Summary of Natural Language Generated SQL Statements

Sen Yang¹, and Yang Liu*¹

¹Computer College, University of Defense Technology, Changsha, 410073, CN
*Corresponding author’s e-mail: liuyang_@nudt.edu.cn

Abstract. The entry cost of database query SQL statement is high, which is difficult for most database users. Therefore, natural language automatic generation of SQL sentences has gradually become the leading research direction in recent years. Previous researchers have done much excellent work in this area. This article mainly summarizes generating SQL statements in common natural language at present and conducts a detailed analysis. We summarize the current problems faced by SQL statements’ natural language generation and look forward to future development trends.

1. Introduction
With the development of industrial technology, a large amount of data has begun to be information. For example, electronic medical records, electronic files, and electronic menus are gradually widely used in various fields. Relational databases are usually used to store these massive amounts of digital data to facilitate management and operation, and maintenance. However, many people are not familiar with database languages and have certain thresholds for use. However, how to query the required information from these relational databases through natural language has become one of the most popular research directions in natural language processing. That is to say, how to convert human natural language query description into executable database query statement SQL.

The main task of the natural language query interface is that the main task of the natural language query interface is NL2SQL. NL2SQL mainly involves two aspects of tasks, one is the process of encoding natural language queries and text in database data tables, and the other is converting the encoding results into SQL statements to implement NL2SQL. The key of NL2SQL's mission is to eliminate the differences in expression and structure between natural language queries, the structure and content of data tables in the database, and SQL statements. The difficulty lies in how to map the query intent in the natural language query to the standard description in the database to construct an accurate executable SQL statement. Previous workers have made many efforts in this regard. Therefore, this article mainly outlines related technologies in the field of natural language generation of SQL statements. And we analysed the future development trend.

2. NL2SQL based on pipeline method
Although the natural language query interface has been developed for a long time, it is still challenging to understand the semantics of natural language queries accurately. Early NL2SQL needed to formulate corresponding grammatical and semantic rules for different databases manually, but this method lacked mobility and scalability [1]. It is more inclined to build an adaptive NL2SQL interface that has nothing to do with the database's structure and content.

The process of converting natural language into intermediate expressions in the pipeline method relies on the regular description of natural language queries, so it cannot handle some complex and
changeable natural language descriptions. In recent years, with the continuous development of deep learning technology, neural network models have become more common to process NL2SQL. Its advantage over the former is that the diversity of natural language descriptions does not limit it, and it can also extract critical semantic information from complex expressions.

The pipeline method usually uses some intermediate expressions to convert natural language queries into SQL. Unger et al. [2] proposed in 2012 to transform natural language queries into a query template called SPARQL and use WordNet to fill the slots in this query template, as shown in figure 1. Jagadish et al. [3] proposed the NaLIR system in 2014. The system uses an intermediate expression of a "parse tree", which is fed back to the user and the user selects the "parse tree" that best matches their ideal query. The decoder converts the "parse tree" selected by the user into SQL. The entire process requires the user to interact with the system to determine the final result, and the degree of automation is not high. The above-mentioned pipeline-based methods [2][3] to define their semantic coverage in advance clearly, and there are also some rule restrictions on natural language query description, so they cannot flexibly deal with users' questions in different ways and have certain limitations.

The method based on the pipeline is to transform natural language query into a kind of interrogative expression and then transform these interrogative expressions into SQL statements [4]. Popescu et al. [5] proposed a system called PRECISE, which categorizes natural language queries through a classification method to better deal with contextual relations semantically and measure the generality of these query problems empirically. PRECISE uses WordNet [6] to mark the grammar of a natural language and map the words in the dictionary to complete the generation of SQL statements. It is tough to understand natural language. Even in the communication between people, ambiguity sometimes arises. Therefore, we cannot expect a natural language query interface to operate without errors. The user will get a wrong result because the system misunderstands or processes the natural language query, so it is impossible to judge whether the query result is what they want. To take a step back, even if the user realizes that the result is wrong, there is no way to improve SQL statements generated by the system. To solve this problem, Li et al. [7] proposed a system called NaLIR in 2016. The system uses a human-computer interaction method. When the system cannot clearly understand natural language queries, it will return a request. The user must eliminate the ambiguity by returning multiple possible candidate expressions to the user for selection. In this way, the user does not need to reorganize the query expression language, which reduces the user's burden.

The advantage of the pipeline method is that it does not require a large number of "natural language query-SQL statement" pairings because obtaining a large amount of labeled data is expensive and time-
In 2019, Jagadish et al. [8] used the historical query log in the SQL statement system to build a TEMPLAR system, as shown in figure 2. These historical query logs only contain SQL statements without their corresponding natural language query descriptions, thus avoiding a lot of manually labeled work. The system uses the information in the historical query log to improve SQL statement generation's effect, including how to select the correct database table name and column name. TEMPLAR can be added as an extension layer to those mentioned above pipeline-based natural language query interface system. The pipeline-based method's disadvantage is that it cannot handle some complex and changeable natural language descriptions. It can only handle a few relatively fixed expressions, and it relies on pre-defined templates and manual design features, so the field migration is also poor. The expression and grammatical structure of Chinese natural language queries are inherently changeable and complex. Therefore, the pipeline technology is relatively limited.

![Figure 2. The overall architecture of an NLIDB augmented with TEMPLAR [8]](image)

3. **NL2SQL based on deep learning method**

In recent years, with the continuous development of deep learning technology, neural networks have become more and more mainstream in generating SQL statements for natural language queries. The method based on deep learning has many advantages over the method based on the pipeline. For example, the pipeline method needs to convert natural language query into an intermediate expression. In this process, semantic information may be lost and rely on manual templates. Moreover, features and methods based on deep learning do not have these problems. Methods based on deep learning can be divided into weakly supervised learning and supervised learning according to how much training data is used.
3.1. Weakly supervised learning
As discussed above, statistical methods based on deep learning cannot avoid the need for a large amount of training data. However, it is challenging to obtain labelled data such as "natural language query-SQL statement" pairing. The use of weakly supervised learning means that a relatively small number of "natural language query-SQL statements" can be paired for model training. It can liberate researchers from the complicated work of collecting and labelling data.

With the continuous development of deep learning technology, neural network models can better solve the complex and changeable problem of natural language description. Early research treated NL2SQL as a sequence generation problem and used the Seq2Seq model [9] to complete this task. Li et al. used the Seq2Seq+Attention model [10] to improve SQL generation accuracy through attention enhancement methods. Wang et al. used the Seq2Seq+Copying model [11] to locate a part of the input natural language description through the copy machine and use it directly as the corresponding part of the output SQL statement. The copied part mainly includes the columns of the data table in the natural language description. Name and some string or a numeric value. However, the general Seq2Seq model does not consider that there are still specific rules in SQL statements' format and syntax, so the generated SQL may produce grammatical errors. Therefore, Xu et al. [12] proposed an SQLNet model in 2017, using a pre-defined SQL query template, and then predicting and filling its various parts. Compared with the Seq2Seq model, this method can greatly reduce the output SQL query's syntax errors, thereby improving the accuracy.

Liang et al. [13] proposed the MAPO (Memory Augmented Policy Optimization) model, a method based on weakly-supervised learning. MAPO turns the NL2SQL task into a reinforcement learning task based on natural language query characteristics to SQL statement and some basic elements of reinforcement learning. The main idea of reinforcement learning is based on the interactive learning of agent and environment. The agent influences the environment through action, and the environment returns to state and reward. MAPO adopts different actions for a particular environment. The main idea is to express the objective function of maximizing the expected return as a weighted sum of two items: one is the expectation of the high return trajectory in the memory buffer, and the other is the expectation of the high return trajectory in the memory buffer. MAPO is the first to apply the reinforcement learning method to converting natural language queries to SQL statements and achieves the desired effect. Although the MAPO algorithm applies reinforcement learning to natural language queries to generate SQL statements, although the desired effect can be achieved, there are two problems: sparse returns and false returns. To address these two problems, Agarwal et al. [14] proposed corresponding solutions.

3.2. Supervised learning
Although weakly supervised learning has the advantages of fast training data collection and low cost of data labeling. However, through sufficient training data paired with "natural language query-SQL statement" for supervised learning, the model obtained will be even better in effect. Supervised learning also has a lot of related research work.

Dong et al. [15] used the Encoder-Decoder deep learning neural network structure for the first time to process natural language queries into a logical expression or SQL statement. Two deep learning models are proposed in the paper. The first model, Seq2Seq, simply deals with sequence generation, while the second model, Seq2Tree modifies the decoder into a hierarchical tree structure to capture SQL statements The structure information.

Simultaneously, the paper also proposed the Seq2SQL [16] model to handle this task. The model decomposes the task into three sub-problems, including aggregation operations, SELECT column names and WHERE clauses:
- Predict the aggregation operation of natural language queries through the classification model.
- Use the classification model to predict the SELECT column names.
- Use a pointer network to predict the conditions in the WHERE clause.
The first two parts of the prediction are supervised learning, using the cross-entropy loss function as the objective function, and the third part uses the policy gradient in reinforcement learning to deal with the WHERE clause's disordered nature.

Researchers introduce information such as the contents of some database tables or external knowledge to improve the accuracy of converting natural language queries to SQL statements. Yu et al. [17] proposed a TypeSQL method, which follows the idea of SQLNet based on template filling, using an external knowledge base to label entities in natural language queries, and using the labeling results as a feature and word embedding Spliced together as the input of natural language coding, and uses Bi-LSTM model coding. The database column name is used as the query in the attention mechanism, and the previous encoding results are used as the key and value. The attention mechanism is used to obtain the natural language query and the entity information contained in the column name's weighted condition. In the end, the results of each subtask are improved compared to the SQLNet model. At present, most of the existing studies have not considered the data table structure in the database. Song et al. [18] proposed a method called HSRNet, which is a pattern-aware neural network. HSRNet models the data table structure as a hierarchical pattern map and encodes this information into sentence expressions through a neural network. Compared with the end-to-end model, HSRNet decomposes the task into three modules, including a column name prediction module, a template decoder, and a detailed improvement module to adjust the intermediate process. Experimental results show that this decomposition structure can significantly improve accuracy.

4. Conclusion
This article provides a comprehensive overview of the field of converting natural language into SQL statements. The pipeline method and the deep learning method are distinguished, and the existing methods and the evolution process are respectively explained. At present, the conversion of natural language into SQL statements has been widely used in intelligent databases, and significant progress has been made. However, it also faces many challenges: unable to recognize complex sentences and reasoning. It performs well in simple statements but performs poorly in complex statements and designing multiple databases. In the future, the conversion of natural language to SQL statements may be devoted to solving the problems of handling the conversion of complex natural language to SQL statements, improving query speed, and reducing training costs.

References
[1] Küpper, Detlef, Strobel, M., & Rösner, Dietmar F. (1993). Nauda: a cooperative natural language interface to relational databases. Acm Sigmod Record, 22(2), 529-533.
[2] Unger C, Bühmann L, Lehmann J, et al. (2012). Template-based question answering over RDF data. Proceedings of the 21st World Wide Web Conference, ACM, 639-648.
[3] Li, F., & Jagadish, H. V. . (2014). Constructing an interactive natural language interface for relational databases. Proceedings of the Vldb Endowment, 8(1), 73-84.
[4] Li, F., & Jagadish, H. V. . (2014). Nalir: an interactive natural language interface for querying relational databases.
[5] Popescu A M, Etzioni O, Kautz H. (2003).Towards a theory of natural language interfaces to databases. Proceedings of the 8th international conference on Intelligent user interfaces. ACM. 149-157.
[6] Miller, G. A., Richard, B., Christiane, F., Derek, G., & Miller, K. J. (1990). Introduction to wordnet: an on-line lexical database*. International Journal of Lexicography, 3(4), 235-244.
[7] Li, F., & Jagadish, H. V. . (2016). Understanding natural language queries over relational databases. Acm Sigmod Record, 45(1), 6-13.
[8] Baik, C., Jagadish, H. V., & Li, Y. . (2019). Bridging the Semantic Gap with SQL Query Logs in Natural Language Interfaces to Databases. 2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE.
[9] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. Computer ence.

[10] Dong, L., & Lapata, M. (2016). Language to Logical Form with Neural Attention. Meeting of the Association for Computational Linguistics. Office for Official Publications of the European Communities.

[11] Wang C L, Brockschmidt M, Singh R. (2018). Pointing out SQL queries from text. Microsoft Research: Artificial Intelligence.

[12] Xu X J, Liu C, Song D, et al. (2018) SQLNet: generating structured queries from natural language without reinforcement learning. arXiv:1711.04436.

[13] Liang, C., Norouzi, M., Berant, J., Le, Q., & Lao, N. (2018). Memory augmented policy optimization for program synthesis and semantic parsing.

[14] Agarwal R, Liang C, Schuurmans D, et a1. (2019) Learning to Generalize from Sparse and Underspecified Rewards. Proceedings of the 36th International Conference on Machine Learning. 130-140.

[15] Dong, L., & Lapata, M. (2016). Language to Logical Form with Neural Attention. Meeting of the Association for Computational Linguistics. Office for Official Publications of the European Communities.

[16] Zhong V, Xiong C, Socher R. (2017) Seq2SQL: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv: 1709.00103.

[17] Mellah, Y., Ettifouri, E. H., Bouchentouf, T., & Belkasmi, M. G. (2020). Text2SQLNet: Syntax Type-Aware Tree Networks for Text-to-SQL. Innovation in Information Systems and Technologies to Support Learning Research.

[18] Liu, T., Luo, F., Xia, Q., Ma, S., & Sui, Z. (2019). Hierarchical encoder with auxiliary supervision for neural table-to-text generation: learning better representation for tables. Proceedings of the AAAI Conference on Artificial Intelligence, 33, 6786-6793.