International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 5, ISSN No. 2455-2143, Pages 278-282
Published Online September 2020 in IJEAST (http://www.ijeast.com)

CONVERTING NATURAL LANGUAGE QUERY TO SQL QUERY

Buddhaditya Rath
Department of Computer Engineering
University of Mumbai, Mumbai, Maharashtra, India.

Abstract— **This program aims to develop a system that converts a natural language statement into a SQL query to obtain information from the relevant database. The native language input statement taken from the user is transmitted through various natural language processing techniques such as Tokenization, Parts of Speech Tagging, Stemming and Lemmatization to get the statement the way you want it. The statement is also processed to determine the type of question. The final question is done by changing the basic categories and conditions in their question form and combining the question mark with the basic question. Currently, the system only works with the Oracle SQL database. We are exploring ways to use native English words - sentences or fragments of sentences - to extract data from SQL, a small experiment of natural language with machine language problems. We aim to contribute to the goal of strong natural language in the data recovery system.**

I. INTRODUCTION

Indigenous Language Processing is an area under Artificial Intelligence used to build smart computers that can communicate with a person as a person. It closes the gap of the human-machine. The main purpose of the Indigenous Language Quiz is to translate English sentences electronically. Apart from all the challenges, it is widely used for research purposes. Natural Language Processing can be used to access the database by asking queries in Natural Language and getting the required results. Asking questions in native language for information is a very simple and easy way to access data, especially for users who are not familiar with the complex language of handling question languages such as SQL (Structured Question Language). There are many challenges in converting a natural language query into a SQL query as an ambiguous meaning that one word can have more than one meaning. In this case, one-word maps in more than one sense. Another challenge is the construction of a complex SQL query and the next challenge is about Discourse information where the previous sentence affects the translation of the next sentence for example if a user enters the SELECTION and INSTALLING simultaneously, such a case is not understood in the system.

II. SCRIPTURE STUDY

The problem we address is a subcategory of a broader problem; natural language to machine language. SQL is opportunistic for its distinctive, high level language and close connection to the underlying data. We utilize these characteristics in our project. SQL is tool for manipulating data. To create a system which can generate a SQL query from natural language we need to make the system which can understand natural language. Most of the research done until now solves this problem by teaching a system to identify the parts of speech of a particular word in the natural language which is called tagging. After this the system is made to understand the meaning of the natural query when all the words are put together which is called parsing. When parsing is successfully done then the system generates a SQL query using proper syntax of Oracle SQL.

III. EXAMINING THE SYSTEM

The existing method for the query from SQL information manually But some improvements have been made in recent years to help with questions that require the use of Probabilistic Context-Free Grammar (PCFG). The current standard used is QuePy and similar, non-integrated projects are the same. These programs use old techniques; QuePy has not been updated in over a year. The key website has an interactive web application to show how it works, which shows the area for improvement. key answers factoid questions as long as the structure of the question is simple. Recent research such as SQLizer provides algorithms and methods that can greatly improve current open source projects. However, the SQLizer website does not use natural English to question the feature found in their 2017 paper. We look forward to proving these innovations.

IV. EXISTING PROGRAM OR RESEARCH GAP

The following are some of the input types that can currently be managed by our system. Find the power of class number 3128 in building Taylor 3.

```
SELECT *
 FROM the classroom
```

WHEN classroom capacity = '3128' AND classroom construction = 'Taylor'

In this particular example, the system failed to determine whether to take 'classroom volume' or 'class number' as n-gram. Therefore, mapping failed.

Who teaches Physics?
SELECT *
FROM THE DEPARTMENT THERE
department name = 'Physics'

In this example, the included query module of our program is able to map Physics so that 'door name' is defined from the 'door' table. But it fails to identify the 'who' refers to the person (pastor). Our system combines column value references in the native language. It can hang trying to match the column value with the word in the schema.

V. PROBLEM STATEMENT & OBJECTIVES

PROBLEM STATEMENT: This project uses natural language processing techniques to work with text details to construct SQL queries with the help of a corporation we have developed. En2sql is provided in clear English language as input returns a well-structured SQL statement such as output.

OBJECTIVE: The aim of our project is to generate accurate and valid SQL queries after natural language analysis using open source tools and libraries. Users will be able to access the SQL statement of the 5 keywords by passing an English sentence or a piece of a sentence. We wish to do so in a way that advances current open source projects in terms of robustness and usability.

VI. SCALE

We will be using the question SELECT, INSERT, DELETE, UPDATE and WHERE categories. We hope to use many phrases such as joining, merging, ordering, limiting, etc. but we cannot commit to this very challenging due to their additional problem and difficulty of time. The study will begin with a focus on statements (“get / find number of employees”) and then extend to questions (“Who is Bob?”). The statements are easier as it gives more keywords to interpolate the SQL query structure. There are many relations databases. While their SQL syntax is similar, it can differ for more complicated queries. We will focus on Oracle SQL as it is an open source database with a large user base.

VII. SUGGESTED SYSTEM

The proposed approach aims to use SQL information to create a corpus that will help identify SQL command names namely SELECT, INSTALL, DELETE, UPDATE, and map token with the appropriate POS. Word matches will be calculated with input schema tokens (table names, column names, data) to enter table names, column names, and data comparisons in question.

VIII. DATA ANALYSIS & DISCUSSION

DATASET: We will build our corpus by scanning the schema of the table name, column name, column types, key relationships, and data. This will be a specific schema database. The other corpus will contain all the necessary building materials. It will contain words that are likely Select, Add, Delete, Where to help create a question by entering. We also use the POS corpus of Stanford and WordNet corpus with nltk.

SETUP: Using En2SQL you will need to track packages. Python3, NLTK, pymysql, POS Tagger [14] Stanford, Oracle MySql, and Yelp SQL Dataset [13]. You will first need to set up a MySQL data user, and then upload the Yelp SQL data to the database. We include a POS tagger with a code. Launch the requirements.txt file (via pip3) to enter the python package requirements (nltk and pymysql). Update data contact details in the db.config.py file. Enter the native language questions in input.txt, each line. Launch the main.py. file.

RESULT & ANALYSIS: The corpus that can be used to test our system is not easily accessible and depends on the database. Therefore, we tested our system on an integrated computer for bank language-related language statements and university details. The university and banking website contains 11 and 6 tables respectively. However, the system can work on any complex database. The natural language statement should be one sentence. The system has been tested on a campus of about 75 of the university's native language statements and 50 related to the bank database. System accuracy is found to be approximately 86%. The program offers the same SQL query as output where the same native language statement is represented in different ways. If the system fails to generate a SQL query that matches any native language statement, an error message is displayed. Here are a few of the results provided by the program on the university's corpus:

i. Find the student name where instructor name is ’Crick’.
SELECT DISTINCT student.stud name FROM instructor INNER JOIN advisor ON instructor.ID = advisor.instID INNER JOIN student
ON student.ID = advisor.stud ID WHERE instructor.name = ’Crick’

In this database, the tables ’student’ and ’instructor’ are linked through the table ’advisor’. So, we can see that this query deals with multiple tables which are joined by INNER JOIN.

ii. Find all student name whose credits are between 90 and 100 and department name is ’Finance’ or ’Biology’. SELECT DISTINCT student.stud name FROM student
WHERE (student.tot cred BETWEEN ’90’ AND ’100’) AND (student.dep name = ’Finance’ OR student.dep name = ’Biology’)

The above query showcases multiple conditions within the WHERE clause. This query also involves use of BETWEEN clause and logical clauses like AND, OR.

iii. List all student names whose credits are 50 in decreasing order of credits. SELECT DISTINCT student.stud name FROM student
WHERE student.tot cred = '50' ORDER BY student.tot cred DESC 7

Another type of query is the one involving sorting its result based on some attribute. For this purpose, the query uses the ORDER BY clause to sort the results in decreasing order.

iv. Give the department name where maximum salary of instructor is greater than50000.
SELECT DISTINCT instructor.dep name FROM instructor
GROUP BY instructor.dep name HAVING MAX(instructor.salary) >'50000'

In SQL, when an aggregate function is compared to constant, like in this case maximum of salary is compared to 50000, then the query involves use of HAVING clause instead of a WHERE clause. Also, whenever HAVING is used, the results are supposed to be grouped by the attributes in the SELECT clause.

v. Give the department name where salary of instructor is greater than average of salary.
SELECT DISTINCT instructor.dep name FROM instructor
WHERE instructor.salary > (SELECT AVG(instructor.salary)
FROM instructor)

This query showcases a special case of nested queries. Whenever an attribute is compared to the result of an aggregate function, i.e. in this case salary greater than average of salary, we have to use nested query.

vi. Find the course taught by Crick. SELECT DISTINCT teaches.course id FROM teaches
NATURAL JOIN instructor WHERE instructor.name = 'Crick'

Till now, we have seen cases in which an attribute associated to the value is mentioned in the natural language statement. In this case, we handle cases where attribute is not mentioned. We find out the most appropriate attribute for the given value.

vii. o Publish in alphabetic order the names of all instructors. o Give names of all the instructors in alphabetical order. o Give instructors names in ascending order.
SELECT DISTINCT instructor.name FROM instructor
ORDER BY instructor.name ASC

As seen in this example, there can be multiple ways of representing the same natural language statement. The system gives the same SQL query as the output when the same natural language statement is represented in different ways.

vii. Insert a student whose id is 5, name is Josh, department name is Physics and credits are 150.
INSERT INTO student (student.ID, student.stud name, student.dep name, student.tot cred) VALUES ('5' , 'Josh' , 'Physics' , '150')
In addition to the data retrieval queries, our system also provides a natural language interface to insert data into the database. Other DML queries such as UPDATE and DELETE are also provided by the system.

ABNORMAL CASE EXPLAINATION: Some input table names, column name contains underscore, short forms as a result of which it becomes unusual and difficult to distinguish between a stand word, a common word. So we should either add it to the corpus or talk openly before using it. The accuracy of the questionnaires indicates minute fluctuations. Some English statements are less instructive. Example: Who is Bob? This question when asked in a large database creates confusion to find the right answer. Sometimes it gives and takes away what is right and sometimes it gives a vague result.

IX. ALGORITHM DESIGN

Following will be our algorithm.

1. Scanning the database: Here we will scan the database to find table names, column names, primary and foreign keys.
2. Input: We will take the sentence as input from the user (using input.txt).
3. Tokenize and Tag: We will mark the line and use POS tagging to mark words.
4. Syntactic parsing: Here we will try to map the table name and column name with the given natural query. Also, we will try to identify different attributes of the query.
5. Redundancy Filtering: Here we will try to eliminate demolition such as if while creating a map we create a merger requirement and if not necessary then remove the additional table.
6. Query Formation: Here we will form a complete SQL query based on MySQL syntax.
7. Query Execution: Here we will execute the query on database to get results.

X. DETAILS OF HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements • 4GB RAM. • 10 GB HDD. • Intel 1.66 GHz Processor Pentium 4

Software Requirements • Windows XP/Vista or higher • Python 3.6.3.

XI. DETAILED DESIGN

Our project uses Python 3.6. Python has many libraries that are easily accessible and proven. All our required libraries support Python 3.6. Tools used for the NLTK3 python library will be used for input inputs. This library serves as a tool for computer language tools. The following is a list of modules we will use. The token module provides basic classes for processing certain text objects, such as words, or sentences. NLTK Tokenizer is used to make a token.

XII. DATA COLLECTION

For SQL domain information we will create a corpus that will contain exactly the same words as SQL syntax in SELECT, LIMIT, FROM, etc. This is common among open source projects we have seen. Most of the open source projects we have tested use such keywords, so finding a kind keyword will be easier. If our English to keyword mapping results are not favourable, we may use the online thesaurus API. The Oracle SQL database will be built with data from the public Yelp SQL Database [13]. We have chosen the Yelp Database because it is large enough, has many tables, and we have some domain information about Yelp. This data will be used as a corpus and test. The corpus will be constructed from table names, column names, table relationships, and column types. The corpus database will be used unattended to store the agnostic database. A set of architectural questions will be used as a starting point for questions. Indigenous language tokens will be compared to this.

XIII. OUTPUT AND TESTING

A program that removes a SQL architecture query that works in a database and attempts to answer an input question or statement. The output is displayed in standard output and in put.txt.

Testing our code will begin by creating a specific schema that contains table-related data, column name and column data. The other corpus will contain data related to the SELECT query command. And then we will give you a general natural language statement to test it. It will take the native language input and then use two copies and thus issue a SQL query. We will take the file for

extract the query and run it against the MySQL Yelp Database, test the functionality of the query. After the run, we will take the resulting data and compare it with our expected results. In the final test, we will check the question accurately and make sure the wrong question does not return the correct information. We will need to build a natural English set with the expected pairing of the output. If the question passes the first two default tests, you will need to be tested manually to be ready. If all these tests pass, the question is correct. With this test we will build the accuracy of the system.

XIV. SOLTION STRUCTURE

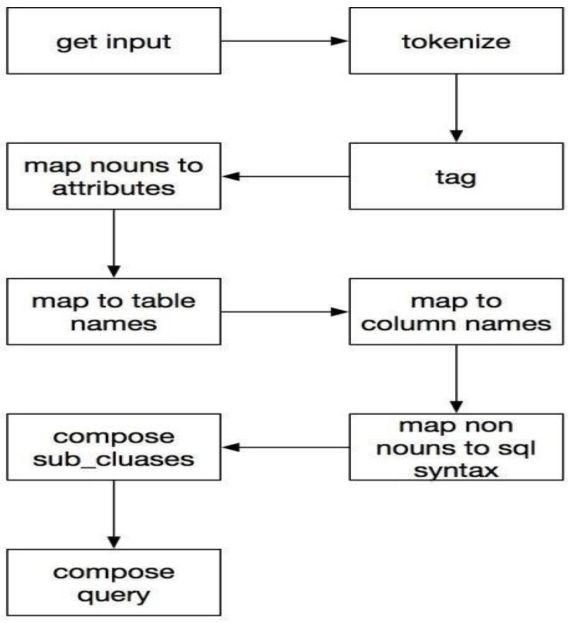

Figure 1: Solution Structure

XV. RESULT & DISCUSSION

EFFECT OF PSEUDO-CODE USE

We propose a system that seems to overcome the shortcomings of an existing system that adopts a natural language sentence such as input, which is transmitted through various NLP phases to form the final SQL query.

TOKENIZE AND TAG

The original language input query is divided into different tokens with the help of token, word token, from the 'NLTK' package. Token collections made with the tag of the speech part using the Stanford POS tag. All processes that follow this step use these tagged tokens to process.

ANALYZE TAGGED TOKENS

Depending on the tags of the previous step, the name map and action list are updated with a single iteration over the tokens. Tokens associated with affiliate activities are also assigned with their proper names using the pre-built alphabet. The decision as to whether a native language statement represents a data retrieval question (SELECT) or a DML question (INSERT, UPDATE, DELETE) is taken at this stage with the help of some 'data sorting' to indicate the type of question. For example, if words like ‘insert’ and similar specific words appear in the input, the question type is

'INSERT' and so on. For any type of query, the 'S' (SELECT), 'W' ('O', 'O' (ODA,) test tags are arranged in nouns that indicate their categories. For this, we have designed data dictionaries for various categories. These data dictionaries contain a token-clause term pair, e.g. The line data dictionary rate is "number": "COUNT", "count": "COUNT", "value": "SUM", "sum": "SUM", "average": "AVG", "means": " AVG. "Therefore, if any of these tokens are met, it is likely that they will have an integrated clause and the names will be properly marked with a clause.

WORD MANAGEMENT MAP

Using the noun map and verb list, the table set is prepared, which will hold the tables that are needed in the query to be formed. This is based on the fact that the table names are either nouns or verbs. The noun map is used to find the attributes which are needed in the final query. The attributes, the table associated with the attribute and the clause tag are stored in an attribute-table map which is used in the final stage of query formation. This is done using the string-matching algorithm that we have implemented in our system. The words in the input sentence need not exactly be as they are in the database. The stemmer and lemmatizer are applied on the words before they are matched using our string-matching algorithm. The data obtained during this step i.e. table set and attribute-table map, is most likely to be in the final query, however, it might be refined later.

FILTER REDUNDANCY AND FINALIZE CLAUSES OF THE QUERY

Using the various data dictionaries described, the system has already decided which categories are most likely to be in the final question and put the information into categories. However, some details should be completed in this section. The information related to the GROUP BY and HAVING clause is collected using previous data and basic SQL rules. For example, if the consolidation function is compared to a permanent, i.e. 'MAX (salary)> 40000', then the clause 'HAVING' should be used instead of the clause 'WHERE'. As mentioned in the previous step, data mining should be done. Here, obsolete tables and attributes are removed using other filtering algorithms. For example, one of the algorithms filters a table with its corresponding symbols that are the basis of a particular table in a table. e.g. if the table set has [table1, table2] and table1 has symbols [a1, a2] and table2 has [a1, a2, a3] after the previous steps, then table2 is sufficient to represent all the required attributes which is why table1 has been removed. There are various other algorithms used to filter the results and complete the table set and table of table symbols.

FORM THE FINAL QUERY & EXECUTE

Depending on the relationship between multiple tables, the decision of INNER JOIN or NATURAL JOIN is taken. For example, if there are two tables. If these two tables have one common feature and are called the same in both, then there is a NATURAL BETWEEN Tables. But if the standard attribute is named separately for both tables, then there is an INNER JOIN between the tables.

XVI. CONCLUSION

This project has given us a great opportunity to come up with a solution for writing boring questions. This project nevertheless helps to solve basic questions but over time it can be developed to manage complex questions, familiarity and can be expanded with NoSQL. We were able to read and use NLTK, cosine, tf-idf for python3. We found around 30-50% accuracy in basic queries.

XVII. REFERENCES

[1] A Natural Language Database Interface Based on a Probabilistic Context Free Grammar, IEEE International Workshop on Semantic Computing and Systems 978-0-7695-3316-2/08 $25.00 © 2008 IEEE DOI 10.1109/WSCS.2008.14.
[2] Domain Specific Query Generation from Natural Language Text, The Sixth International Conference on Innovative Computing Technology (INTECH 2016) 978-1-5090-2000-3/16/$31.00 ©2016 IEEE
[3] Generic Interactive Natural Language Interface to Databases (GINLIDB), Proceedings of the WSEAS International Conference on Evolutionary Computing ISSN : 1790-5109 ISBN : 978-960-474-067-3 Natural Language Interface to Database Using Modified Co-occurrence Matrix Technique, 2015 International Conference on Pervasive Computing (ICPC) 978-1-14799-6272-3/15/$31.00(c)2015 IEEE.
[4] Natural language to SQL Generation for Semantic Knowledge Extraction in Social Web Sources, Indian Journal of Science and Technology, Vol 8(1), 01- 1, January 2015 ISSN (Online) : 0974-5645 ISSN (Print) : 0974-6846 DOI : 10.17485/ijst/2015/v811/54123
[5] Natural Language Query Processing Using Semantic Grammar, Gauri Rao et al. / (IJCSE) International Journal on Computer Science and Engineering Vol.02, No.02, 2010, 219-223 ISSN 0975-3397.
[6] SQLizer : Query Synthesis from Natural Language, Proc. ACM Program. Lang., Vol. 1, No. OOPSLA, Article 63. Publication date : October 2017 .
[7] Synthesizing Highly Expressive SQL Queries from Input-Output Examples, PLDI'17, June 12-23, 2017, Barcelona, Spain ACM. 978-2-4503-4988-8/17/06…http://dx.doi.org/10.1145/3062341.3062365.