A MINI-PROJECT REPORT

on

NLP2SQL

Submitted by

Akrati Chaturvedi
161500048
Manika Agrawal
161500304
Sajal Agrawal
161500475

Department of Computer Engineering & Applications

Institute of Engineering & Technology

Table of Contents

DECLARATION

We hereby declare that the work which is being presented in the Mini Project Report **"Title : NLP2SQL",** in partial fulfillment of the requirements for Mini project LAB, is an authentic record of our own work carried under the supervision of **Mr. Pankaj Sharma, Assistant Professor, GLA University, Mathura**.

Akrati Chaturvedi
161500048
Manika Agrawal
161500304
Sajal Agrawal
161500475

CERTIFICATE

This is to certify that the above statements made by the candidate are correct to the best of my knowledge and belief.

_____

Project Supervisor

Mr. Pankaj Sharma

Assistant Professor

Institute of Engineering and Technology

Department of Computer Science and Applications

Date:

ACKNOWLEDGEMENT

I would like to thank Mr. Pankaj Sharma, Assistant Professor at GLA University, Mathura for providing the guidance on Natural Language Processing. We work under his guidance and supervision. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us.

This project would never have seen the light of day without the help and guidance we have received.

Akrati Chaturvedi
161500048
Manika Agrawal
161500304
Sajal Agrawal
161500475

1. INTRODUCTION

1.1 OVERVIEW AND MOTIVATION

Databases are becoming more common and are becoming increasingly important in the current applications and websites. They often have to be used by people without a high competence in the matter and do not exactly know their structure. This is why translators for natural language to SQL queries are developed. Unfortunately, most of these translators are confined to a single base because of the specificity of the architecture of it.

For many years, databases (DB) are inevitable for all Web sites or applications that manage large amounts of information, such as user accounts (banks, transport agencies, social networks, video games, etc.). Internet has gradually democratized and popularized but the databases for their part remain abstract for many people. Some positions requiring no training or IT data administration still require working closely with databases, such as accounting or secretarial for example. This is in order that a person having no competence in the field of management DB, cannot directly administer, but at least understand how it works, interact with it and above perform simple tasks (query, add, delete), the translators of the natural language database query language emerged. For fifty years the problem of querying a natural language database is recurring and is the subject of much research. Most developed tools are very efficient but unfortunately for most compatible with a single source natural language and / or one target database. They are developed solely to be the interface a database and are exclusively compatible with it. Because of structure, vocabulary and extremely different naming convention from one base to another, porting non-compatible multi-tool bases on a different basis from that provided initially is difficult and would make anyway ineffective. It is from this observation that this article aims to design a translator for querying any database. Having defined some concepts and presented the state of the art will describe how to extract the information related to a target database in order to know its structure and vocabulary, then cross this information with the keywords of question, and generate output equivalent SQL query most likely. The request will be generated based on the presence, number and order of

keywords identified in the sentence entered by the user. To fine will present the evaluation of our approach, comparing the capabilities of our application to those already existing applications and evaluating performance on a set of test queries.

1.2 OBJECTIVE

The objective of our project, NLP2SQL is to generate accurate and valid SQL queries after parsing natural language using open source tools and libraries. Users will be able to obtain SQL statement for the major 5 command words by passing in an English sentence or sentence fragment. We wish to do so in a way that progresses the current open source projects towards robustness and usability.

We investigate avenues of using natural English utterances - sentence or sentence fragments – to extract data from an SQL, a narrower inspection of the broader natural language to machine language problem. We intend to contribute to the goal of a robust natural language to data retrieval system.

1.3 SUMMARY & SIMILAR APPLICATION

This project makes use of natural language processing techniques to work with text data to form SQL queries with the help of a corpus which we have developed. En2sql is given a plain English language as input returns a well-structured SQL statement as output.

The existing approach is to generate the query from the knowledge of SQL manually. But certain improvement done in recent years helps to generate more accurate queries, which are as follows:

QuePy - 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.

SQLizer – 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.

2. PROJECT DESIGN

This report includes Data Flow Diagrams (DFD), Use Case Diagram, Sequence Flow Diagrams and Algorithms Flowcharts to represent the design of our project.

2.1 DATA FLOW DIAGRAMS

LEVEL 0 : DATA FLOW DIAGRAM

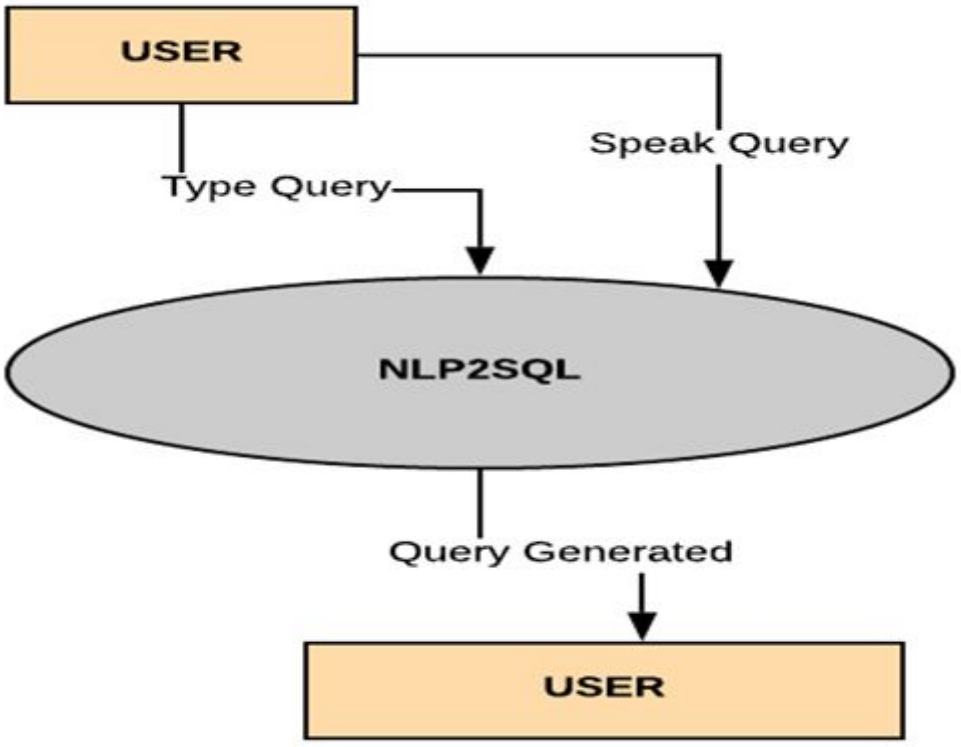

Fig 1: Level 0 - Data Flow Diagram

LEVEL 1 : DATA FLOW DIAGRAM

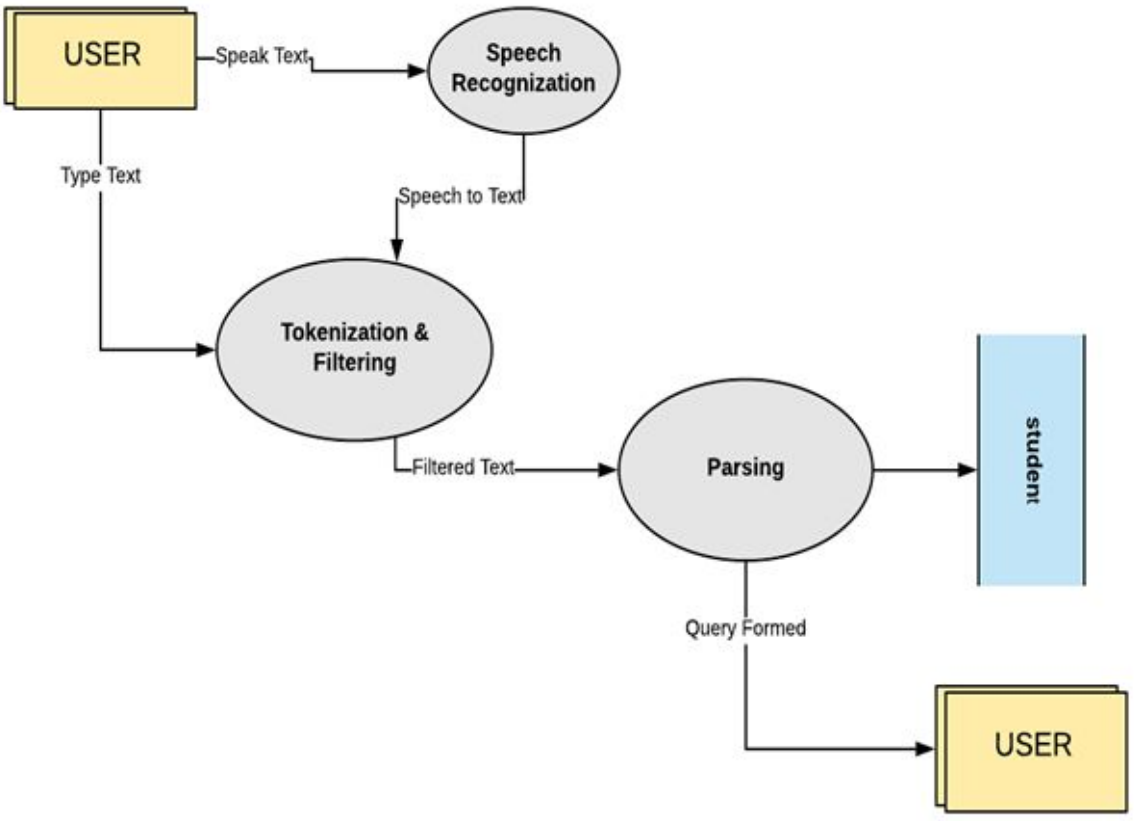

Fig 2: Level 1 - Data Flow Diagram

2.2 USE CASE

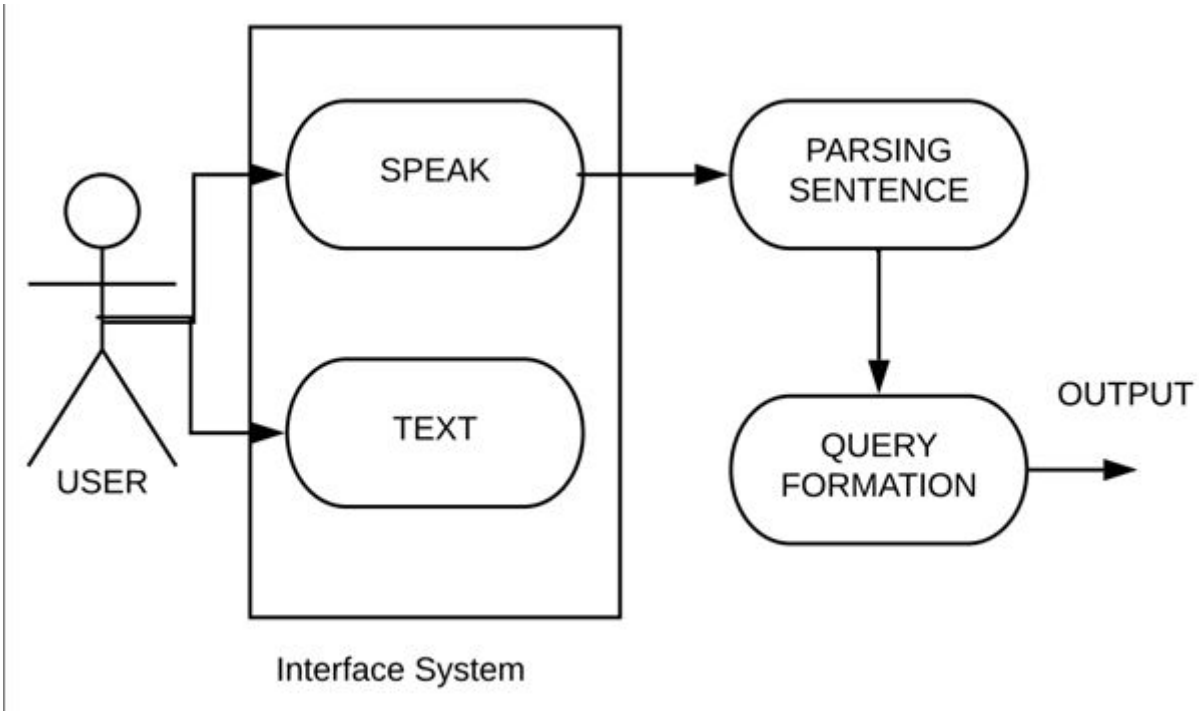

Fig 3: Use Case

USE CASE	DESCRIPTION
Speak :	User can speak the query through microphone of the system
Text :	User can write the query
Parsing Sentence :	The query as string is tokenized into words and then filtered to form SQL query
Query Formation :	The filtered query is passed through this algorithm and the query is formed as output query

Table 1: Use Case Description

2.3 SEQUENCE DIAGRAM

USER INTERACTION

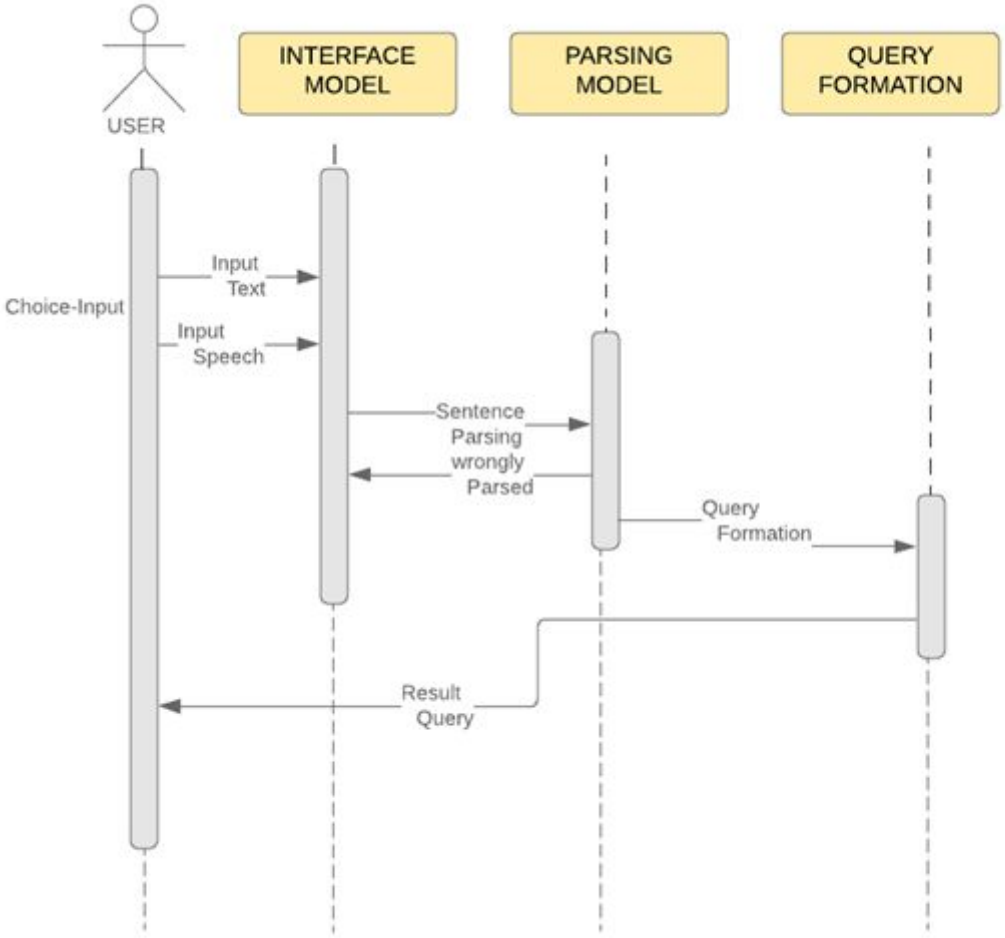

Fig 4: Sequence Diagram

2.4 ALGORITHM FLOWCHART

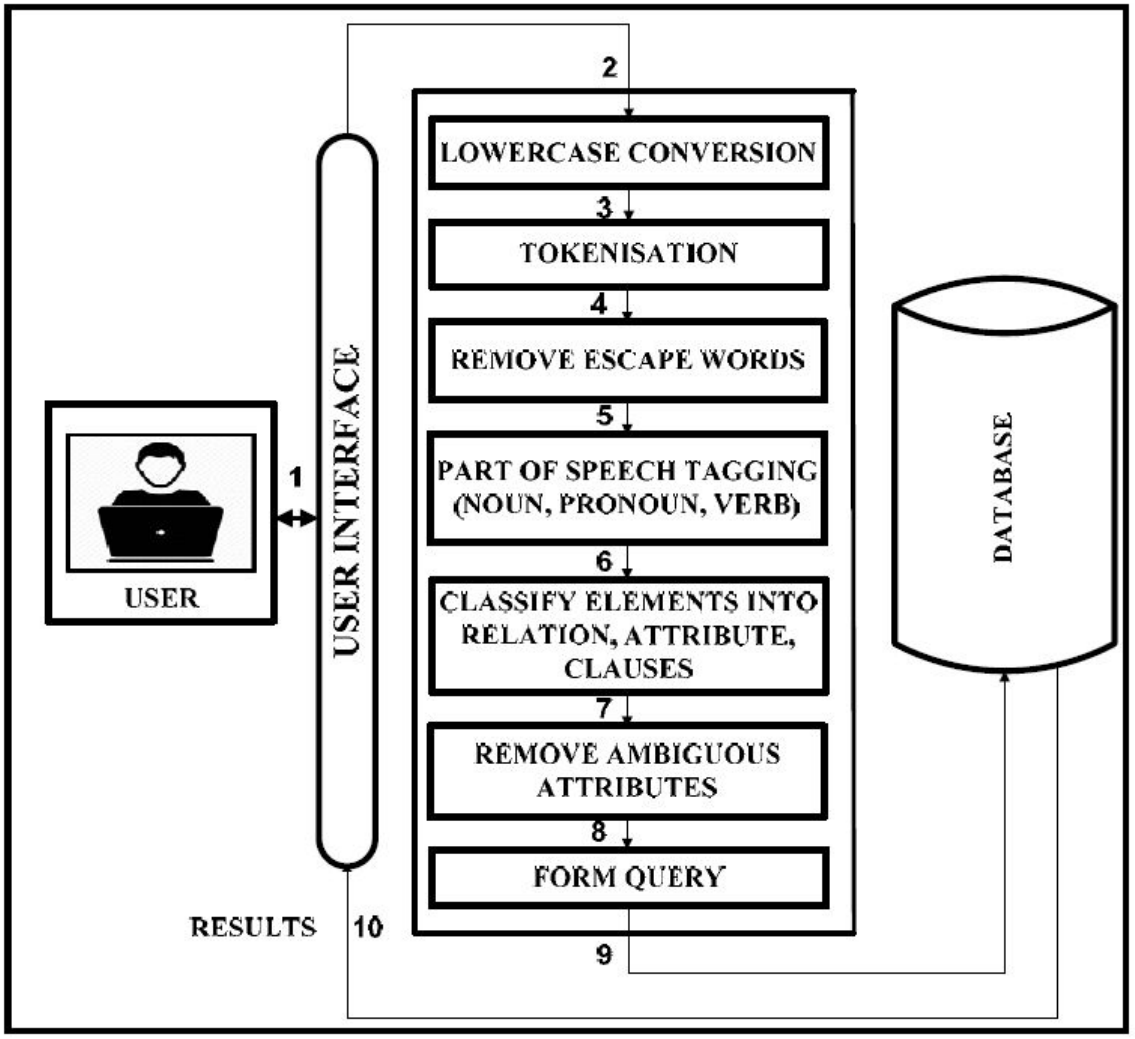

Fig 5: Algorithm Flowchart

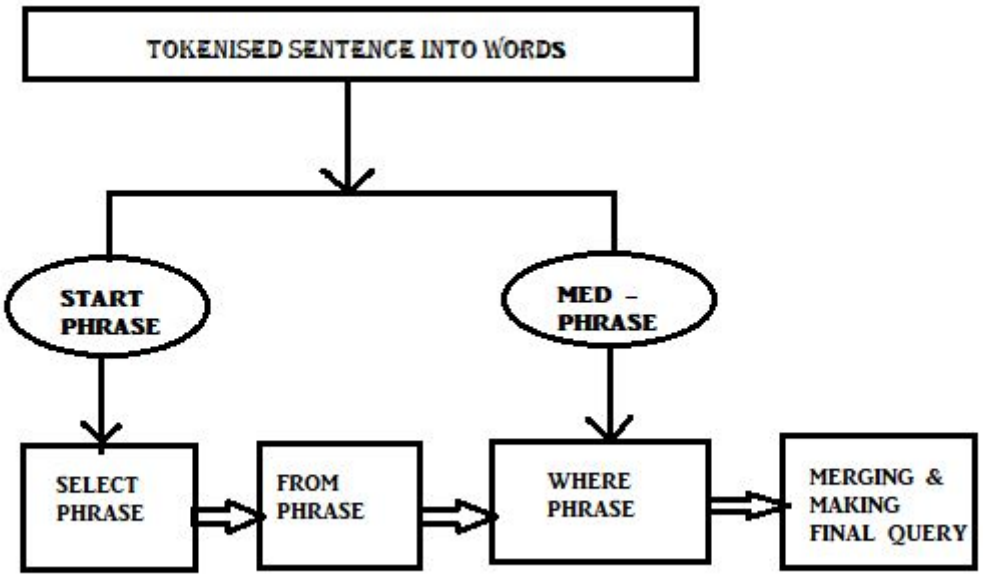

Fig 6 : Parsing Phrase to analyse query type

2.5 DETAILED DESCRIPTION OF DATABASE TABLE

Name of Field	Data Type	Explanation
rollno	int	University rollno of the student (PK)
name	int	Name of the Student
phone	varchar	Contact No. of Student
section	char	Section of Student

Table 2- student table

3. IMPLEMENTATION AND USER INTERFACE

This project was implemented as follows -

ALGORITHM

- Accept the input from the user either in the form of speech and convert it to text or directly in the form of text.
- If you take the input in speech, then convert it to text by Speech Recognition using Android.
- Split the input query and store it in a list, i.e. tokenize the input sentence.
- Find all the attributes of all the tables.
- Examine the query and find the table present in the query and the attributes present in the query.
- Find the attributes which belong to table present in the query.
- Find the attribute which do not belong to the table in the query (if any).
- Now find the tables which will contain the pair of ((attribute which do not belong to the table in the query), (other attributes present in the table in the query)).
- Select any one table. Thus we will obtain the tables required for natural join.
- For a natural join query, find out the common attribute of the 2 tables and form the inner query.
- Then form the outer query according to the different conditions.. Merge both of them and generate the final query. For a simple query, generate the final query by checking the different conditions accordingly.
- If there are 2 tables, then perform a natural join on the 2 tables with appropriate attributes of the tables.
- Obtain the conditions of the where clause (single condition or multiple condition by finding the ―and ‖ word in the input query), aggregate function (checking whether any aggregate function (like sum, avg, count , etc) present

in the query) and the relational operators between the conditions from the list of attributes. Add these to the final query.

- Print the final query.
- Write the generated SQL query in a ―log file‖ along with an index number and time of the query using the system clock, so that we can retrieve the query generated at a particular time giving the time as input.

NATURAL LANGUAGE PROCESSING (NLP)

Natural language processing is a field of computer science concerned with the interactions between computers and human (natural) languages.

These include spoken language systems that integrate speech and natural language.

The goal of NLP is to enable communication between people and computers without resorting to memorization of complex commands and procedures.

NLTK [Natural Language Toolkit]

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, part-of-speech tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

SPEECH RECOGNITION LIBRARIES

1. SpeechRecognition [Used to recognize the speech and convert it into text]
2. Pyaudio [PyAudio is required if and only if you want to use microphone input]

VISUAL STUDIO

Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

GITHUB

Github is an open source code repository and collaborative development platform.It offers a location for online code storage and collaborative development of massive software projects. The repository includes version control to enable hosting different development chains and versions, allowing users to inspect previous code and roll back to it in the event of unforeseen problems.

Our github link for the project **https://github.com/3musk/nlprepo1.git**

USER INTERFACE

Using Command Prompt

```
>>>
== RESTART: C:\Users\Akrati\Desktop\nlprepo1\nlprepo1\nlprepo1\__init__.py ==
Welcome to NLP to SQL. Generate your SQL query from natural language

Enter your choice
1. Speech
2. text
3. Exit
2
find all student
['find', 'all', 'student']
start phrase ['find', 'all']
from phrase ['student']
where phrase []
columns of slect []
column of where []
SELECT * FROM student
Continue using text option(y/n):
n
Enter your choice
1. Speech
2. text
3. Exit
3

Thanks!!!
>>>
```

Fig 7: Command Prompt Demo

Using GUI

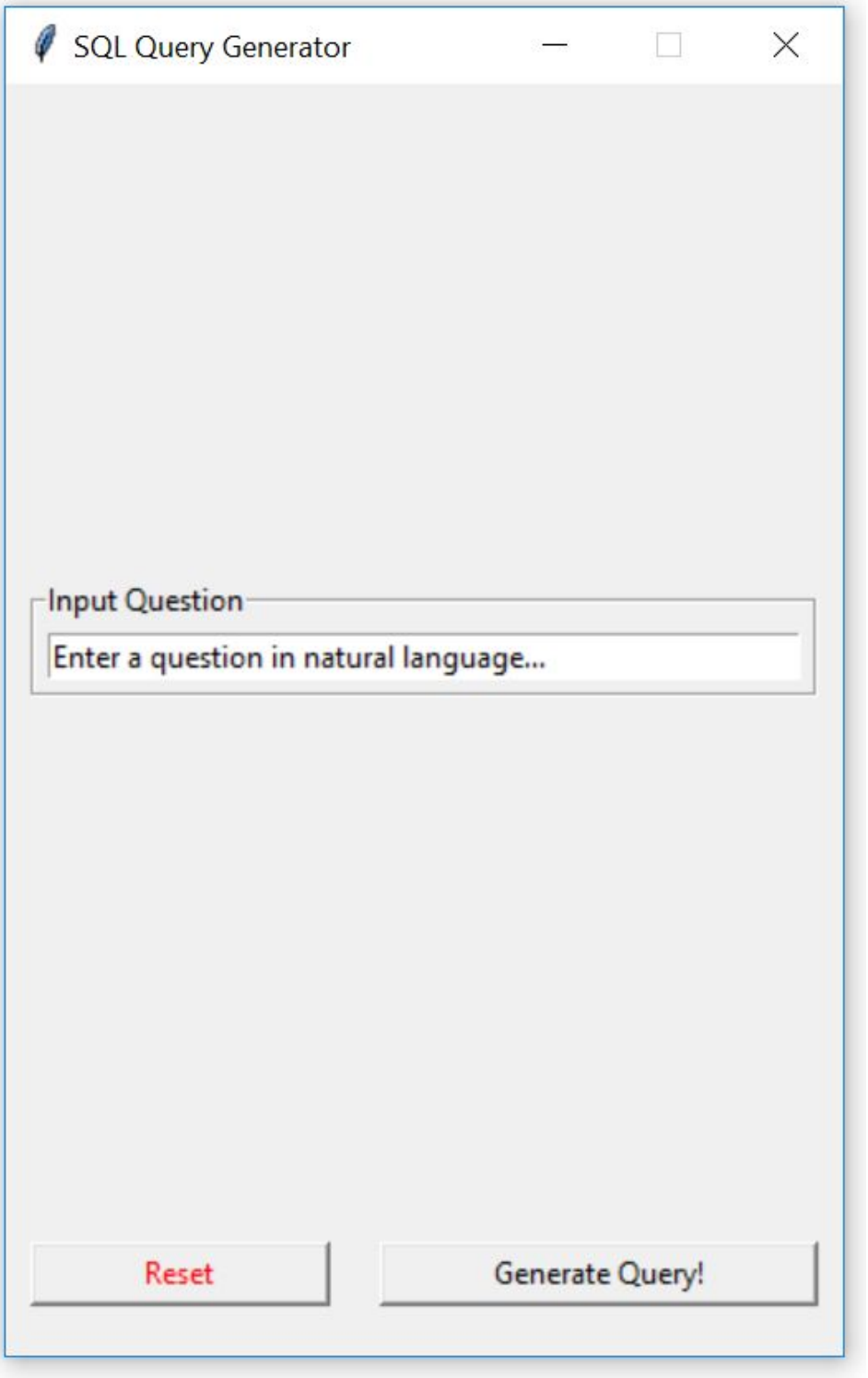

Fig 8: GUI Demo-1

Parsing Done (Query Formed)

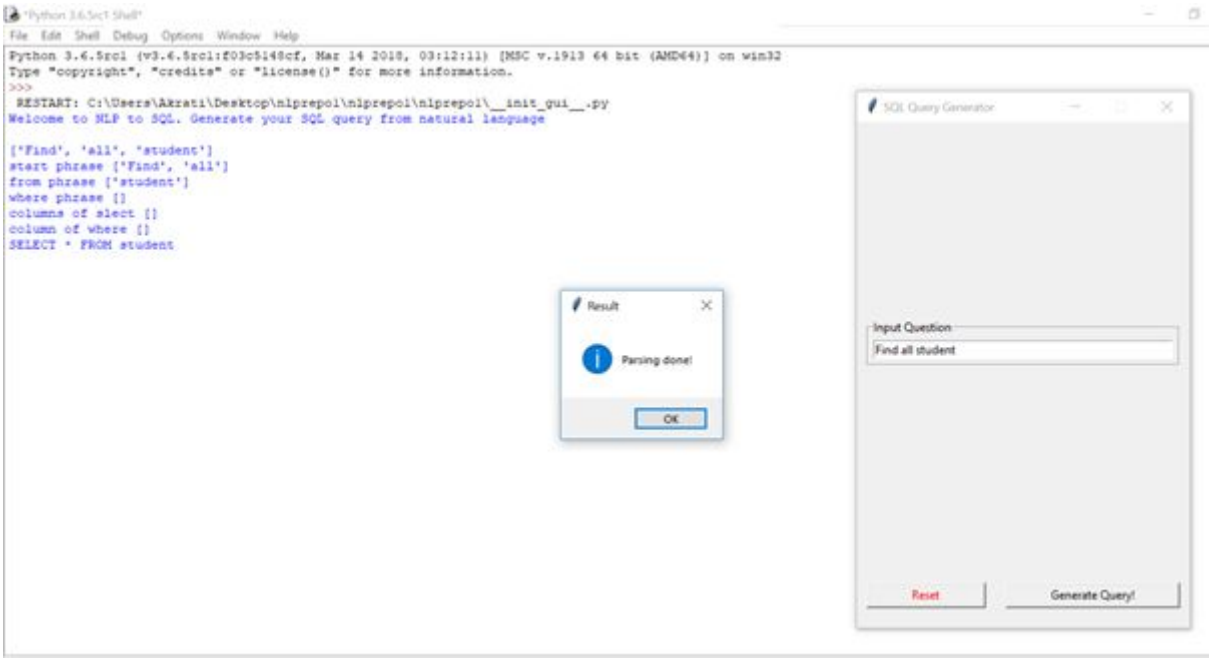

Fig 9: GUI Demo-2

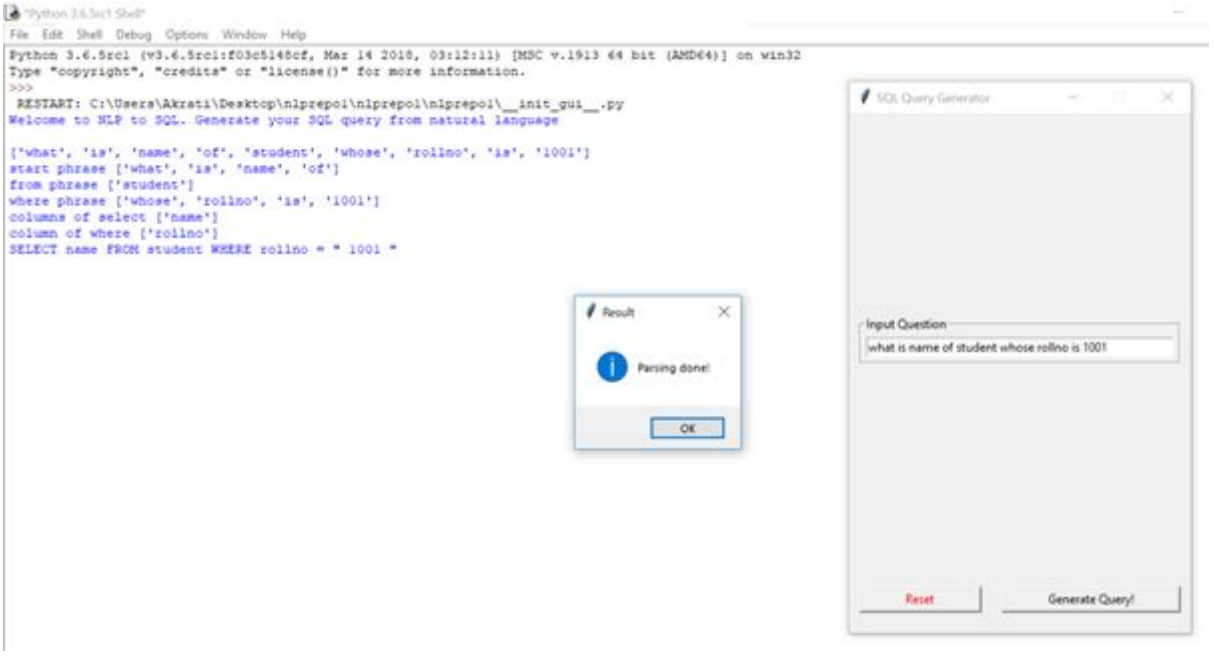

Fig 10: GUI Demo-3

4. TESTING

S. No.	Input text	Expected Output	Actual Output
1	Display all student.	SELECT * FROM student	SELECT * FROM student
2	Find detail of student whose rollno is 1001	SELECT * FROM student WHERE rollno = " 1001 "	SELECT * FROM student WHERE rollno = " 1001 "
3	Find phone of student with name Akrati	SELECT phone FROM student WHERE name = " Akrati "	SELECT phone FROM student WHERE name = " Akrati "
4	Get name, phone, section of student whose rollno 1005	SELECT name,phone,section FROM student WHERE rollno = " 1005 "	SELECT name,phone,section FROM student WHERE rollno = " 1005 "
5	What is name of all student.	SELECT name FROM student	SELECT name FROM student

Table 3: Testing-1

S. No.	Input text	Expected Output	Actual Output
1	Select student whose name is Manika and phone is 9876543210	SELECT * FROM student WHERE name = “ Manika “ and phone = " 9876543210 "	SELECT * FROM student WHERE name phone = " 9876543210 "
2	Which student has 1003 as rollno.	SELECT * FROM student WHERE rollno = ” 1003 "	SELECT * FROM student WHERE rollno = " rollno "

Table 4: Testing-2

5. Bibliography/References

- Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., Slocum, J. ―Developing a natural language interface to complex data‖, in ACM Transactions on database systems,1978,pp.105147.
- Huangi,Guiang Zangi, Phillip C-Y Sheu ―A Natural Language database Interface based on probabilistic context free grammar‖, IEEE International workshop on Semantic Computing and Systems 2008
- https://github.com/FerreroJeremy/ln2sql
- https://pypi.org/project/SpeechRecognition/
- https://stackoverflow.com/

6. Appendices

CODE SNIPPETS

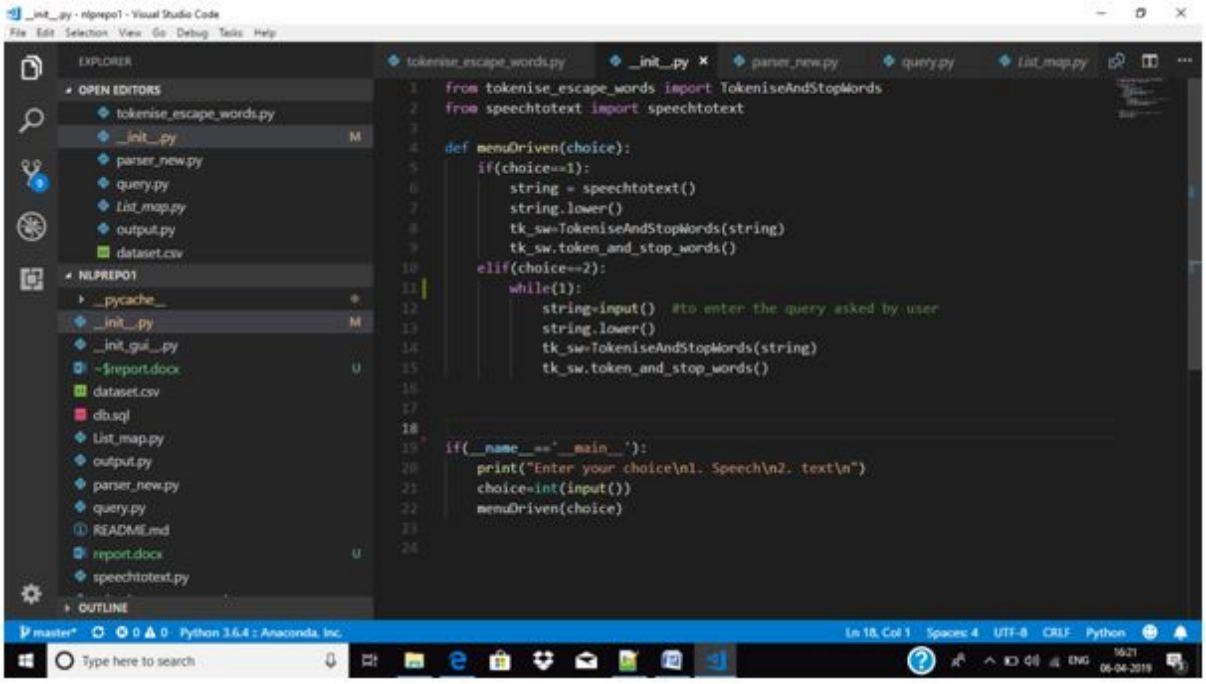

Fig 11: Code Snippet-1

This is the initial file "__init__.py" which starts with asking the choice to enter query by typing or by speaking. It takes the query as string then passes on to the other module or class where it gets its further computation.

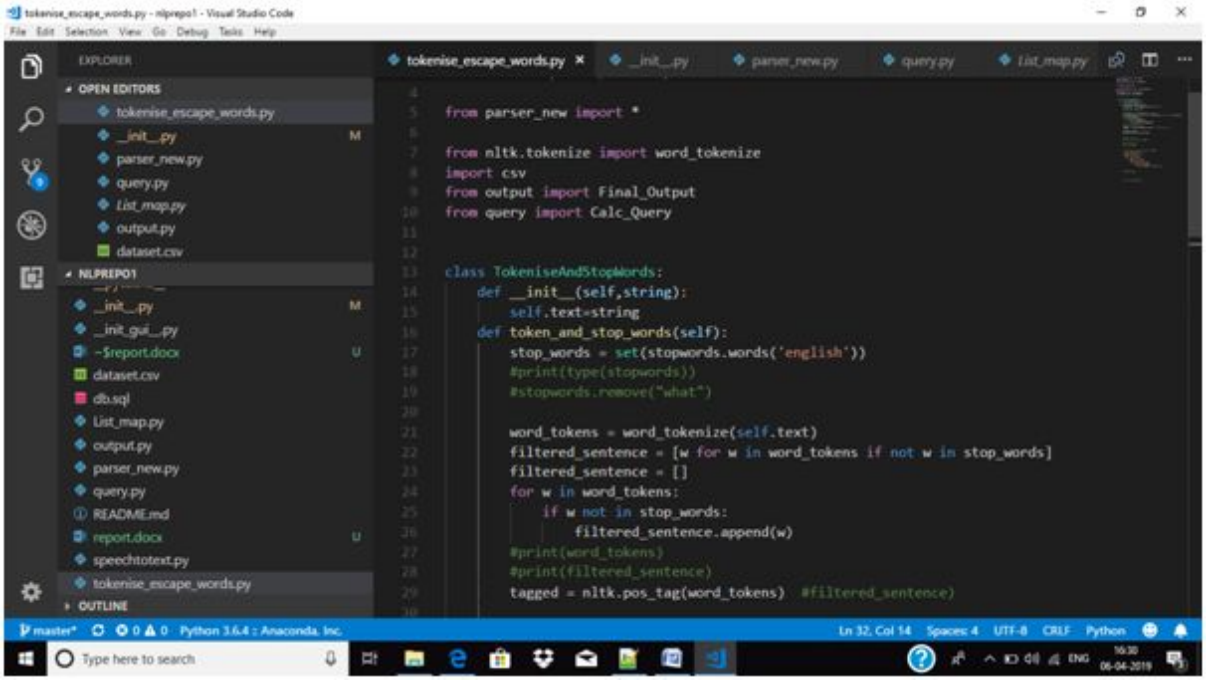

Fig 12: Code Snippet-2

In this module (class) the sentence gets tokenized and done part-of-speech tagging. After which the escape words are taken out from the tokenized list and the filtered sentence is then passes to parsing to convert it into desired query.

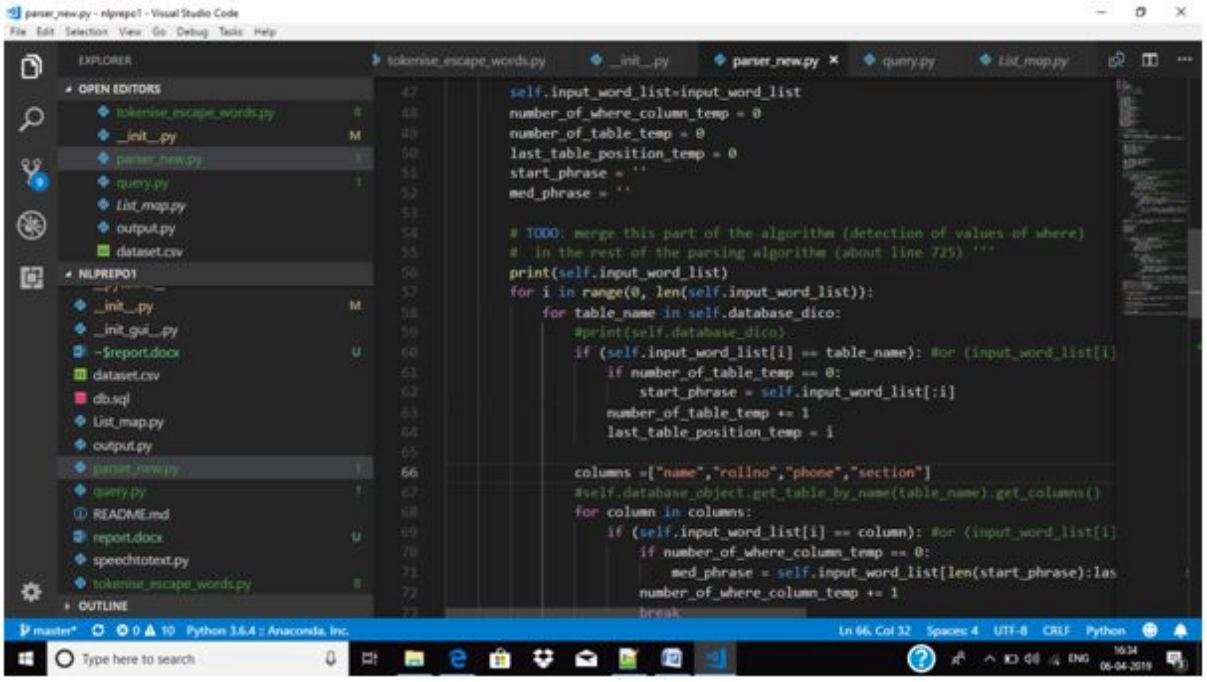

Fig 13: Code Snippet-3

In this code class , the tokenized sentence is broken into two parts the start phrase and the med phrase where the start phrase is the SELECT phrase and then concatenated the FROM phrase(which has the table name, from which the data to be retrieved) and then the med- phrase contains the WHERE phrase. This broken query is passed onto the output file.

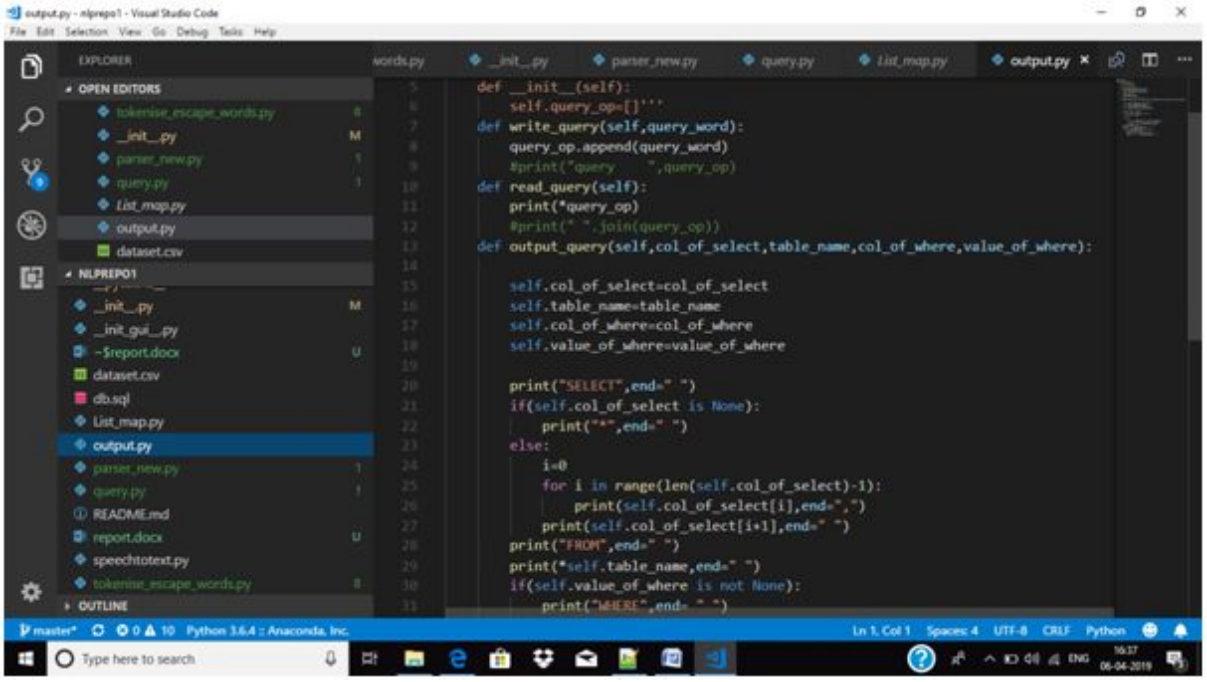

Fig 14: Code Snippet-4

This is the output file of the last module which generates the output for the entered sentence. This takes the broken query from parsing file and joins it to make the desired or the final query.