Enhancing real-time data querying in chemical engineering: A bilevel vocabulary-constrained seq2seq approach

Zeheng Zhao, Jinsong Zhao

Department of Chemical Engineering, Tsinghua University, Beijing 100084, China

[zh-zhao21@mails.tsinghua.edu.cn](mailto:zh-zhao21@mails.tsinghua.edu.cn), jinsongzhao@tsinghua.edu.cn

Abstract

Real-time interaction between front-line operators and the data stored in SQL databases is an essential part of the intelligent chemical process industry, and NL2SQL (natural language to SQL) is currently lacking in the process industry. First, we constructed an NL2SQL dataset with characteristics of the process industry, which poses challenges to existing NL2SQL algorithms. Next, we proposed a seq2seq model with bilevel vocabulary constraints to adapt to the characteristics of the process industry. The first level is a global restriction to identify the entities and functions used in the SQL output. The second level is a local restriction to limit the range of the next output word determined by BNF (Backus-Naur form) grammar, simplifying the training of the model in complex sentences. On the proposed ChESQL dataset, the Jaccard similarity increased from 69.94 % to 74.10 %, and the exact matching rate increased from 4.36 % to 7.95 %.

**Keywords**: NL2SQL, natural language processing, seq2seq, machine translation.

* 1. Introduction

A real-time question-answering system helps in decision-making and improves accuracy in chemical plants. A crucial aspect of the question-answering system is translating NL questions into SQL queries. Our contribution to the NL2SQL system in chemical engineering includes two parts. First, we introduce ChESQL, a new dataset with few precedents in the industrial field, based on an alarm management database from a petroleum refinery. The dataset comprises 500 prototype questions, which are augmented to 2130 NL-SQL pair instances (Figure 1). The dataset exhibits three characteristics: extensive use of MySQL functions for real-time calculations; numerous entities in the dataset, which matches the case in chemical engineering; being derived from real-world problems, which makes it more complex. Second, we propose a seq2seq (sequence-to-sequence) model with bilevel vocabulary constraints (Figure 2) to address these challenges. The first level is global and involves creating a vocabulary subset for the output SQL query. The second level is local and relies on a BNF based on MySQL syntax. By using bilevel constraints, we enhance precision both lexically and grammatically.

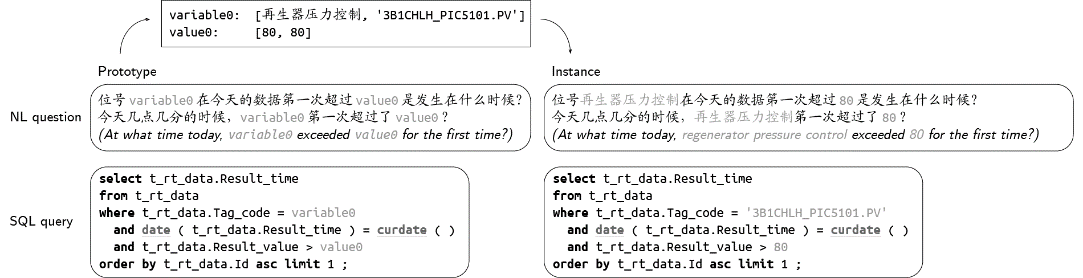


Figure 1: An example in ChESQL, in both prototype and instance forms.

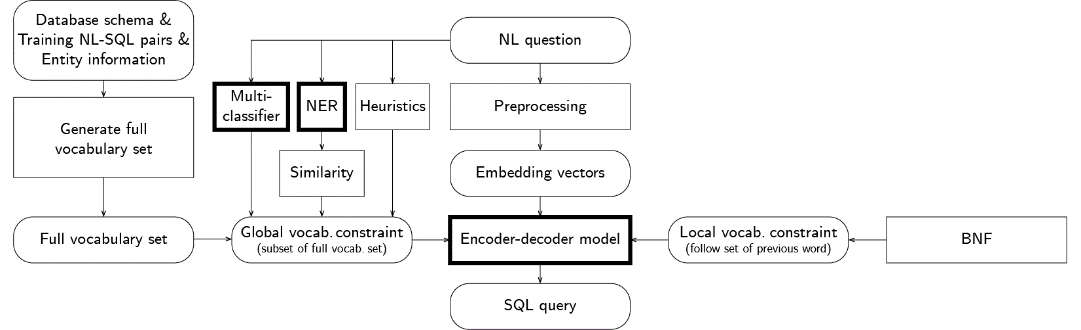


Figure 2: Architecture of the proposed model. Bold blocks represent neural models.

* 1. Dataset
     1. Dataset overview

ChESQL is a dataset created specifically for the process industry. The lengths of the 500 distinct SQL queries and 2130 NL questions are shown in Figure 3(a)(b). Figure 1 shows an example NL-SQL pair from ChESQL, and Figure 4 shows another pair from GeoQuery (Iyer et al., 2017), which is a commonly used dataset. Through comparison, it can be seen that in ChESQL, the three characteristics of the process industry mentioned before are reflected. Specifically, first, ChESQL involves more functions (shown in underlined bold). Although the number of functions in ChESQL is limited, they are all related to data processing (such as average and max) or time-related operations in MySQL (such as timediff and curdate), which are naturally required in the process industry. Second, the dataset contains many named entities and numbers (shown in light gray). Moreover, they may have different representations in SQL and NL, and the relation between the two can only be known through definitions but not through the training of neural models. Third, the SQL queries in ChESQL can be complex. On the one hand, the dataset is taken from a real industrial process and verified by engineers. On the other hand, it is a feature of SQL. Simple questions may also have complex expressions, so the length of the SQL queries and NL questions are not always related (Figure 3(c)). In addition to the length, complexity is also reflected in the number of SQL keywords contained in each SQL query (Figure 3(f)). Through comparison with other existing single-domain datasets in Table 1, we see that ChESQL has the highest average count of functions and distinct keywords and the second highest number of distinct SQL query prototypes and NL questions. Please note that although we present ChESQL with NL questions in Chinese, compared to other datasets in English, the difference is only reflected in average NL length, and other comparisons are totally fair.

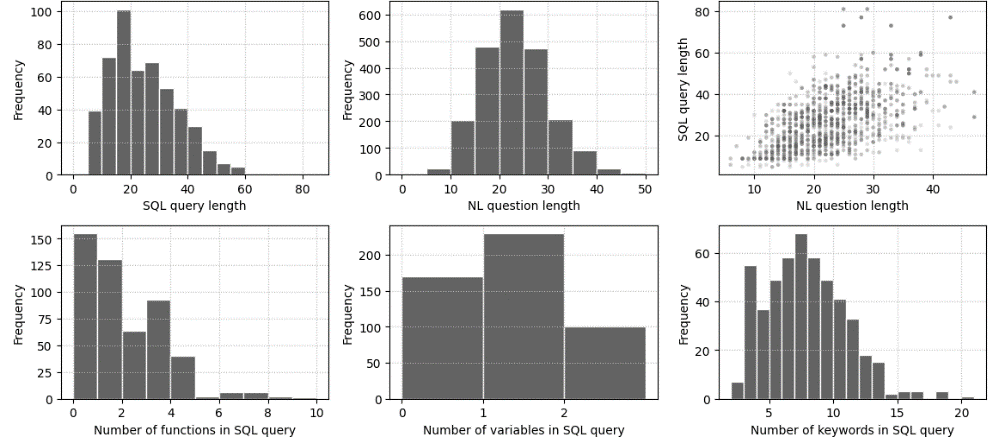


Figure 3: Basic statistics of ChESQL. (a) Histogram of the lengths of 500 distinct SQL queries. (b) Histogram of the lengths of 2130 NL questions. (c) Scatter plot of the lengths of NL-SQL pairs. (d) Histogram of the number of functions in SQL queries. (e) Histogram of the number of variables in SQL queries. (f) Histogram of the number of keywords in SQL queries.

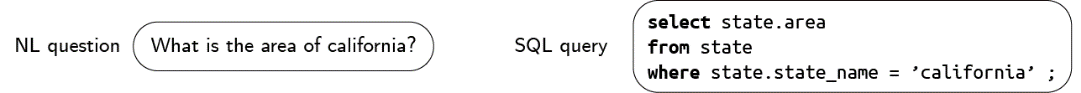
Figure 4: Sample NL-SQL pair from GeoQuery.

Table 1: Comparison of ChESQL and other single-field NL2SQL datasets.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset | #SQL | #NL | Ave SQL len | Ave NL len | Ave # of functions | Ave # of variables | Ave # of keywords | Ave # of distinct keywords |
| ChESQL | **500** | **2130** | 24.5 | 22.8 | **1.62** | 0.88 | 7.36 | **6.52** |
| Academic | 185 | 196 | 37.8 | 13.2 | 0.54 | 1.59 | 7.46 | 5.21 |
| ATIS | **947** | **5280** | 100.8 | 10.5 | 0.22 | 4.36 | 19.29 | 5.50 |
| GeoQuery | 246 | 877 | 27.8 | 7.5 | 0.92 | 0.44 | 7.51 | 4.10 |
| IMDB | 89 | 131 | 30.5 | 10.2 | 0.30 | 1.34 | 6.16 | 4.70 |
| Restaurants | 23 | 378 | 30.6 | 10.1 | 0.35 | 1.78 | 6.13 | 4.09 |
| Scholar | 193 | 817 | 39.0 | 6.6 | 0.70 | 1.56 | 8.82 | 6.41 |
| Yelp | 110 | 128 | 29.3 | 9.9 | 0.45 | 1.98 | 6.22 | 4.63 |

* + 1. Construction of the dataset

The dataset contains 4 tables with a total of 24 columns. The definition of each table is as follows: Table t\_rt\_info is the tag information table, containing meta-information such as tag codes, and upper and lower alarm thresholds. Table t\_rt\_data is the real-time data table, containing the timestamps and measured values of tags described in Table t\_rt\_info. Table t\_model\_data is the model definition table, which defines the inference models used when an alarm occurs. Table t\_inference\_data contains the inference results. These four tables are typical as they include the whole process of defining data, obtaining data, defining custom functions, and applying custom functions.

To generate a prototype NL-SQL pair, we first write an NL question and its corresponding SQL query, and then label the words that can be replaced, such as entities and numbers. Then the prototype pair is used to generate more instance pairs by replacing the labeled words. Finally, we check the validity and consistency of SQL queries and the diversity and unambiguity of NL questions.

When splitting the dataset into training and test sets, we use the method described in (Finegan-Dollak et al., 2018), providing question-based and query-based division.

* + 1. Evaluation metrics

The evaluation criteria of the dataset include the following three. Jaccard similarity evaluates how close the output SQL query is to the true query in vocabulary, therefore it can test if the NL2SQL system can find out the correct functions and entities. The SQL validity rate evaluates how many output queries are grammatically correct. The exact matching rate assesses how many output queries are identical to the true queries.

* 1. Model
     1. Generating the full vocabulary set

Compared to machine translation tasks, the output vocabulary in NL2SQL tasks depends heavily on the dataset, which needs to be decided case by case. As shown in Figure 2, after obtaining the dataset, we added the table and column names according to the table schema, the correspondence between the entities in NL and SQL according to the entity information, and the functions according to the SQL queries in the training set. Then SQL keywords such as SELECT are added. Finally, some values, such as 0, 1, True, and False, and punctuation are added. Together, these make up the full vocabulary set ***w*** required for the output in the seq2seq model.

* + 1. Global vocabulary constraint

In ChESQL, there are a lot of functions and entities, resulting in a relatively large full vocabulary, and the entity correspondence is difficult to train. Therefore, we introduce a first level of vocabulary constraint, sifting out the words for each input NL question. As shown in Figures 2 and 5, multi-classifier and NER are used to accomplish this task.

The multi-classifiers include two separate classifiers, identifying which tables and which functions will be used, respectively. These two types of words have small vocabulary, high frequency in the dataset, and weak correspondence with NL expressions, so the multi-classification strategy is suitable. Once the classification is finished, only the corresponding functions and tables (including the columns in the table) are added to the global vocabulary constraint .

NER is used to identify the entities. Entities have a large vocabulary, low frequency in the dataset, and strong correspondence with NL expression. After NER recognizes the entities, the SQL form of the entities will be added to .

Finally, we use a heuristic approach to identify the numbers. Numbers in NL questions are added directly to .

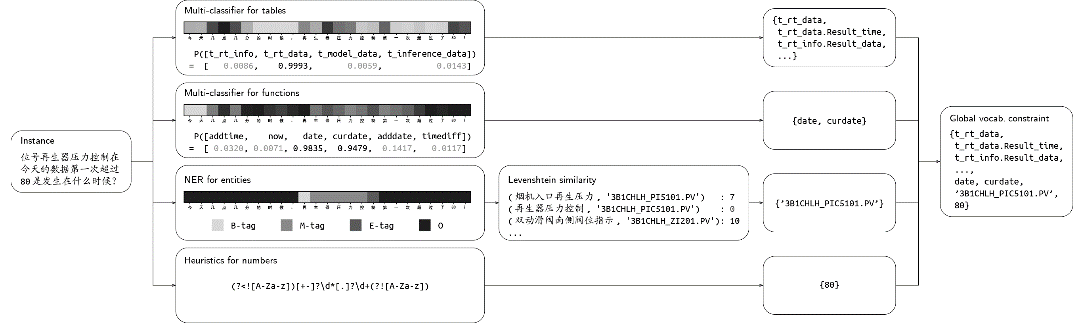


Figure 5: The flow chart for generating global vocabulary constraints from an NL question.

* + 1. Local vocabulary constraint

When the model translates NL questions into SQL queries, further restrictions are needed to ensure the validity of the output. Therefore we introduce the second level of vocabulary constraint, which is a local one. BNF can specify the syntax of languages using simple notations, which allows a sentence to be parsed. The follow set contains all valid words that can directly follow one word, which is decided after BNF is defined. For example, in MySQL, “SELECT” can be followed by any column but never another “SELECT” or “FROM”. This way, the validity of the output SQL query is guaranteed.

* + 1. Seq2seq model with bilevel vocabulary constraint

The global and local vocabulary constraints are put into use in the seq2seq model. Denote the 0-1 indicator matrix as ***I***global and ***I***local, each with size *l*×*s*, where *l* is the target length of the output SQL query and *s* is the size of the full vocabulary set ***w*** discussed in Section 3.1. The (*i*, *j*)-th element in ***I***global and ***I***local represents if the *j*-th word in ***w*** is allowed for the *i*-th place in the output SQL query under the global vocabulary constraint and the local vocabulary constraint .

When training, we design a loss function that punishes when a word that does not satisfy the bilevel vocabulary constraints has a high output weight. More specifically, the loss function is

where ***O*** is the output weight matrix with size *l*×*s*, Sumsqr(***A***) is the operation of summing all squared values in matrix ***A***, and *α* and *β* are two adjustable parameters.

When predicting, for each output token, we mask the words that do not satisfy the bilevel vocabulary constraints, i.e., multiplying the decoder output weight vector ***d*** by the 0-1 indicator vectors ***i***global and ***i***local (both with size 1×*s* as we predict only one word at a time) before performing the argmax operation.

* 1. Experiment
     1. Setup

We use ChESQL and query-based split, which results in a training set including 1740 NL-SQL pairs with 408 distinct SQL queries and a test set including 390 NL-SQL pairs with 92 distinct SQL queries. For word embedding, we use chinese-bert-wwm-ext (Cui et al., 2021). For the multi-classifier, we use NeuralNLP (Tencent, 2019), which is a TextRNN framework based on attentioned GRU. For NER, we use NCRF++ (Yang and Zhang, 2018), which is a framework based on BiLSTM-CRF. For the encoder-decoder model, we use a transformer model with the attention mechanism. We train the seq2seq model for 50 epochs, using fine-tuned *α*=*β*=10-3.

* + 1. Results
       1. Multi-classifier and NER

In the multi-classifier, each NL question is labeled by one or multiple classes. And we assess the model label-wise. The result after 20 training epochs is shown in Table 2.

Table 2: Model evaluation for the two multi-classifiers trained separately for tables and functions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tables/Functions | Data split | #True labels | #Predict labels | #Right labels | F1-score |
| Tables | Training | 2474 | 2474 | 2473 | 0.9996 |
| Test | 552 | 558 | 535 | 0.9640 |
| Functions | Training | 3928 | 3876 | 3841 | 0.9844 |
| Test | 832 | 828 | 824 | 0.9928 |

In NER, each token in the NL questions is labeled by the BIO tagging. Therefore, the performance of the model can be assessed by its accuracy and F1-score. The result after 20 training epochs is shown in Table 3.

Table 3: Model evaluation for the NER model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data split | #All tokens | #Right tokens | Accuracy | F1-score |
| Train | 46926 | 46889 | 0.9992 | 0.9917 |
| Test | 10788 | 10693 | 0.9912 | 0.9664 |

* + - 1. NL2SQL translation

We have the plain seq2seq model as the baseline. The three aforementioned evaluation metrics are listed in Table 4. Both the question-based and the query-based split are used.

Table 4: Model evaluation for the NL2SQL task.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Proposed model with constraints | | Baseline seq2seq | |
| Dataset split | Query-based | Question-based | Query-based | Question-based |
| Jaccard similarity | 0.9772/0.7410 | 0.9871/0.8181 | 0.9555/0.6994 | 0.9561/0.7985 |
| SQL validity rate | 1.0000/1.0000 | 1.0000/1.0000 | 0.9028/0.5225 | 0.9707/0.7271 |
| Exact matching rate | 0.7826/0.0795 | 0.8948/0.2535 | 0.6092/0.0436 | 0.6939/0.1718 |

* + 1. Analysis
       1. Multi-classifier and NER

From the results in Table 2 and Table 3, we can see that the multi-classifier and NER work with high accuracy. Therefore, these additional modules play an effective role in aiding the seq2seq model in deciding which words are valid.

* + - 1. NL2SQL translation

Compared to the baseline model, our model performed better in all evaluation metrics. Jaccard similarity in the test set increased from 69.94 % to 74.10 % in the query-based split, which means the output of the model contains more correct vocabularies after we add on the global vocabulary constraint. The SQL validity rate achieves 100% in our model in all trials as it is inherited from the local vocabulary constraint. In terms of the exact matching rate, the bilevel vocabulary constraint does improve the performance, e.g., from 4.36 % to 7.95 % in the test set of the query-based split. However, the exact matching rate still has a large potential, as in all setups, this metric exceeds 60 % in the training set and remains below 20 % in the test set. This can be further improved by considering result matching rate, using a larger dataset or even large language models.

* + - 1. Question-based and query-based tasks

Question-based and query-based splits offer challenges at different levels. Question-based split has the same structure of SQL queries in both the training set and test set, so it becomes a classification task rather than a translation task. A query-based split ensures that there are no queries derived from the same prototype in the training set and test set. In all models, the evaluation metrics of the question-based split are better than those of the query-based split. This also shows that in future NL2SQL tasks, we need to place more emphasis on query-based dataset splitting, as it is closer to practical applications.

* 1. Conclusion

We constructed ChESQL, which is a Chinese NL2SQL dataset in the field of chemical engineering, featuring a large number of entities and functions. We then proposed a seq2seq model with bilevel vocabulary constraints to enhance the NL2SQL translation performance. A global vocabulary constraint narrows down the vocabulary range to aid finding the correct entities and functions. A local vocabulary constraint enables the system to be aware of the grammatical state and thereby generate valid queries. This model outperforms the baseline seq2seq model, improving Jaccard similarity, SQL validity rate, and exact match rate.

References

Cui, Yiming, Che, Wanxiang, Liu, Ting, Qin, Bing, Yang, Ziqing, 2021, Pre-Training With Whole Word Masking for Chinese BERT, IEEE, 10.1109/TASLP.2021.3124365.

Finegan-Dollak, Catherine, Kummerfeld, Jonathan K., Zhang, Li, Ramanathan, Karthik, Sadasivam, Sesh, Zhang, Rui, Radev, Dragomir, 2018, Improving Text-to-SQL Evaluation Methodology, ACL, 10.18653/v1/p18-1033.

Iyer, Srinivasan, Konstas, Ioannis, Cheung, Alvin, Krishnamurthy, Jayant, Zettlemoyer, Luke, 2017, Learning a Neural Semantic Parser from User Feedback. ACL, 10.18653/v1/p17-1089.

TENCENT, 2019, NeuralNLP-NeuralClassifier, https://github.com/Tencent/NeuralNLP-NeuralClassifier.

Yang, Jie, Zhang, Yue, 2018, NCRF++: An Open-source Neural Sequence Labeling Toolkit, Association for Computational Linguistics, 10.18653/v1/p18-4013.

Zhong, Victor, Xiong, Caiming, Socher, Richard, 2017, Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning, arXiv, 10.48550/ARXIV.1709.00103.