Minor Project

Palla Sai Bharath

July 5, 2020

Introduction

Converting natural language sentences into SQL queries are a big deal in today's world. Claims are made in papers and documents whose data is dependent on relational databases. Here we come up with a solution which is primitive and helps to convert simple English language sentences into SQL queries. The approach is simple, we read the sentence, tokenize it following up with stemming and other methods to convert it into an SQL query.

Tokenizing, stemming

We first tokenize the sentence and remove all words that add no extra meaning to the sentence. Words like "that, the, for, ... ". We also convert meaningful words into symbols during tokenization. Words like "Less than, Greater than, in-between" are marked as comparators and will be used to create a semantic structure in later compile stages.

Also the initial parts of the code, reads the table and associates which keywords appear in their corresponding tables. Keywords appearing in more than one table are marked for equi-joins. Once all words are either as symbols, tables and literal values, compilation begins.

Let the sentence be:-

student who got more than 10 and less than 45 marks with roll_no more than 50-marks.

Now the tokenizer outputs the following list as the intermediate tokens:-
('student', '>', '10', '<', '45', 'marks', 'roll_no', '>', '50')
('tab', 'sym', 'val', 'sym', 'val', 'col', 'col', 'sym', 'val')

Compilation

Once the above two lists are generated, the algorithm then checks if it is a single table or requires joining of two tables, keep in mind that once we can deal with one join between two tables we can extend it to as many tables as we want. The symbols, values are matched with a corresponding columns in the table and a dictionary is created. One can see this, when one runs the program "interpreter.py" in this directory.

Once the dictionary is created, the program can easily start converting the semantic structure from dictionary to an SQL query. Below is a sample dictionary created by the program which is used later to convert it into SQL query.

{'tab': {'list': ['student'], 'conds': [{'name': 'marks', 'preds': [['>', '10'], ['<', '45']]}, {'name': 'roll_no', 'preds': [['>', '50']]}]}, 'join': False, 'conds': [ ], 'cols': ['marks']}

Compiler

Once this dictionary is created, we can convert it into an SQL query this ease. The output of the function dic_to_table when the input is above dictionary is given below:-

SELECT marks FROM student WHERE marks > 10 AND marks < 45 AND roll_no > 50

Limitations

This program is a bridge between natural language and SQL query. Though the inputs are not high level English language sentences, they are weakly structured sentences, which can be converted into SQL statements within no time. Also note that this does not limit the language to be English, the only semantic binding done to words is at symbols where words like 'Less than' is linked to '<', if one has a dictionary that converts any language word into a symbol also should work.
Also since the column of interest in not known, we use a hyphen('-') to tell the program which columns we are interested in. Also the natural language sentences which need join, highlight the tables to be joined by asterisks between two tables to show what the join is on.

Example:-
... *student, teacher* ...
This means, there is a natural join on the two tables 'student' and 'teacher'.

This interpreter bridges the gap between a very weakly semantic sentence structure to an SQL query, though these sentences are not how people speak or write, even people with no programming experience should be able to write this Semi-English sentences to convert them into SQL queries.