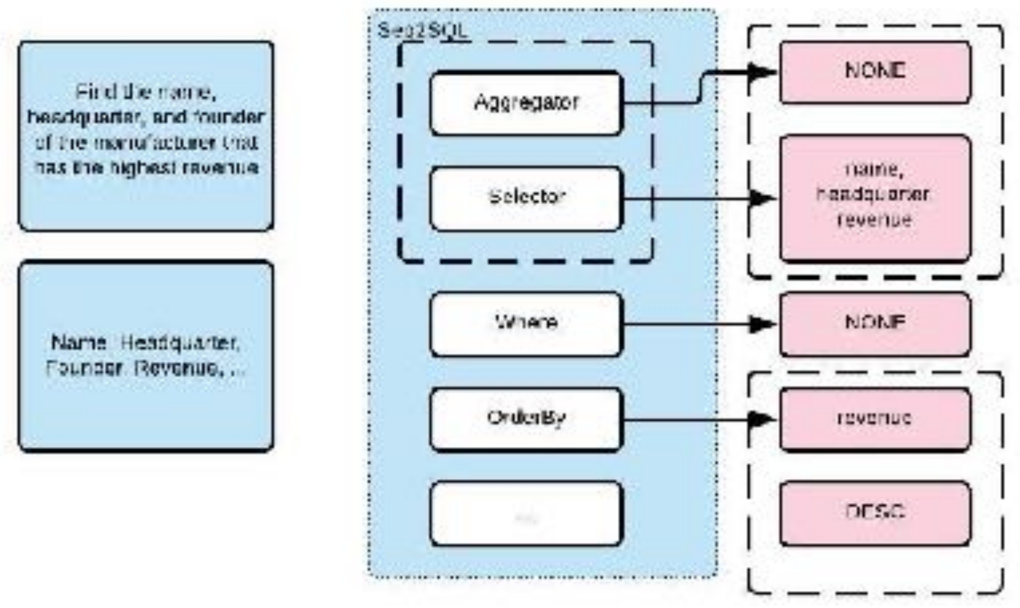

Find the name, headquarter, and founder of the manufacturer that has the highest revenue
Name, Headquarter, Founder, Revenue, ...
Seq2SQL
Aggregator
Selector
Where
OrderBy
NONE
name, headquarter revenue
NONE
revenue
DESC

Figure 2. Seq2SQL Extension for New Data

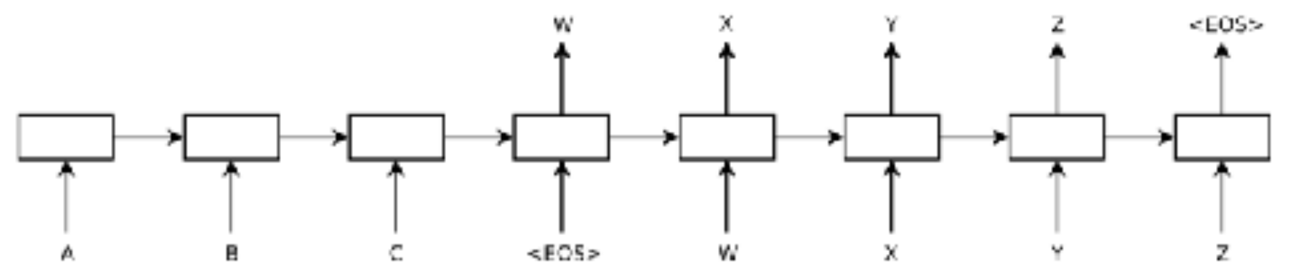
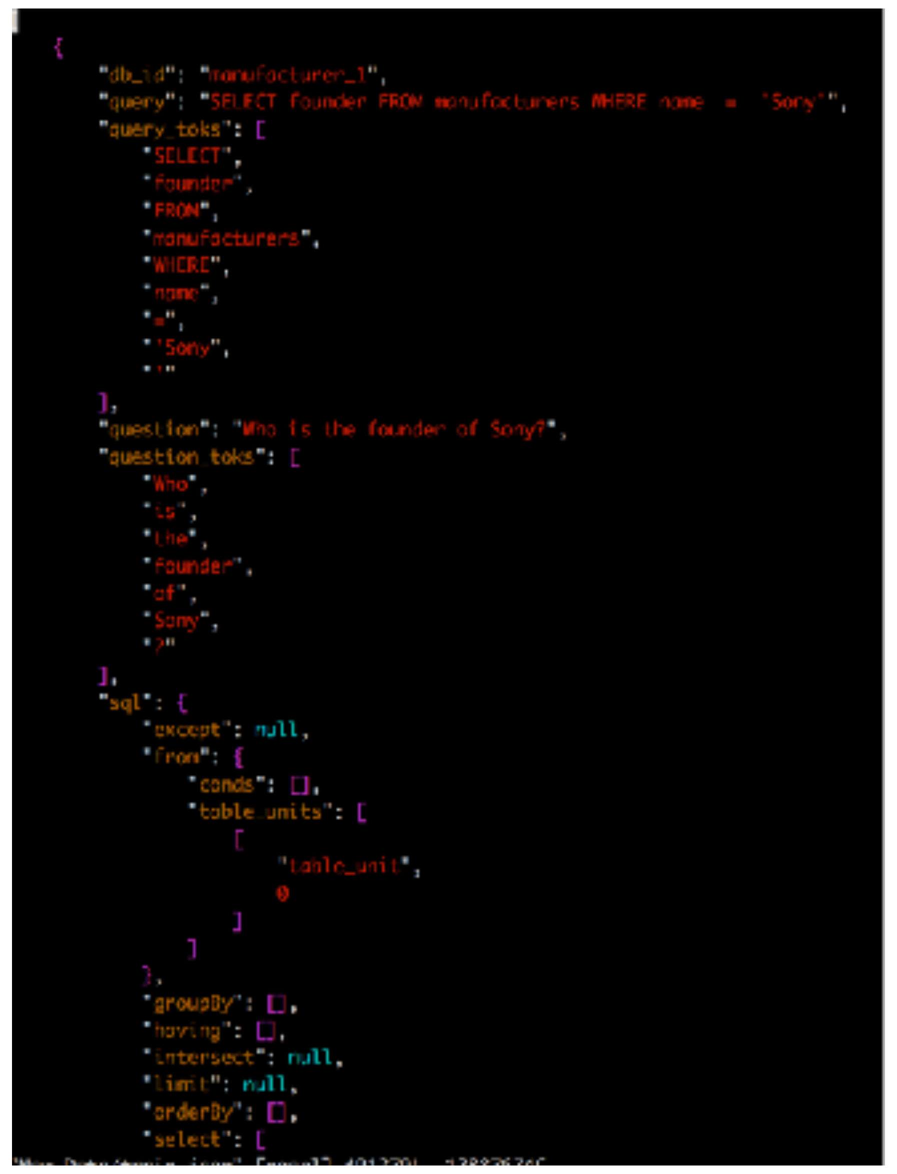
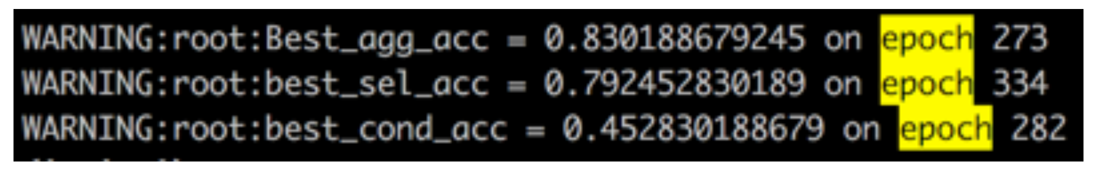


Figure 3. Encoder - Decoder Architecture



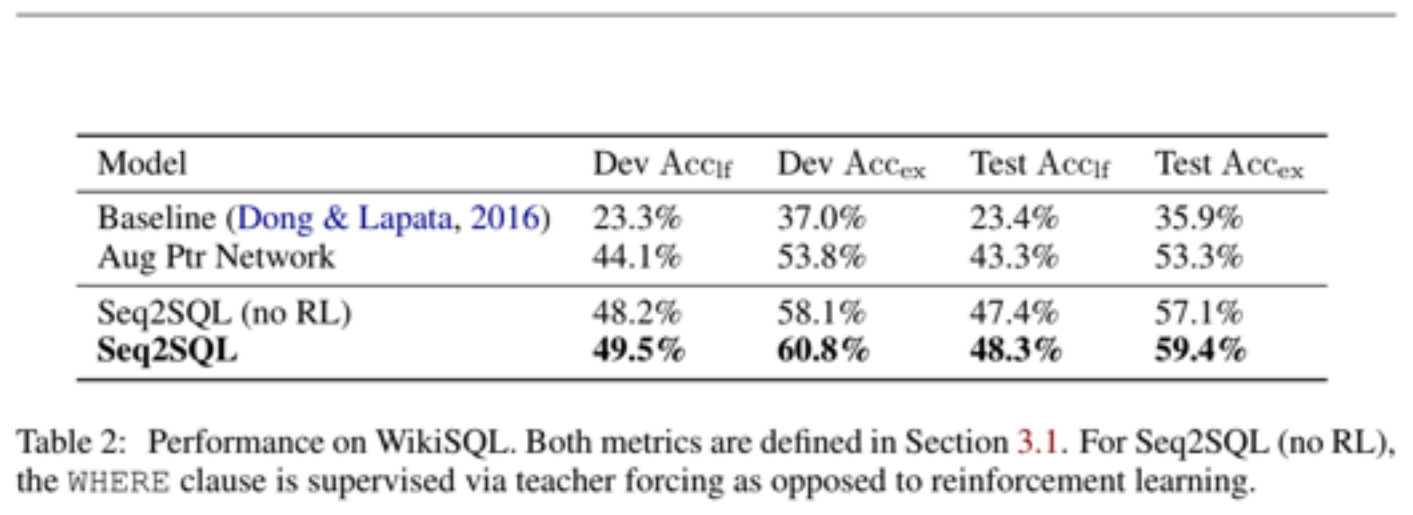
```
{
    "db_id": "manufacturer_1",
    "query": "SELECT founder FROM manufacturers WHERE name  =  'Sony'",
    "query_toks": [
        "SELECT",
        "founder",
        "FROM",
        "manufacturers",
        "WHERE",
        "name",
        "=",
        "'Sony",
        "'"
    ],
    "question": "Who is the founder of Sony?",
    "question_toks": [
        "Who",
        "is",
        "the",
        "founder",
        "of",
        "Sony",
        "?"
    ],
    "sql": {
        "except": null,
        "from": {
            "conds": [],
            "table_units": [
                [
                    "table_unit",
                    0
                ]
            ]
        },
        "groupBy": [],
        "having": [],
        "intersect": null,
        "limit": null,
        "orderBy": [],
        "select": [
```

Figure 4. Data Set Encodings



```
WARNING:root:Best_agg_acc = 0.830188679245 on epoch 273
WARNING:root:best_sel_acc = 0.792452830189 on epoch 334
WARNING:root:best_cond_acc = 0.452830188679 on epoch 282
```

Figure 5. Screenshot of Some Results



Model	Dev Acc$_{lf}$	Dev Acc$_{ex}$	Test Acc$_{lf}$	Test Acc$_{ex}$
Baseline (Dong & Lapata, 2016)	23.3%	37.0%	23.4%	35.9%
Aug Ptr Network	44.1%	53.8%	43.3%	53.3%
Seq2SQL (no RL)	48.2%	58.1%	47.4%	57.1%
Seq2SQL	**49.5%**	**60.8%**	**48.3%**	**59.4%**

Table 2: Performance on WikiSQL. Both metrics are defined in Section 3.1. For Seq2SQL (no RL), the WHERE clause is supervised via teacher forcing as opposed to reinforcement learning.

Figure 6. Zhong 2017 Accuracies

The Model: Extending Seq2SQL

I modified the Zhong 2017 a little bit for the completion of the different task. In the Zhong 2017 paper, the authors used an augmented pointer network to generate the SQL query, which generates a scalar attention score that represents the chance that SQL token s is the correct translation at each position t of the query. However, the Zhong 2017 paper split the generation of the SQL query into 3 separate neural network architectures. For them, the SELECT and AGG predictions were substantially simpler than the WHERE sequences. However, for our new dataset, we could have many columns selected from different tables and the aggregator operations could be applied to any permutation of those columns. As a result, I used the attentional encoder-decoder architecture for the SELECT and AGG operators, and I predicted those two operators together. This eliminated some problems with the previous approach. First, it makes it easier to match up the AGG operator predicted with its corresponding SELECT prediction. Second, it is extensible for any number of columns and aggregator operators. However, there are still some issues. Because the embeddings are based on the column names, the model has difficulty picking the right column when the two columns from different tables have the same name.

Conclusion

This is a unique problem formulation that provides a lot of opportunities for future research. In particular, even beyond the baseline Seq2SQL model, the current accuracies will leave a lot of room for improvement given the difficulty of the problem. Moreover, for my part, I hope to finish up implementing the architecture for GROUPBY, HAVING, etc. However, there is still much to be done in terms of predicting the SQL structure of the query translation (e.g. whether there is a GROUPBY or not).

Acknowledgement

I would like to thank Professor Radev, Tao Yu, and James Ma for their wisdom and help on this project.