Title: Relational Database Based on Natural Language

CS Engineers Team: He Zhang, YiChiao Yang,
Kun Chen, Kai Shi, Xiang Li

1 Introduction

Natural language processing is an important research field of Human-computation interaction(HCI). NLP could build a bridge between computer and people to communicate with each other. In Lunar system, it can answer questions about rock samples from moon by using a program which utilizes an Augmented Transition Network (ATN) parser and Woods' Procedural Semantics [1]. Gauri Rao proposes the architecture to translate NL into SQL based on semantic grammar [2]. A.M. Popescu describes a theoretical framework called PRESCISE, which introduces the concept of Semantically Tractable Sentences that are sentences whose semantic interpretation is done by the analysis of some dictionaries and semantic constraints [3]. Fei Li utilizes dependency parser, parser tree node mapper and parse tree structure adjustor to build NaLIR system [4]. A.O. Enikuomehin proposes a time saving algorithm for human language queries from RDB [5]. In [6] a python framework called Quepy in introduced, which can create SPARQL query from NL.

In our project, we attempt to build two algorithms based on keyword searching to convert English queries to SQL used in relational databases. The first algorithm is so-called precise mapping algorithm, we build lexicons for attributes, values, tables and actions. It compares input words with lexicons word by word before creating SQL. If database system has lots of attributes, it takes system lots of time to process. In this situation, we propose an improved algorithm, in which we do precise mapping for the attributes except the primary key and improve the range searching and multiple conditions functions searching. The running time can be greatly decreased in the second algorithm.

2 Proposed Algorithms

There are two algorithms proposed in this project. The first one is called precise mapping algorithm, and the second one is called improved algorithm. Both algorithms are based on keyword searching. Java on eclipse IDE is utilized to realize these algorithms.

2.1 Related Lexicons

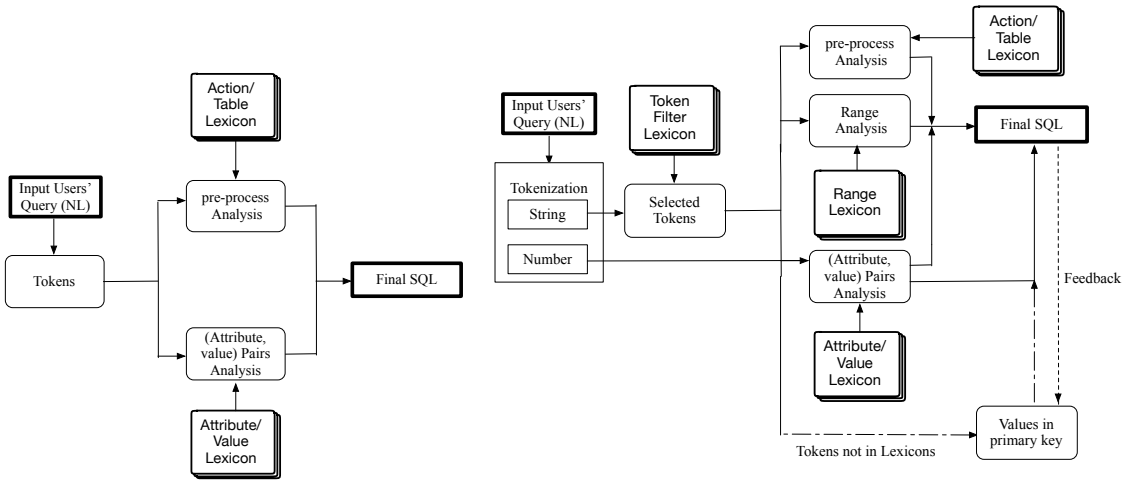
Lexicons are created to store the keywords (selected tokens) in the process of mapping. In total there are four types of lexicons. Token filter Lexicon normally contains verb, adjective, Adverb., which is used to delete irrelevant tokens and improve efficiency. Action/Table Lexicon is used to map the operation keywords such as select, delete, and names of table. Range Lexicon can capture the keywords such as not, larger, between, etc., which all can be used in range selection.

Attribute/Value lexicon includes attributes and corresponding values. For precise mapping algorithm, its Attribute/Value lexicon contains all attributes and their values, but for improved algorithm, its lexicon will exclude the primary key and values in primary key.

2.2 Description of algorithms

Fig.1 shows the flow charts of these two algorithms. In precise mapping algorithm, when we have an input users' query, it will firstly change into individual tokens, then these tokens will be mapped to

Action/Table Lexicon and Attribute/Value Lexicon word by word in two analysis parts. As precise mapping algorithm can be seen as a part of improved algorithm, so we will describe the mapping process in detail in the improved algorithm.



(a) Precise Mapping Algorithm (b) Improved Algorithm

Fig.1 Flow charts of two algorithms

In a table in relational database, we know values in primary key are distinct, so the number of these values may be vary large, but for other attributes, their values may have repetition. E.g., if the attribute is gender, it may obtain two values, male and female. So the improved algorithm only searches the attributes and values except the primary key, and this algorithm uses method of exclusion to find the values belong to primary key from users' query.

In improved algorithm, when there is users' input NL for query, the sentence is tokenized by regular expression method into two parts, string tokens and number tokens. They are processed separately, which improves efficiency. For string tokens, they would be checked and reduced by token filter lexicon and after that, be mapped by three other lexicons. If mapped, they would be deleted from token set. The rest tokens would be recognized as primary key values. For number tokens, they are directly sent for attribute value mapping. All mapped tokens are used to form SQL query with rules introduced by examples later. If there is a chance that left tokens are not primary keys, feedback is designed to deal with this situation.

Comparing with precise mapping algorithm, improved algorithm has two advantages. Firstly, its running time is excepted to be smaller. There is no need to match the primary key in the lexicons one by one, by using this way it saves much time. Also, it can support more functions, such as range selection, "select all" function. Second advantage is that it's generic and efficient. Some recent researches are invisible in translation part. It's like a black box.

3 Examples

For verification and testing running time, a table in RDB is built with 4 attributes shown in Table 2, there are Name (Primary key), department, gender and grade. In this case it is obvious that last three

attributes contain dramatically fewer values. ICARD of the attributes in the table are as follows: Name, 20,000; Department, 10; Gender, 2; Grade, 100.

Table 1: Table of Student

PRIM:Name	Department	Gender	Grade
Lily	CS	Female	80
Lucy	ECE	Female	70
Mike	BIO	Male	68
Jack	ECO	Male	90
…	…	…	…

3.1 Example 1-Range Select

Input NL: select student who is male and grade is not above 70

Step1: In Tokenization part, NL is divided into String part and number part. String - select student who is male and grade is not above; Number – 70.

Step2: In the Selected Token part, the token filter lexicon is used to delete all irrelative words. So tokens such as “who”, “is”, “and” are removed from the string.

Step3: Then Pre-process Analysis, Range Analysis and (Attribute, Value) Paris Analysis parts will become active and use their corresponding lexicons to group the action, range, attributes and their values. The tokens which not belong to any of these lexicons will be will be grouped in values in primary key part automatically. In this case, there is no word left, so the values in primary key part will be set in NULL. Fig.2 shows the mapping process.

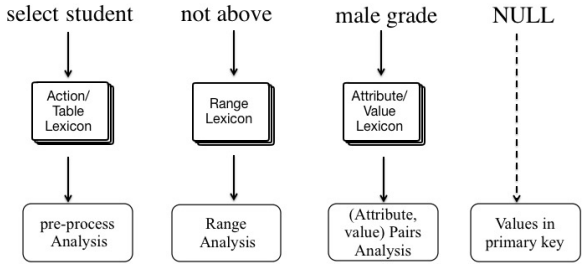


Fig.2 Mapping process of selected tokens in ex.1

Table 2: Table of student

Attribute	Value
PRIM:name	NULL
(gender)	Male
grade	70

Step4: In the pre-process analysis part, SQL query is select. Then we should create an attribute and value table. Like this table shown in Table 2, because the value in primary key is NULL, so the name goes after SELECT. We only have one table in this example, so the table name student goes after FROM. According to this table, the gender = male. And grade <= 70 go after WHERE. So this sentence will be changed into SQL query as follows.

SELECT	X1, X2 …	→	SELECT	name	
FROM	FROM TABLE		FROM	student	
WHERE	Cond1, Cond2…		WHERE	gender=male AND	grade<=70

3.2 Example 2-Value in Primary Key

Input NL: select grade whose name is Lucy

The second improvement of the new algorithm is we do not have primary key in Attribute/Value lexicon. This example will show this improvement. Look at this sentence above. Still the same, we will delete useless words. Then three lexicons are used to group the selected tokens, as shown in Fig.3. As we know, lucy cannot be mapped to the three lexicons in Fig.3, then lucy will go to values in primary key part automatically. Also as shown in Table 3, the value of grade is NULL, so grade goes after SELECT. The name of the table goes after FROM. In our algorithm, lucy will link to primary key automatically. So the name = lucy goes after WHERE.

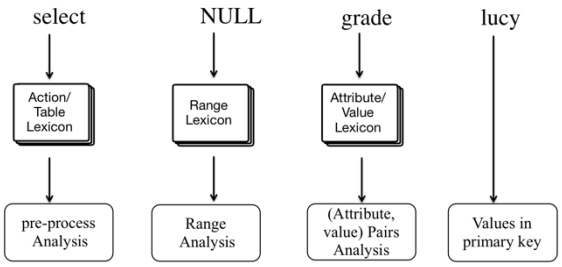


Fig.3 Mapping process of selected tokens in ex.2

Table 3: Table of student

Attribute	Value
PRIM:name	lucy
grade	NULL

3.3 Running Time Analysis

For the most important part- efficiency part, let's compare the running time of two proposed algorithms in Fig.4. Improved algorithm is the orange one, and precise mapping algorithm is the blue one. Data is the number of primary key. Even though the table becomes larger from 20 to 20000, the improved algorithm can always only spend less than 10ms. But our previous algorithm presented explosive growth. It's obvious improved algorithm has high efficiency.

Data	Algorithm1 (4Attributes)	Algorithm2 (4Attributes)
20	27.531414	8.594459
200	30.876006	8.906872
2000	40.438085	9.087095
20000	101.220836	9.310834

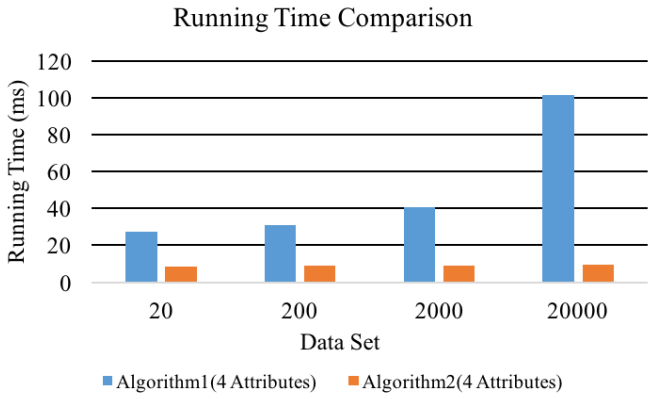


Fig.4 Comparison of RT

4. Demo

Link: https://www.youtube.com/watch?v=gVOU4LSH41g&feature=youtu.be

Fig.5 shows three examples in on-line demo.

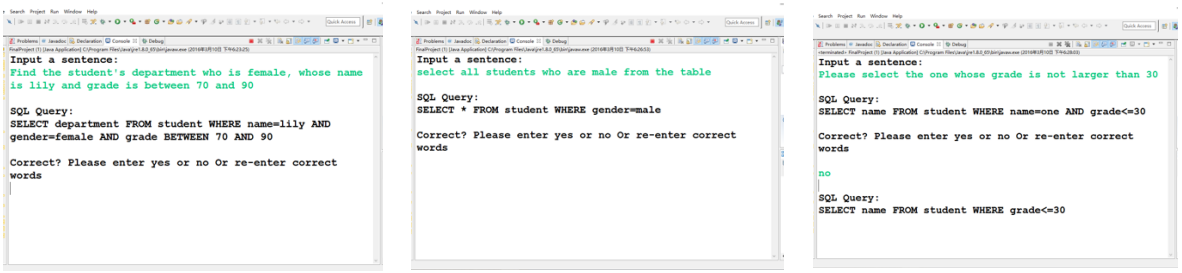


Fig.5 Demo

5. Conclusion and Future Work

5.1 Conclusion

Two algorithms based on keyword searching are proposed in this project. Lexicons are built to store the keywords appeared in the tables in RDB. The proposed precise mapping algorithm can do mapping of NL to lexicons word by word and this process is time-consuming. So on the basis of precise mapping algorithm, we propose an improved algorithm, which only do precise mapping for attributes excluding primary key, and can automatically match the values in primary key from NL. And improved algorithm has more functions such as range searching, multiple conditions searching and users' feedback.

There are mainly two advantages of our project. Firstly, it breaks the limitation of the database. Traditional database cannot interact with NL. Our API makes it more intelligent. The second advantage is that it's extendable and portable. Extendable means it's easy to support other language, such as Chinese, Spanish. Portable means it's easy to support different types databases.

5.2 Future Work

For future work part, fault tolerance module can be added, which enables the algorithms to recognize spelling mistakes and grammar mistakes. For example, if typing student to "studen", the program can still return the correct SQL query.

Another goal is to improve the efficiency of token filter lexicon. Word-parsing module can be added to alien words. Also, semantic analysis part is considered. It makes NL to be more similar to SQL query. E.g., it can translate the sentence "What is grade of Tom" into "Grade of student with name is Tom", which is similar to SQL query "select grade from student where name = Tom".

6. Reference

[1] W. Woods, R. Kaplan, and B. Webber. The Lunar Sciences Natural Language Information System, Final Report, Technical Report 2378, Bolt Beranek and Newman Inc., 1972.

[2] G. Rao, C. Agarwal, etc. Natural Language Query Processing Using Sematic Grammar. IJCSE, Vol. 2, No. 2, pp. 219-223, 2010

[3] A.M. Popescu, O. Etzioni, and H. Kautz. Towards a Theory of Natural Language. Interfaces to Databases. Proceedings of the 8th international conference on intelligent user interfaces –IUI '03, pp. 149-157, 2003.

[4] F. Li, H.V. Jagadish. Constructing an Interactive Natural Language Interface for Relational Databases. Proceedings of VLDB Endowment, Vol. 8, Issue 1, pp.73-84, 2014.

[5] A.O. Enikuomehin and D.O. Okwufulueze. An Algorithm Solving Natural Language Query Execution Problems on Relational Database. International Journal of Advanced Computer Science and Application IJACSA, Vol. 3, No. 10, pp. 169-175, 2012.

[6] QUEPY. http://quepy.machinalis.com/