NLP Final Project Proposal: NL2SQL

Tinglong Liao (tl2564), Xue Bai (xb347), Yuhan Yao (yy2564)

1. Introduction & Objective

Among programmers, computers are notoriously infamous for being fastidious about command integrity. That is why natural language is insufficient as computer commands to extract data from databases and why Computer Science and Data Science students at NYU Shanghai are encouraged to take the course Databases so that we can learn the exact SQL syntax to perform such tasks. In hope of empowering more non-programmers to interact with SQL databases and more efficient use of stored data, this project is designed to convert natural language into SQL commands (NL2SQL) accurately and efficiently.

With the rise of Conversational User Interface (CUI) such as voice assistants and chatbots, Semantic Parsing became a hot research topic in the field of Natural Language Processing, in which extrapolating formal meaning representation from natural language has attracted increasing attention. Among various natural languages, we will choose Chinese as our context language; among tasks such as NL2BASH, NL2Python, NL2Java and so on, we will choose SQL as the target. Questions described in natural language and corresponding SQL statements will be used to train and test NL2SQL models. We will compare models such as BERT-based variations and LSTM with a pointer network in search of the best possible model for this task via different evaluation matrices.

2. Dataset & Pre-processing

We will be using the dataset from Github provided by Zhuiyi Technology [1][2]. The dataset is divided into three sets, with 40, 000 labeled data as the training set, 5,000 labeled data as the validation set and 10, 000 unlabeled data as the test set. All three sets include two files. The first file contains the id and name of the tables and the name, data type, and values of the columns in each table. The second file contains the requests in plain Chinese, the id of the required table, and the SQL statements that correspond to the requests. Each SQL statement has four labels: the id of the selected columns, the aggregate functions associated with the selected columns (avg, sum, etc), the relationship of conditions in where clause (and, or), and a list of conditions. The list of conditions contains several sublists. Each sublist represents a condition, which is described by three labels: the id of the conditional column, the operation associated with the conditional column (<, >, ==, etc), and the value associated with the conditional column. An example of the data can be seen in Fig.1.


```
{"table_id": "69cc7e75334311e9b7f8542696d6e445", "question": "厦门的这一周的成交面积是多少",
"sql": {"agg": [0], "cond_conn_op": 0, "sel": [1], "conds": [[0, 2, "厦门"]]}}
```


Figure 1. An example input entry of the model

Since there are numbers, dates, and words that are expressed in different ways but have the same meaning in the requests, it may be necessary to use name-entity recognition or word similarity to change different expressions into the same form. To better incorporate the given information, we will also need to combine the two files together using the id of the table. The column names will be added to the request, separated by [sep] in order to form better input data.

3. Methodology & Evaluation

Since this dataset can be viewed as the Chinese version of the more popular WikiSQL dataset, we plan to recreate the models in the original paper as our baseline [3]. Namely, we plan to start by implementing a pointer network and encoded by an LSTM model. In that paper, the question sequence is fed into the model and at each hidden state produces a word in the SQL vocabulary(select, from, where, and column names). In addition, we plan to try several BERT-based models, which treat column names both as a label and as part of the input. Instead of only feeding in the question as input, we want to concatenate the input and column names to be the input. This is because most of the time column names have certain meanings that are relevant to the question. For example, when the question is "What's the postal code for Shanghai", the column we are selecting is likely to be named as code, post_code, p_code, etc. In addition, the original paper generates all the SQL slots in one model. We want to experiment if training multiple models that are designed to fill in different slots would be better. The idea comes from the intuition that select, condition, and aggregation clauses are independent in a sentence, so filling them separately might help improve accuracy and speed up the training process. If time permits, we would like to do more literature review on the state-of-the-art models of WikiSQL to inspire more sophisticated models.

For evaluation, the paper presents two evaluation matrices: logic form accuracy and execution accuracy [3]. The logic form accuracy stands for the accuracy that the generated SQL sequence is exactly the same as the label. The execution accuracy, on the other hand, focuses on the accuracy of producing the same result/logic. While logic form accuracy cares about the ordering of conditions/column selection, the execution accuracy only evaluates whether the logic is right. In this project, we aim at correctly filling the select, condition, and aggregation slots, so we are only going to use execution accuracy to evaluate the model. In addition, the accuracy for selecting the right column(s), the accuracy of getting the right condition(s), and the accuracy of having the right aggregation is also going to be evaluated.

References

[1] ZhuiyiTechnology. "ZhuiyiTechnology/TableQA." Github, 2020, github.com/ZhuiyiTechnology/TableQA?spm=5176.12282036.0.0.499949b3lKty5H.
[2] Sun, Ningyuan, et al. "TableQA: a Large-Scale Chinese Text-to-SQL Dataset for Table-Aware SQL Generation." *ArXiv.org*, 10 Jun. 2020, arxiv.org/abs/2006.06434.
[3] Zhong, Victor, et al. "Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning." *ArXiv.org*, 9 Nov. 2017, arxiv.org/abs/1709.00103.