GP: Context-free Grammar Pre-training for Text-to-SQL Parsers

Liang Zhao, Hexin Cao, Yunsong Zhao
OneConnect Financial Technology Big Data Lab
Shanghai, China
{zhaoliang146,caohexin771,zhaoyunsong244}@ocft.com

ABSTRACT
A new method for Text-to-SQL parsing, Grammar Pre-training (GP), is proposed to decode deep relations between question and database. Firstly, to better utilize the information of databases, a random value is added behind a question word which is recognized as a column, and the new sentence serves as the model input. Secondly, initialization of vectors for decoder part is optimized, with reference to the former encoding so that question information can be concerned. Finally, a new approach called flooding level is adopted to get the non-zero training loss which can generalize better results. By encoding the sentence with GRAPPA and RAT-SQL model, we achieve better performance on spider, a cross-DB Text-to-SQL dataset (72.8 dev, 69.8 test). Experiments show that our method is easier to converge during training and has excellent robustness.

1 INTRODUCTION
In recent years, with the development of artificial intelligence technology, how to directly generate SQL statements that interact with database systems through the analysis of natural language has become one of the research hotspots. Current research work usually adopts a Natural Language Interface to Database (NLIDB) to realize the interaction between user’s questions and the database system to obtain and analyze data(Baik et al., 2019).

The core problem of NLIDB is to convert the input text information into SQL statements (Text-to-SQL). In order to solve this problem, there are two main approaches at present: (1) The method based on rule template, that means, the natural language is classified according to the common SQL grammar, and the corresponding SQL templates belong to different categories(Popescu et al., 2004, Unger et al., 2012, Li and Jagadish, 2014). This type of method requires manual summarization of experience and has a high time cost. In addition, with the switch of application scenarios, the existing templates are often difficult to meet the requirements, and the migration is poor; (2) Based on the deep learning method, the neural network is used for end-to-end implementation(Zhong et al., 2017, Yu et al., 2018a,b, Bogin et al., 2019, Guo et al., 2019). This method can be self-optimized by continuously adding sample information. It has the advantages of high accuracy and strong stability, and is receiving more and more attention from the academic community. By incorporating the BERT encoder, the accuracy on the WikiSQL dataset can reach above 90%. However, these deep-learning methods does not achieve satisfactory performance on a cross-domain Text-to-SQL scenario such as Spider. As is show in Figure 1, this...
SQL query include nested clauses like GROUP BY and connections in multiple tables. Such grammar details are rarely concerned by users so they are hardly mentioned in questions. Bailin Wang et al. proposed a relation-aware framework named RAT-SQL and achieve state-of-art accuracy on Spider dataset. On the other hand, pre-training language models based on structured table data and natural language of users are developed. At early stages, BERT(Devlin et al., 2018) and RoBERTa(Liu et al., 2019) for contextual sentences are applied in cross-domain Text-to-SQL scenario, but the relation between the tables and fields of the database is not considered. A grammar-augmented pre-training model (GRAPPA) describing the joint representations of textual and tabular data is presented (Yu et al., 2020). By combining the pre-training model with other downstream methods like RAT-SQL, the accuracy on cross-domain tasks can be greatly improved.

In this paper, a context-free grammar pre-training (GP) approach is proposed. Instead of pre-training primary input vectors, this method is intended for downstream models. In the preprocessing module, the input natural language questions are split into several single words. Using n-gram algorithm, columns can be detected by matching schema information. One of its value will be added so a new question sentence is generalized as the model input. For the design of loss function, we adopt flooding level, a new method to avoid local minimum values.

On the basis of GRAPPA/RAT-SQL framework, experiments show that our approach reaches a much higher accuracy on Spider test set. Results also prove that this method has excellent robustness.

2 RELATED WORK

Pre-training models for NLP parsing Text-to-SQL task contains both unstructured user question and structured schema information. Early research use usual pre-training models like Elmo(Peters et al., 2018), BERT(Devlin et al., 2018) and RoBERTa(Liu et al., 2019), to represent textual information for unstructured language questions. There has been great improvement in joint textual-tabular field like question answering(Chen et al., 2020) and table semantic parsing(Yu et al., 2018c) by learning better representations from the input text and table information, but most of them consider single tables. Recent pre-training work focus on achieving high-quality cross-modal representation. TaBERT (Yin et al., 2020) is pre-trained by using millions of web tables. It can represent complete structure for different tables and make some matrix computations in table semantic parsing. However, the noisy context information weakens its performance on Text-to-SQL task. In this paper, we adopt GRAPPA, the grammar-augmented pre-training method using a novel text-schema link objective and masked language modeling (MLM). By combining GRAPPA as feature representation layers with other downstream models, there have been great accuracy on Spider dataset.

Neural networks for Text-to-SQL Previous networks are intended to solve problems in single table dataset like WikiSQL. The Seq2SQL model based on the strategy mode(Zhong et al., 2017) is applied in Text-to-SQL tasks and achieves
59.45% SQL execution accuracy on WikiSQL dataset. Then TypeSQL (Yu et al., 2018a) is proposed, which further extracts the keywords in the question sentence by combining external knowledge and database field enumeration values. The above method has achieved obvious results in single-table query, but it is not enough to solve the complex mode of multi-table query. EditSQL (Zhang et al., 2019) uses an editing mechanism to introduce historical information for user queries, and its matching accuracy on Spider dataset reaches up to 32.9. IRNet (Guo et al., 2019) adopts an intermediate representation named SemQL to translate complex SQL queries into a syntax tree. Using pointer network (Vinyals et al., 2015) for downstream tasks, it achieves an accuracy of 54.7 on Spider test set. Graph neural networks are also concerned to represent the relations for schema information. Global gated graph neural network (Bogin et al., 2019) is designed to train the structure of database patterns and apply it in the encoding and decoding stages. Recently RAT-SQL (Wang et al., 2019) uses a relation-aware self-attention mechanism for schema encoding, feature representation and schema linking. It obtains the state-of-art accuracy of 65.6 on Spider test set.

3 METHODOLOGY

3.1 Context-free Grammar Pre-training

RAT-SQL utilizes the Syntactic Neural Model (SNM) proposed by (Yin and Neubig, 2017) to generate the SQL $P$. Yin etc. believe that existing methods treat code generation as a task of natural language generation, but the syntax of the target programming language is not considered. Unlike natural languages, programming languages, especially SQL, have strict grammar rules. According to these rules, SNM is an essential method which improves the accuracy of the model by limiting the search space of the decoder.

In addition, the basic framework of SQL grammar is context-free with the specific natural language description. For example, no matter what natural language description is, the first clause of SQL is always select, and the next clause is always from. The loss value in the initial training stage of RAT-SQL is extremely large, which mainly comes from $P$ errors generated by the decoder.

In view of the above situation, we propose a Context-free Grammar Pre-training (GP) method to pre-train the parameters on the decoder side. The semantic information of the encoder is replaced by zero vectors. The probability equation of RAT-SQL using LSTM to output a sequence of decoder actions is:

$$Pr(P|y) = \prod_t Pr(a_t|a_{<t}, y)$$  

where $y$ is always $[0]$ in the stage of GP and $a_{<t}$ are all previous actions. The LSTM’s state updating
strategy will be modified correspondingly as:

\[
\begin{align*}
    m_t, h_t &= f_{LSTM}(a_{t-1} \parallel [0] \\
    &\parallel h_{p_t} \parallel a_{p_t} \parallel n_{f_t} \parallel m_{t-1} \parallel h_{t-1})
\end{align*}
\]

where \( m_t \) and \( h_t \) is the LSTM cell state and output in step \( t \), \( a_{t-1} \) is the embedding of the previous action, \( p_t \) is the step corresponding to expanding the parent AST node of the current node, and \( n_{f_t} \) is the embedding of the current node type. We use \([0]\) to replace the former \( z_t \) that obtained by using multi-head attention on \( h_{t-1} \) over \( y \).

Since GP no longer depends on semantic information, it cannot predict column names or table names. In order to not change the framework of RAT-SQL, it is assumed that each sample has only one column and one table, therefore

\[
\begin{align*}
    Pr(a_t = SelectColumn[0]|a_{<t}) = 1 \\
    Pr(a_t = SelectTable[0]|a_{<t}) = 1
\end{align*}
\]

To prevent overfitting, the number of decoder Grammar Pre-training steps is limited as 300.

### 3.2 Question-Schema Serialization and Encoding

We generally adopt the serialization method of RAT-SQL. Because the utilized pre-trained semantic model is GRAPPA, the question tokens are preceded by \(<s>\) and end up with \(</s>\).

Then, columns and tables are spliced in sequence according to the order of the schema provided by Spider dataset, and we use \(<s>\) as the separator.

As mentioned in (Lin et al., 2020), modeling with only table/field names and their relations is not always enough to capture the semantics of the schema and its dependencies with the question. Notably, we append values to mentioned columns only if they exactly match the question. For the example in Figure 2, the keyword \textit{volvo} in the question appears in both column \textit{make} and column \textit{model}, respectively. Therefore, the token \textit{volvo} has a Column-Part-Match(CPM) relationship with column \textit{make} and has a Column-Exact-Match(CEM) relationship with column \textit{model}. Intuitively, Exact Match has a greater probability as the correct column. In order to strengthen this relationship, we put \textit{volvo} after the column \textit{model} during serializing while column \textit{make} not. The sequence can be converted as

\[
S = \langle s \rangle, Q, \langle /s \rangle, C_1, \langle /s \rangle, C_2, V_2, \langle /s \rangle, ..., T_1, \langle /s \rangle, T_2, \langle /s \rangle, ..., \langle /s \rangle
\]

In RAT-SQL, the vector representation of a column or a table is the average of the first and last token. Experiments show that this encoding method may lose important information, so another method is used by computing the average of all tokens’ vector of the column or table. If a column is followed by a value, the representation of the column is calculated by all column tokens and value tokens, as shown in Figure 3.

### 3.3 Flooding

In deep learning, it often occurs that training loss keeps decreasing while the validation loss suddenly starts to rise. (Ishida et al., 2020) proposed a simple and tricky loss function \textit{flooding} to make validation loss continue decreasing:
Figure 3: An illustration of encoder model

\[ \tilde{J}(\theta) = |J(\theta) - b| + b \] 

(6)

where \( b > 0 \) is the flooding level specified by the user, and \( \theta \) is the model parameter. It is assumed that to a certain extent, the existence of parameter \( b \) can prevent the model from falling into the local optimum during the optimization process. However, unsuitable \( b \) usually lead to gradient explosion.

4 EXPERIMENTS

4.1 Experimental Setup

The Adam optimizer (Kingma and Ba, 2014) with default hyperparameters is adopted. In the stage of GP, learning rate is set to \( 7.44 \times 10^{-4} \). Due to GPU memory limitation, we set \( bs = 3 \) and \( num\_batch\_accumulated = 4 \), where \( bs \) and \( num\_batch\_accumulated \) are the gradient accumulation parameters of RAT-SQL, that equivalent to batch size of 12. Because of GP and a smaller batch size, comparing to RAT-SQL, we adjusted the initial learning rate of GRAPPA from the original \( 3 \times 10^{-6} \) to \( 2 \times 10^{-6} \), and the initial learning rate of other model parameters from \( 7.44 \times 10^{-4} \) to \( 5.44 \times 10^{-4} \). The rest of setups are the same with RAT-SQL.

4.2 Dataset and Metrics

Spider (Yu et al., 2018c) is a large-scale complex and cross-domain text-to-sql dataset. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains.

The metric adopted to evaluate model performance is Exact Match Accuracy proposed by (Yu et al., 2018b). This metric measures the model’s performance on without generating values.

4.3 Results

While RAT-SQL and GRAPPA have been open sourced, the offline result is worse than that announced on the leaderboard in our experiments, as shown in Table 1. The reason can be explained by random seed or equipment differences. In this section, we mainly compare model performance based on offline results.

| Model                  | Leaderboard | Offline |
|------------------------|-------------|---------|
| RAT-SQL+Bert           | 69.7        | 66.7    |
| RAT-SQL+GRAPPA         | 73.4        | 71.7    |

Table 1: Comparison between offline and leaderboard results on dev set
Figure 4: Grammar Pre-training loss value curve

Figure 5: Loss curve comparison between with GP and without GP

**GP** Figure 4 shows that in first 50 steps of GP, the training loss drops significantly, then remains at about 53. To prevent overfitting, the number of Grammar Pre-training steps is limited, even if the loss is still dropping in a tiny speed. We then use the pre-trained decoder to train our model, the training loss is maintained at a stable level compare to without GP, as shown in Figure 5.

**Flooding** Equation 6 shows that there is a extra parameter $b$ in loss function, and the model performance is extremely sensitive to $b$ and learning rate $lr$, a slightly larger $b$ may cause the model to gradient explosion during training. Table 2 shows several examples about different parameter combination, $\emptyset$ means that the combination of this parameters will cause the gradient to explode.

| $b$       | $lr$         | bert_lr | Dev.     |
|-----------|--------------|---------|----------|
| 0.1       | $7.44 \times 10^{-4}$ | $3 \times 10^{-6}$ | $\emptyset$ |
| 0.2       | $5.44 \times 10^{-4}$ | $2 \times 10^{-6}$ | $\emptyset$ |
| 0.02      | $5.44 \times 10^{-4}$ | $2 \times 10^{-6}$ | 70.6 ± 0.6 |
| 0.01      | $5.44 \times 10^{-4}$ | $2 \times 10^{-6}$ | 71.4 ± 0.7 |

Table 2: The influence of different parameters $b$ and $lr$ on the results. $\emptyset$ means that the combination of this parameters will cause the gradient to explode.

**Serialization with value** By using the method that append a value after the related column, there is a slight reduction in column selection errors.

Table 3 shows the improvements of Flooding(Fld.), Serialization with value(val.) and GP, respectively. The best result is 73.1 on Dev. offline.

| model                               | Dev.     |
|-------------------------------------|----------|
| RAT-SQL+GRAPPA                      | 71.5 ± 0.2 |
| RAT-SQL+GRAPPA with Fld.            | 71.4 ± 0.7 |
| RAT-SQL+GRAPPA with Fld. val.       | 71.8 ± 0.6 |
| RAT-SQL+GRAPPA with Fld. val. GP    | 72.5 ± 0.6 |

Table 3: Our final results. Fld. means Flooding, val. means serialization with value.

### 5 CONCLUSION

The final result on Spider is 72.8 on Dev. and 69.8 on Test. Compared to the result of RAT-SQL+GRAPPA, the Dev. and Test. results of RAT-SQL+GRAPPA+GP is more closer, which means that our model is more robust, as shown in Table 4. Moreover, tuning parameters is a complex and delicate task, the slightest difference is a thousand miles away. The most influential hyperparameters improve model performance, the results are not stable, where best result may be as high as 72.1, and the lowest result may be only 70.7 even if we use the same parameters.
is learning rate, when other parameters are exactly the same, a tiny difference in the learning rate will lead to completely different results. We believe that our model still has great potential, but we still need to find suitable hyperparameters.

## REFERENCES

Christopher Baik, H. V. Jagadish, and Yunyao Li. Bridging the semantic gap with sql query logs in natural language interfaces to databases. 2019.

Ana-Maria Popescu, Alex Aramanasu, Oren Etzioni, David Ko, and Alexander Yates. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics, pages 141–147, Geneva, Switzerland, aug 23–aug 27 2004. COLING. URL https://www.aclweb.org/anthology/C04-1021.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In Proceedings of the 21st international conference on World Wide Web, pages 639–648, 2012.

Fei Li and HV Jagadish. Constructing an interactive natural language interface for relational databases. Proceedings of the VLDB Endowment, 8(1):73–84, 2014.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103, 2017.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. Typesql: Knowledge-based type-aware neural text-to-sql generation. arXiv preprint arXiv:1804.09769, 2018a.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. arXiv preprint arXiv:1810.05237, 2018b.

Ben Bogin, Matt Gardner, and Jonathan Berant. Global reasoning over database structures for text-to-sql parsing. arXiv preprint arXiv:1908.11214, 2019.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. arXiv preprint arXiv:1905.08205, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. Grappa: Grammar-augmented pre-training for table semantic parsing. arXiv preprint arXiv:2009.13845, 2020.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. arXiv preprint arXiv:1802.05365, 2018.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. arXiv preprint arXiv:2004.07347, 2020.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingming Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. arXiv preprint arXiv:1809.08887, 2018c.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. arXiv preprint arXiv:2005.08314, 2020.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianzhe Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. Editing-based sql query generation for cross-domain context-dependent questions. arXiv preprint arXiv:1909.00786, 2019.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. Advances in neural information processing systems, 28:2692–2700, 2015.
Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. arXiv preprint arXiv:1911.04942, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.

Takashi Ishida, Ikko Yamane, Tomoya Sakai, Gang Niu, and Masashi Sugiyama. Do we need zero training loss after achieving zero training error? arXiv preprint arXiv:2002.08709, 2020.

Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. arXiv preprint arXiv:1704.01696, 2017.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. arXiv preprint arXiv:2012.12627, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.