Natural Language Sentence to SQL Query Converter

Dhairya Chandarana
Department of Information Technology Vidyavardhini's
College of Engineering and Technology
Mumbai, India

Mohit Mathkar
Department of Information Technology Vidyavardhini's
College of Engineering and Technology
Mumbai, India

Prof. Anagha Patil
Department of Information Technology Vidyavardhini's
College of Engineering and Technology
Mumbai, India

Deepchand Dubey
Department of Information Technology Vidyavardhini's
College of Engineering and Technology
Mumbai, India

***Abstract*—Nowadays,the information in the IT sector is growing tremendously at a high speed. A large portion of the information is stored in the databases, mostly in relational databases. Such relational databases uses a special query language called Structured Query Language (SQL) in order to access or manipulate the data stored in them. Due to this, it becomes difficult for non-expert users to fetch data from relational databases, as they have no knowledge about SQL. Thus a need arises for a simple and efficient software model that can solve this problem of getting data easily from databases. This paper presents an approach to building such a system that uses natural language instead of SQL to retrieve data. We are building this system using Natural Language Processing and Deep Learning which is capable of understanding user questions and generating complex queries. This helps technical or non-technical people to easily access the database and get desired data without knowing much about SQL.**

***Index Terms*——SQL, Natural Language Processing, Syntactic, Semantic, Data Dictionary**

I. Introduction

Individuals around the globe, collect, stock, and use huge bulks of data daily, through their computers. These data may be found on computers or cloud storage and can access them remotely from another computer using the internet. A database or data warehouse is the repository storage system that is used to store such data. A Database Management System (DBMS) is used to manage and control the data stored in databases. Structured Query Language (SQL) is a special type of language that is used to access data from RDBMS. A lot of research has been carried out for using Natural Language (NL) in place of Structured Query Language to resolve the problem of SQL for non-professional users, which has initiated the building of a special framework called Natural Language Interface to Database (NLIDB). It is a part of a more considerable method called Natural Language Processing, or NLP in short. The main aim behind the Natural Language Processing based research is to develop an application based software model that is simple and user friendly.

II. Overview

The project aims to build a Natural Language Interface to Database (NLIDB) System. In this, when the user inputs a query in the form of natural language, the system will then first convert the query in a form that the Database Management System will accept, i.e. SQL. It then executes the generated SQL query and return the actual result to the user. Such a type of system can be used in various applications such as Intelligent ChatBot, Robotics, Business Intelligence, Departments which need data but don't have SQL knowledge.

III. Literature Survey

The endeavor in Natural language interface area commenced back in the fifties. At the start of the seventies, Prototype systems had appeared. The pattern matching was the method used for mapping the query from the user to the database by those systems. Formal list processor (FLIP), a language for pattern-matching formed on the LISP framework operates in a manner that if the given input fits in those patterns, in that case, the program will generate a query for the database. The idea to develop the interface instigated new issues to the designers because of the usage of databases that had spread in the 1970s.

Amongst many, the usage of natural language processing was one of the methods, wherein the client is allowed to examine and analyze the given information. Systems had been built over the years which focus only on a specific database to serve a specific purpose [10]. Some of those systems are explained below.

A. The LUNAR System

The questions regarding the samples of rocks obtained back from the moon were answered by The LUNAR system and it was introduced during 1971. Two Databases were used by the lunar framework to achieve its functionality. Among the two, one was for chemical analysis and the other one was for literature references. An ATN(Augmented Transition Network) parser and Woods Procedural Semantics was used by the LUNAR framework. The performance of the LUNAR System was quite amazing as it managed to control 78 percent of requests without reporting any significant problems. If dictionary mistakes were corrected, this proportion increased to 90 percent. But these statistics may be misleading since the system has not been used intensively because of its linguistic capabilities limitation.

B. Ladder

The LADDER program has been developed as a natural language interface for a U.S. Navy ship knowledge database. It uses the technique of semantic grammars, in which the semantic and syntactic processing is left inter-leaved. It parses the input and creates a parse tree, which is then mapped to the database query. [13]. A three-layered design is used to build the LADDER system. One of the features of this system is the Informal Natural Language Access to Navy Data (INLAND), which accepts natural language questions and allows databases to query it. The INLAND queries are then handled by the second component of the LADDER system which is Intelligent Data Access (IDA). For every syntactic unit within the input sentence, the INLAND module constructs fragments of an IDA query. These fragments are then merged to form more recognizable syntactic units. At the sentencing stage, the IDA receives these combined fragments as a command. The IDA writes a response to the user's query as well as prepare the correct sentence of file queries.

The third part of the LADDER system is the File Access Manager (FAM). FAM's function is to locate and control the access of generic files in the distributed database. The LADDER system was implemented in LISP. At the time of it's creation, the LADDER system was able to operate on a database containing 100 attributes and 14 tables.

C. CHAT-80

In the 80's, Chat-80 was the leading NLP system. This system was implemented with a famous programming language at that time known as Prolog. It was an efficient and advanced system. Its World Database contains various database rivers, oceans, countries, cities, borders and English vocabulary that is used for querying the database. This model translates the question asked in English by building a logical form with the help of three functions.

The program translates the English question by developing a logical form as processes of three serial and complementary functions where:

1) Logical constants are used to represent words.
2) adjectives, verbs and nouns are represented by Predicates.
3) Conjunctions of Predicates represent complex sentences or phrases.

Grammatical structure of the sentence is defined by Parser in this system. Prolog Clause helps directly in interpretation and scope rules of translation. The underlying implementation that Chat-80 follows is simple, it emphasises extra control knowledge to the logical form of query and forms a chuck of Prolog program that can be executed directly to get the answer. Control knowledge produced includes generally two forms:

1) The order that Prolog will follow to execute will be decided by command predictions of a query.
2) Divides the program into sub-problems so the amount of Backtracking required by Prolog is less.

IV. Related Work

In "Conversion of Natural Language Query to SQL Query" [1], they have explained basic terms to convert natural language statements to SQL queries. It gives brief information terms such as Tokenization, Lexical, Syntactic and Semantic Analysis. Following are the steps they followed:-

- Tokenization:- The entered query gets tokenized by separating the entire sentence into individual words. These words are then saved in a list and the list is passed for lexical analysis.
- Lexical Analysis:- The lexical analyzer will receive the list and replace the words in the list with the corresponding database word, from the dictionary and will pass it for syntactic analysis.
- Syntactic Analysis:- Here, a dictionary containing the attribute names, table names, and keywords is maintained and every tokenized word is mapped to its appropriate attribute name.
- Semantic Analysis:- In this step, it finds words that represent any conditions and will replace such words with there symbol. For Example: If in the sentence there is a word like "greater than", it will get mapped with the symbol $\geq$.

From the paper "Domain-Specific Query Generation from Natural Language" [2] in reference, they have given detailed information about the framework used for query generation. It includes Input queries, Pre-processing of query, Extracting Domain Specific Data, Semantic Disambiguation, using predefined SQL Syntax to generate SQL Query.

- Tokenization, Part of Speech (POS) tagging, stemming, and removal of stop words are the steps involved in pre-processing of query.
- Words or phrases can have more than one interpretation, can be identical or can be different. Semantic Disambiguation explains how we can overcome this problem.

- Domain-Specific Information Extraction involves categorizing words, and finding out whether they are nouns, verbs, adjectives, etc.
- In the next step, the tokens are mapped with SQL keywords such as INSERT, SELECT, DELETE, etc.

To compute the performance of their system, they used the Recall, Precision and F-measure values, based on which they got 82% correct results.

In the paper, “Nquery-A natural language statement to SQL query generator” [3], their system is totally presented considering MySQL Database. The approach which they have followed is the extraction of certain keywords from natural language sentences by forming tokens and tags goes through natural language processing. Analysis of these tags and tokens is done. Further, from generated information mapping to table names and columns is done. It then removes any redundancy if present and maps clauses in the query. In the next step, it forms a final query and executes it. The analysis report shows the accuracy of the system was found as 86%. Their system is capable of handling Simple queries as well as complex queries containing natural and inner joins. Handles Aggregate functions and various clauses like Order By, Group By.

Limitations of their system is that mapping fails in some queries like, who teaches Physics?. It will take Physics as a Department as it fails in this case to identify that who refers to an instructor.

In the paper [4], authors have given that implementation was carried out using RNN (an autoencoder and decoder converts human language voice to sentence) and LSTM(encoder and decoder coverts sentence to SQL query). So using both in one setup will solve the problem of Semantic Parsing. This paper challenges several ways to ask a question and match all the different types of questions to the desired output. The accuracy of their system was recorded as 60%. They used WikiSQL as their dataset. WikiSql [5] is a set of data(dataset) having 87726 Question samples and their SQL queries and these are all made using 26375 tables from Wikipedia.

In ”SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task” [6], the decoder is structured by the model as a group of recursive modules. The usage of SQL specific grammar helps them to guide the process of decoding, which makes them allow to grab the benefit of well-defined structure of SQL queries’. Modules likewise do not share any parameters, so that they can be trained independently. In their model, they have used Input Encoder, Table-Aware Column Representation, SQL Decoding History, and Attention for Input Encoding. In the project, they have also used cross-domain data augmentation method to expand our training data for complex queries. The difficulty of Cross-domain data augmentation is quite greater in comparison to the in-domain setting. The reason behind this is that the domain-specific words and phrases are tending to be included by question program pairs.

”SQLNet: Generating Structured Queries from Natural Language Without Reinforcement Learning” [7], shows that they have Put forward an idea, Where SQLNet can be used to control and manage the NL2SQL task. The approaches in a most existing system employ a model named sequence-to-sequence, which would suffer from the problem named“order-matters” in which the order actually did not matter. Their model was able to overcome this problem in a better way as compared to the previous attempts that used the technique named reinforcement learning to resolve this problem by only a slight enhancement, for an instance by about 2 points. Now, SQLNet overcomes this problem of “order-matters” by adapting and employing a model named sequence-to-set to retrieve SQL queries in which the order did not matter. To further boost the performance of the sequence-to-set model, they introduced the column attention technique. Overall, we notice that their SQLNet system would improvise over the previous work that is Seq2SQL, by a great marginal amount that ranges from 9 points to 13 points on different metrics. This shows us that their mechanism can favorably resolve the problem of “order-matters”, and illuminate some novel solutions to the problem of structure generation when order doesn’t matter.

V. Proposed System

A. How to develop such a system?

For developing a natural language interface of this type, the system is responsible for decoding users’ queries and convert them into appropriate SQL queries automatically. So the question arises on how to build such systems? To train neural networks on a larger scale of data having questions and SQL pair labels We have applied deep learning methodology. These methods are more robust and scalable compared to rule-based and well-designed systems.

B. Building Interface

The aim to develop this interface is to master the complex text-to-SQL job involving SQL clauses, nested queries, and multiple tables. Moreover, for testing and training, we have used separate databases that aim to build models that deduce to new databases.

1) Dataset: To proceed with this project, the first thing we will need is a huge dataset of English questions and their corresponding SQL query. However, there is a problem where to find these many questions and SQL pair labels to train our model? Building this kind of dataset takes a huge amount of time and effort. The reason for that is the developer has to recognize the schema of the database, frame the questions, and then finally write the appropriate SQL queries of it. This all requires precise database knowledge and must have expertise in SQL.

Another kind of difficulty in creating such a dataset is that there are limited numbers of databases with multiple tables that are non-private. So To resolve the necessity for a high-quality, huge dataset for this work, SPIDER [8] dataset is used. It includes two hundred databases containing multiple tables, Ten thousand one eighty-one questions, and five thousand six ninety-three matching complex SQL queries. This dataset is composed of eleven Yale students that spent a total of thousand man-hours!

All questions and SQL queries in the SPIDER dataset are composed and assessed by eleven students of computer science. The only Text-to-SQL dataset containing both databases involving several tables in several sectors and complex queries of Structured Query Language is the Spider dataset. It also can adapt to new domains along with the capability to deduce new SQL queries and database schemas. Compared to other datasets, Spider includes databases with several tables and includes SQL queries that involve many complex SQL classes. Another comparison concerning the total of previous text to SQL datasets is that the Spider has about two times more nested queries and ten times more ORDER BY and GROUP BY (HAVING) components. Spider includes two hundred different databases that cover one thirty-eight distinct sectors such as club, TV show, government, college, etc.

2) Power of Softmax function: "Softmax function or softmax is a type of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1" [12]. Its output values are in range from 0 to 1 which is good since our neural network model can get rid of binary classification and accommodate various dimensions and classes. This is why softmax function is referred as "multinomial logistic regression", a prediction based analyser, it is similar to multiple linear regression. While training a dataset this function is used to determine the losses. Noise Contrastive Estimation and Cross-Entropy are the best use cases of this function as it helps to optimize the training data and hence increase the probability of predicting the correct sentence or a word. Calculation of conditional embedding $H_{1/2}$ of an embedding H_1 such that embedding H_2 is given.

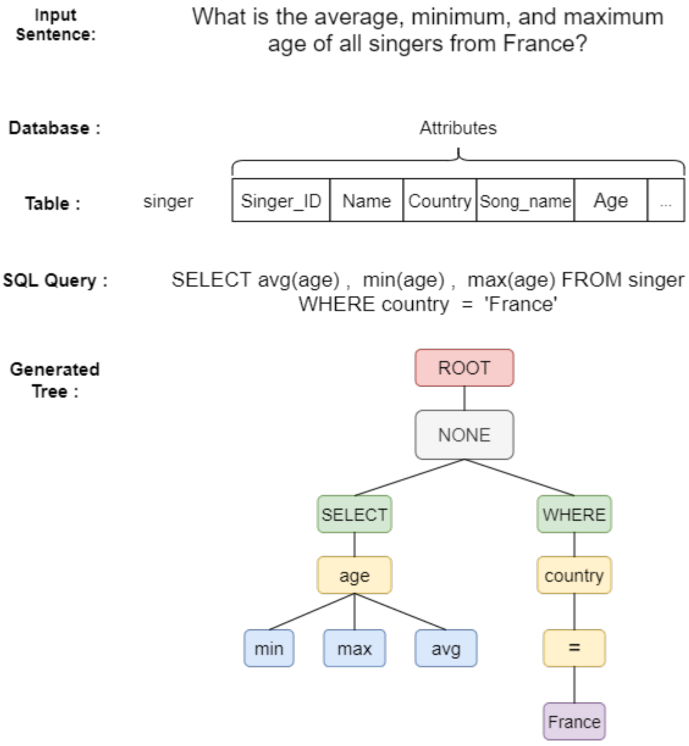
$$H_{1/2} = softmax(H_1 W H_2) H_1 \tag{1}$$

$$P(A) = softmax(X tanh(A)) \tag{2}$$

Here W, X is trainable parameter. Probability distribution is calculated based on score matrix A.

3) Models(Modules):

1) Aggregate Predictor: It handles the prediction of Aggregate function like MIN, MAX, COUNT, AVG, SUM and NONE.
2) Keyword Predictor: It handles prediction of keywords like FROM, GROUP BY, ORDER BY and WHERE.
3) Having Predictor: It handles prediction of HAVING and make sure that it used only with the clause GROUP BY. It gets executed only after there is prediction of GROUP BY clause.
4) Col Predictor: It handles predicting table columns.
5) Operator Predictor: It handles operators like $>$, $<$, =, $\geq$, $\leq$, $!=$, NOT IN, IN, LIKE, BETWEEN.
6) Set Operation Predictor: It handles predicting INTERSECT, EXCEPT, UNION, and NONE, which sometimes helps if we need to run generate nested queries.
7) Root terminal Predictor: The terminal value and new-sub query Root is handled by this module. With the help of this nested query is generated.
8) AND-OR Predictor: This module takes care of prediction of AND, OR if some condition is present in the query.
9) Desc-Asc-Limit Predictor: It handles predicting the keywords associated with ORDER BY clause and it play its role when some arrangement is needed in particular order. If ORDER BY is predicted then only it gets executed.
10) Insert, Delete, Update Predictor: This module handles the prediction of INSERT, DELETE and UPDATE DML commands.



Input Sentence: What is the average, minimum, and maximum age of all singers from France?

SQL Query : SELECT avg(age) , min(age) , max(age) FROM singer WHERE country = 'France'

Fig. 1. Illustration of modules.

Although our models are almost like SyntaxSQLNet [6] and TypeSQL [11], their models are only ready to predict SELECT commands, whereas our models aim at improving the overall efficiency and can also handle various Data Manipulation Language (DML) commands other than SELECT commands, such as INSERT, DELETE and UPDATE queries. Their system only generates the SQL query, which might be very complicated and difficult for non-technical users to understand. Our system executes the query generated by the model on the corresponding database using the sqlite3 module of python, fetches the desired result, and displays it on a comprehensible Graphical User Interface(GUI).

4) Algorithm:

1) Step 1. User give their input in Natural Language.
2) Step 2. Perform data pre-processing tasks such as morphological, lexical, syntactic and semantic analysis.
3) Step 3. Next, perform word embedding on sentence using GLoVe-an unsupervised learning algorithm.
4) Step 4. Call different predictors which the system uses, such as, operator, keyword, etc.
5) Step 5. Recursively call those modules until a proper query is generated.
6) Step 6. Executed the generated query on the corresponding database, fetch the result, and display it on GUI. However, if the model is not able to generate a proper query, it displays the appropriate error message to the user.

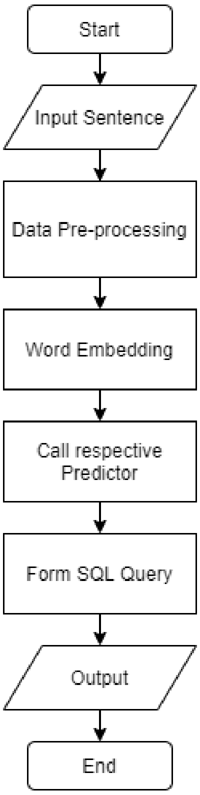


Fig. 2. Flowchart of the system.

VI. Results and Discussion

Our project demonstrates a practical solution to query a database using natural language, for the users that do not have any knowledge about SQL. Our model has high efficiency in generating SQL queries and achieves the exact match accuracy of 60%.

Drawbacks of systems that already exists.

- They cannot handle complex queries very well.
- They can execute only a few type of SQL commands.
- Many existing systems are build for a specific purpose due to which they are restricted only to a certain database.

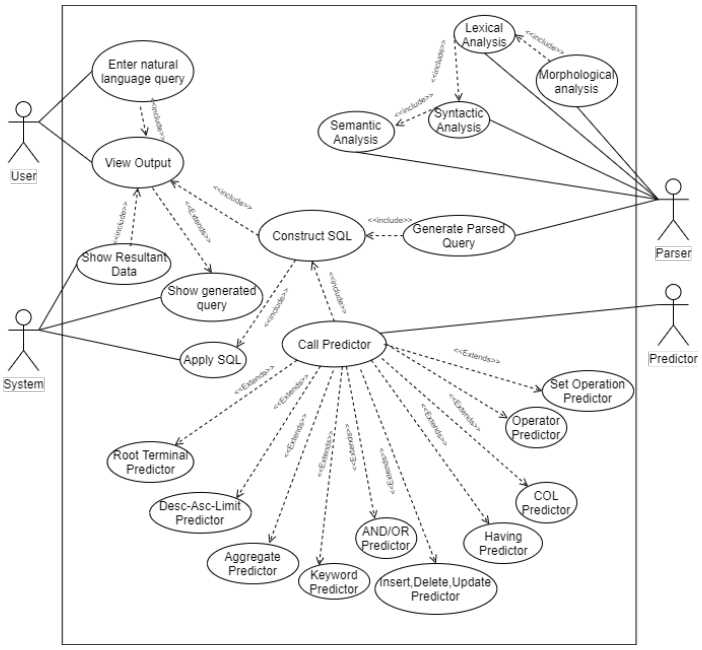
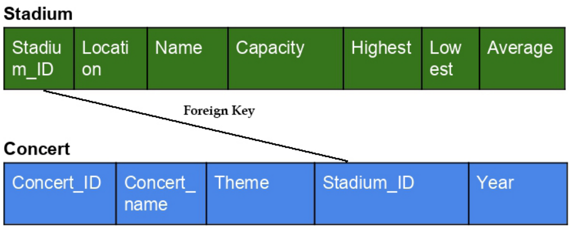


Fig. 3. Use Case Diagram of the system.

Our System - This project is currently capable of handling:-

- Complex queries involving joins.
- Aggregate functions in queries.
- ORDER BY, GROUP BY, HAVING clauses within SELECT command.
- Incorporates a wide range of queries on multiple tables.
- Can handle commands other than SELECT as well, such as UPDATE and DELETE commands.
- Can execute the generated query on the database and return the actual result.



Stadium

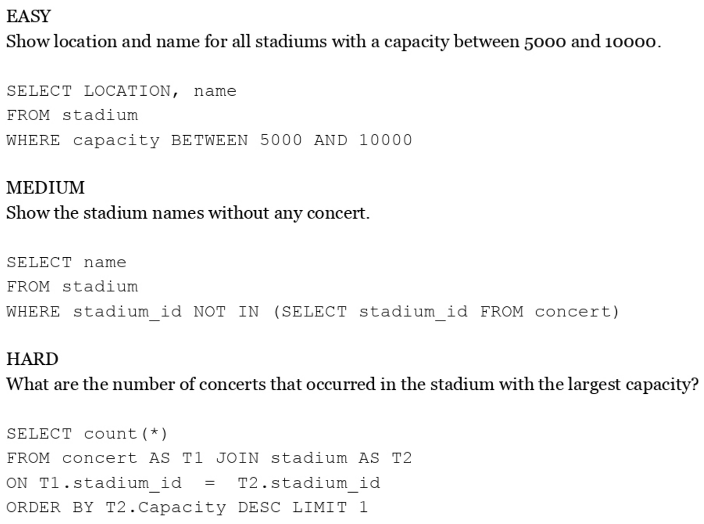
Stadiu m_ID	Locati on	Name	Capacity	Highest	Low est	Average

Concert

Concert_ID	Concert_ name	Theme	Stadium_ID	Year

Fig. 4. Schema of a database named ”concert-singer” from dataset.

The above figure shows the schema of a sample database ”concert-singer” from SPIDER dataset. The Fig. 5 shows the different level of hardness of queries generated by our model. Those queries are executed on the sample database ”concert-singer”. All 206 databases in our dataset are split into 146 train and 60 test as training data and test data respectively. All questions for the same database are in the same split. Still many research work is going on in this field of NLP to develop a robust interface for databases that are able to generate accurate results for a sentence to DB query.



EASY
Show location and name for all stadiums with a capacity between 5000 and 10000.

```
SELECT LOCATION, name
FROM stadium
WHERE capacity BETWEEN 5000 AND 10000
```

MEDIUM
Show the stadium names without any concert.

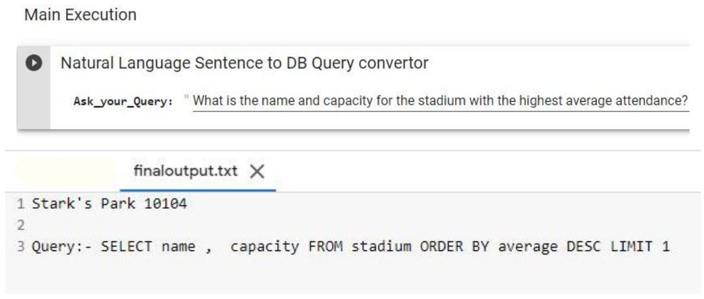
```
SELECT name
FROM stadium
WHERE stadium_id NOT IN (SELECT stadium_id FROM concert)
```

HARD
What are the number of concerts that occurred in the stadium with the largest capacity?

```
SELECT count(*)
FROM concert AS T1 JOIN stadium AS T2
ON T1.stadium_id  =  T2.stadium_id
ORDER BY T2.Capacity DESC LIMIT 1
```

Fig. 5. Predictions made by the system.

With advancement in computational power many natural language Interface for Database able to generate better results. But still it is not commonly used, as more powerful and efficient algorithms need to be developed for optimised query and make it available for general use [9].



" What is the name and capacity for the stadium with the highest average attendance?

```
Stark's Park 10104

Query:- SELECT name ,  capacity FROM stadium ORDER BY average DESC LIMIT 1
```

Fig. 6. Execution and output of query.

VII. Conclusion

In this paper, we proposed a deep neural network for converting natural language to SQL queries. We provided a user interface where users can ask a question by typing their questions directly, and they will get the desired result. Also, our model trained with large datasets so as to handle complex queries (like Clauses, Joins). Thus, now it becomes easy to get data from the database, and even non-technical users can extract the required information from the databases and get the answer to their queries.

VIII. Future Work

1) Our system can only execute queries on MySQL Database, and it can be extended to support more databases like NoSQL, unstructured database.
2) For now, this system support only English languages, other language can be incorporated to make more user-friendly.
3) The system can be modified to support voice commands which will enable users to ask their queries in the form of speech and the software will convert it into text (speech-to-text).
4) We can modify the system by adding a feedback module wherein the users will be asked to provide correct results if the generated result was incorrect and if so, it will be added to the training set. Thus the neural network can be trained through feedback and will be able to achieve higher accuracy.
5) More effective algorithm can be built to manage complex queries.
6) We also intend to do real-time testing with all possible inputs to make our system more accurate and more feasible with real-time users.

References

[1] Abhilasha Kate, Satish Kamble, Aishwarya Bodkhe, Mrunal Joshi, “Conversion of Natural Language Query to SQL Query”, 2nd International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE 2018.

[2] Anum Iftikhar, Erum Iftikhar, Muhammad Khalid Mehmood, ”Domain Specific Query Generation from Natural Language”, Sixth International Conference on Innovative Computing Technology, IEEE 2016.

[3] Nandan Suthankar,Sanket Maharnawar,Pranay Deshmukh ,Yashodhara-Haribhakta ,Vibhavari kamble, “Nquery-A natural language statement to sql query generator” , “55th Annual Meeting of Association for computational Linguistics.

[4] Zhong, Victor, Caiming Xiong, and Richard Socher, ”Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning”, arXiv preprint arXiv:1709.00103, (2017).

[5] Hwang, Wonseok Yim, Jinyeung Park,Seunghyun Seo, Minjoon. (2019). A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization.

[6] .Tao Yu and Michihiro Yasunaga and Kai Yang and Rui Zhang and Dongxu Wang and Zifan Li and Dragomir Radev, ”SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task”, 2018.

[7] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. ”Sqlnet: Generating structured queries from natural language without reinforcement learning”. arXiv preprint arXiv:1711.04436.

[8] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. ”Spider: A largescale human-labeled dataset for complex and crossdomain semantic parsing and text-to-sql task”. In EMNLP.

[9] Jiaqi Guo and Zecheng Zhan and Yan Gao and Yan Xiao and Jian-Guang Lou and Ting Liu and Dongmei Zhang, ”Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation”, 2019.

[10] E. U. Reshma and P. C. Remya, ”A review of different approaches in natural language interfaces to databases,” 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, 2017, pp. 801-804.

[11] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. ”Typesql: Knowledgebased type-aware neural text-to-sql generation”. In Proceedings of NAACL. Association for Computational Linguistics.

[12] towardsdatascience, (2018), Softmax-function-simplified-714068bf8156 — towardsdatascience, [online] Available at: https://towardsdatascience.com/softmax-function-simplified-714068bf8156 [Accessed on 28 Jan, 2020]

[13] El-Mouadib, Faraj Zubi, Zakaria Almagrous, Ahmed El-Feghi, I.. (2009). Generic interactive natural language interface to databases (GINLIDB). 57-76.