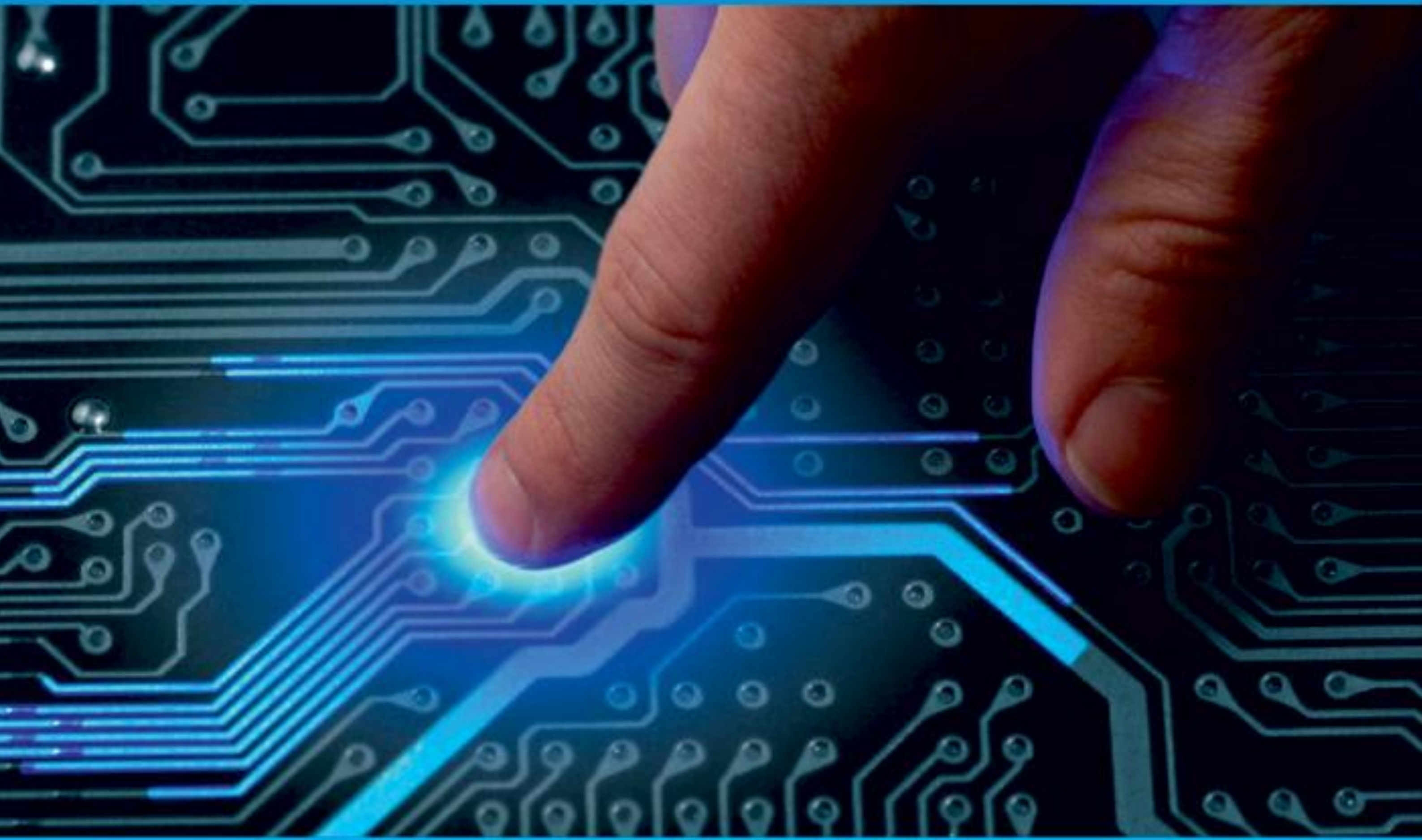
e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 7, July 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.542

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 7, July 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0907148 |

A Comprehensive Exploration on Spider With Fuzzy Decision Text-To-SQL Model

Shailaja D S[1], Dr. Praveen Kumar K V[2]

Department of Computer Science & Engineering, Sapthagiri College of Engineering, Bangalore, Karnataka, India[1]

Department of Computer Science & Engineering, Sapthagiri College of Engineering, Bangalore, Karnataka, India [2]

ABSTRACT: The challenge of natural language processing is from natural language to logical form (SQL). In this project, we present an fuzzy semantic to structured query language (F-SemtoSql) neural approach that is a fuzzy decision semantic deep network query model based on demand aggregation. It aims to address the problem of the complex and cross-domain text-to-SQL generation task. The corpus is trained as the input word vector of the model with LSTM and Word2Vec embedding technology.
A Database is an organized collection of data. It is created to store large amount of data and retrieve it efficiently in less amount of time. To retrieve data from the database Structured Query Language (SQL) is used. SQL has its own syntax of query. To retrieve correct data from the database this query should be written in proper or correct syntax. Thus users should have sufficient knowledge of SQL to retrieve data.

KEYWORDS: F-SemtoSql, Deep learning

I. INTRODUCTION

Three with the improvement of advanced computer, network, communication, and storage technology, massive interactive cross-domain data are creasing with an exponential rate. However, users can intuitively perceive behavioral events in a single shallow semantic parsing. And it does not address the need for users to deal with a large number of data events from deep semantic parsing and cross-domain data sources. At the same time, with the development of artificial intelligence technology, people's requirement for natural language query is increasing.It is no longer sufficient for simple data retrieval, but the system can respond quickly and efficiently to complex query retrieval requirements involving logic and semantics. The expression of natural language map to a machine-understandable structured query language by semantic parsing method, it has become a research hotspot. Text-to-SQL task is one of the most critical subtasks of semantic parsing in natural language processing (NLP).

Databases are very powerful means of storing and retrieving large amounts of data quickly and efficiently. There are many different commercially available database management systems used around the world. However getting data out of these databases is not an easy task. A special database interaction language called SQL (Structured Query Language) is used to communicate with these databases. Using Natural Language to communicate between a database system and its human users has become increasingly important since database systems have become widespread and their accessibility to non-expert users is desirable, if not essential, to facilitate full use of the database system. Natural Language to SQL translator (NLS-to-SQL) is aimed at reducing this complexity of database querying. First it is necessary to use a language that is understood by anybody, whether an expert database programmer or person with no computer knowledge. The best-suited language for this purpose is the English language. This means NLS-to-SQL has to translate English or natural language queries into SQL before retrieving data from database. Keeping this in mind I come up with a technique that converts a natural language statement to its equivalent SQL statement. To make "NLS-to-SQL Translator" more flexible a lightweight approach is used to convert the Natural Language input into its SQL equivalent. This Lightweight approach extracts certain keywords and indicators from the English query using the preprocessor and then using the post processor, generates the SQL statement. The another important feature of this system is it provides a method of updating the dictionary by which we can add new word or phrase that can be mapped further to its equivalent clause in SQL.

II. RELATED WORK

The existing approach is to generate the query from the knowledge of SQL manually. But certain improvement has done in recent years helps to generate more accurate queries using Probabilistic Context Free Grammar (PCFG). The current implemented standard is QuePy and similar, disjoint projects like them. These projects use old techniques; QuePy has not been updated in over a year. The QuePy website has an interactive web app to show how it works, which shows room for improvement. QuePy answers factoid questions as long as the question structure is simple. Recent research such as SQLizer presents algorithms and methodologies that can drastically improve the current open source projects. However, the SQLizer website does not implement the natural english to query aspect found in their 2017 paper. We wish to prove these newer methods.

III. SYSTEM MODEL

This Lightweight approach extracts certain keywords and indicators from the English query using the preprocessor and then using the post processor, generates the SQL statement. The another important feature of this system is it provides a method of updating the dictionary by which we can add new word or phrase that can be mapped further to its equivalent clause in SQL.
The proposed architecture is a standard NLP composed of Lexical, Syntactic, and Semantic analysis.

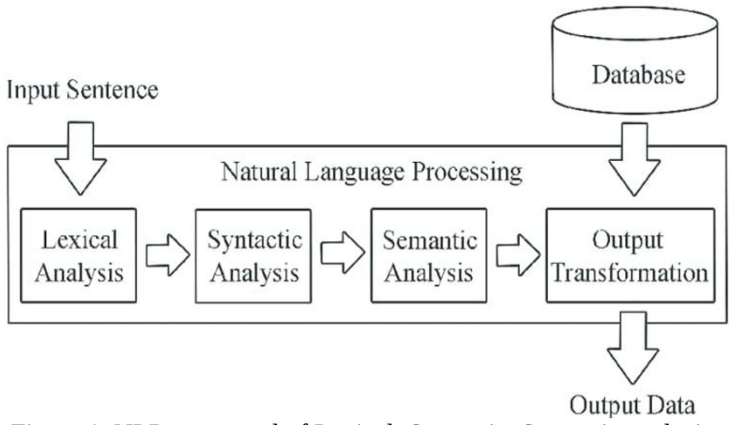


Figure 1. NLP composed of Lexical, Syntactic, Semantic analysis.

The modules included in our implementation are as follows

Data Collection
Stop words removal
Generation of SQL Query
Execution of Query

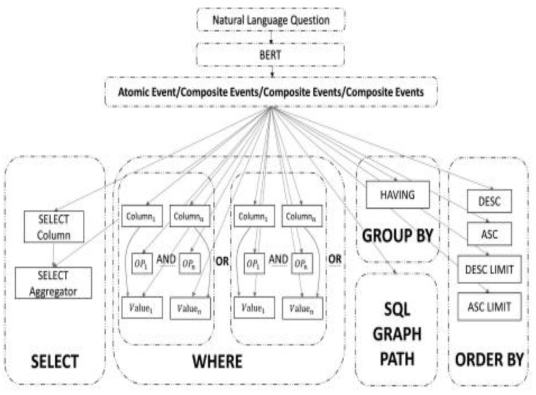


Figure. 2 System Architecture

PROPOSED SOLUTION

This Lightweight approach extracts certain keywords and indicators from the English query using the preprocessor and then using the post processor, generates the SQL statement. The another important feature of this system is it provides a method of updating the dictionary by which we can add new word or phrase that can be mapped further to its equivalent clause in SQL.

The modules included in our implementation are as follows

Pseudo Code:

Procedure Data Collection()

```
Begin:
Step1: Download dataset from Yelp SQL database
Step2:  Create database
Step3: for each file in data collection
        Step3.1: Read the SQL queries and load into the MySql database.
Step4: End for
End
```

Stop words Removal:

Input to this module is an English sentence. Here first the type of query is identified and based on it preprocessor replace numerals, substitute comparison operators and also remove apostrophe. After that the noun clauses (ie: column or table name in the English sentence) are extracted. This output of preprocessor i.e english sentence containing noun clauses, substituted numerals and comparison operators is given to the postprocessor. For this, Convert Word to Number, Noun Clause Extractor , Numerical Operator Substituter interfaces are created.

Procedure Stopword_Removal()

```
Begin:
Step1: Enter the normal sentence from GUI
Step2:  Read the sentences
Step3: Split sentences into words
Step4: for each word
        Step4.1: Replace numerals with comparison operators.
        Step4.2 Remove stopwords using nltk tokens
Step 5: return the cleaned set of words.
Step4: End for
End
```

Generation of SQL Queriess:

1. Scanning the database: Here we will go through the database to get the table names, column names, primary and foreign keys.
2. Input: We will take a sentence as a input from the user (using input.txt)
3. Tokenize and Tag : We will tokenize the sentence and using POS tagging to tag the 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. Filtering Redundancy : Here we will try to eliminate redundancy like if while mapping we have create a join requirement and if they are not necessary then we remove the extra table.
6. Query Formation : Here we will form a complete SQL query based on MySQL syntax.

Procedure Generation of SQLquery()

```
Begin:
Step1: Read the set of words from the previous module
Step2:  Scan for the database name, table name, column names
Step3: Apply the semantic parsing with the collected dictionary in dataset collection
Step4: Filter the redundancy of the parsing result
Step 5: Form the query.
Step4: End for
    End
```

With the domain level knowledge of SQL we will create a corpus which will contain words which are synonymous the SQL syntax to SELECT, LIMIT, FROM, etc. This is common among the open source projects we have seen. Many of the open source projects we have inspected use such keywords, thus coming up with a generous keyword corpus will be easy. If our english to keyword mapping results are not desirable, we may use an online thesaurus api. A MySQL database will be constructed with data from the public Yelp SQL Database. We chose the yelp dataset because it is fairly large, has a good amount of tables, and we have some domain level knowledge about Yelp already. This data will be used as a corpus and for testing. The corpus will be constructed from the table names, column names, table relationships, and column types. The database corpus will be used in an unsupervised manner to keep the program database agnostic. A set of substructure queries will be used as a starting point for the queries. The natural language tokens will be matched to these.

STOP WORDS removal

Input to this module is an English sentence. Here first the type of query is identified and based on it preprocessor replace numerals, substitute comparison operators and also remove apostrophe. After that the noun clauses (ie: column or table name in the English sentence) are extracted. This output of preprocessor i.e english sentence containing noun clauses, substituted numerals and comparison operators is given to the postprocessor. For this, Convert Word to Number, Noun Clause Extractor , Numerical Operator Substituter interfaces are created.

Generation of SQL query

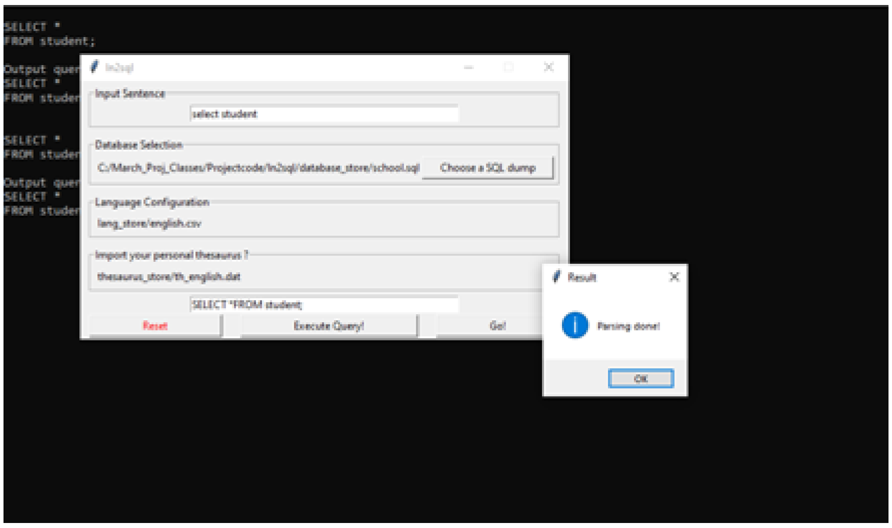
1. Scanning the database: Here we will go through the database to get the table names, column names, primary and foreign keys.
2. Input : We will take a sentence as a input from the user (using input.txt)
3. Tokenize and Tag : We will tokenize the sentence and using POS tagging to tag the words
4. Syntactic 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. Filtering Redundancy : Here we will try to eliminate redundancy like if while mapping we have create a join requirement and if they are not necessary then we remove the extra table. 6. Query Formation : Here we will form a complete SQL query based on MySQL syntax.

Query Execution: Here we will execute the query on database to get results

IV.PERFORMANCE EVALUATION

In this section, fig1 after loading the data from the dataset input sent as normal plain text and sent for generating a query.

Fig 3 constructs the query provided from the input and displays the output.



```
SELECT *
FROM student;

Output quer
SELECT *
FROM studer

SELECT *
FROM studer

Output quer
SELECT *
FROM studer
```

In2sql

Input Sentence: select student

Database Selection: C:/March_Proj_Classes/Projectcode/In2sql/database_store/school.sql

Language Configuration: lang_store/english.csv

Import your personal thesaurus ?: thesaurus_store/th_english.dat

SELECT *FROM student;

Result

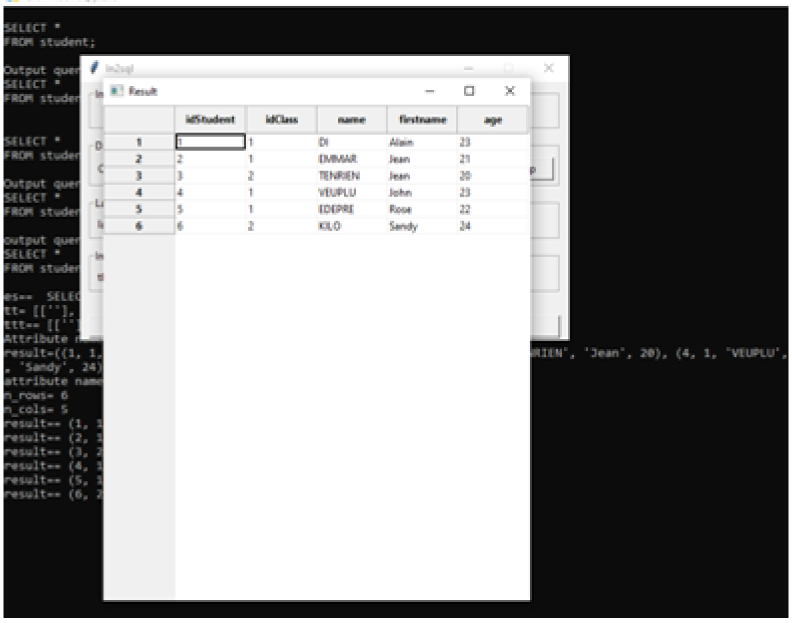
Parsing done!

Fig 3:- Normal query while parsing

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 7, July 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0907148 |



```
SELECT *
FROM student;

Output quer
SELECT *
FROM studer

SELECT *
FROM studer

Output quer
SELECT *
FROM studer

output quer
SELECT *
FROM studer

es==  SELEC
tt= [[''],
ttt== [[''
Attribute n
result=((1, 1,                                    RIEN', 'Jean', 20), (4, 1, 'VEUPLU',
, 'Sandy', 24)
attribute name
n_rows= 6
n_cols= 5
result== (1, 1
result== (2, 1
result== (3, 2
result== (4, 1
result== (5, 1
result== (6, 2
```

	idStudent	idClass	name	firstname	age
1	1	1	DI	Alain	23
2	2	1	EMMAR	Jean	21
3	3	2	TENRIEN	Jean	20
4	4	1	VEUPLU	John	23
5	5	1	EDEPRE	Rose	22
6	6	2	KILO	Sandy	24

Fig:-4 Results displayed after the sql execution

V.CONCLUSION

This project has given us a great opportunity to come up with an solution for writing tedious queries. This project though helps resolving basic queries but with time it can made powerful to handle complex queries, normalization and also can be extended for nosql. We were able to learn and implement NLTK, cosine, tf-idf of python3. We have got accuracy around 30-50% in basic queries.
In future work, we plan to deploy the proposed framework to real-world scenarios so as to further conduct practical evaluations of the proposed algorithm.

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
4. 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
5. 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
6. 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
7. SQLizer : Query Synthesis from Natural Language, Proc. ACM Program. Lang., Vol. 1, No. OOPSLA, Article 63. Publication date : October 2017
8. 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…$15.00 http://dx.doi.org/10.1145/3062341.3062365
9. "Sqlizer API." Easily Convert Files into SQL Databases | SQLizer. N.p., n.d. Web. 27 Feb. 2018.
10.Machinalis. "Quepy." A Python Framework to Transform Natural Language Questions to Queries. N.p., n.d. Web. 27 Feb. 2018.
11."Natural Language Toolkit." Natural Language Toolkit - NLTK 3.2.5 Documentation. N.p., n.d. Web. 27 Feb. 2018.
12.Alarfaj, Salah. "SalN3t/NlpSQL." GitHub. N.p., 11 Dec. 2017. Web. 27 Feb. 2018.
13.Yelp. "Yelp SQL Dataset." Yelp Dataset. N.p., n.d. Web. 27 Feb. 2018.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 7, July 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0907148 |

14.Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70. https://nlp.stanford.edu/software/tagger.shtml