Available online @ https://jjem.jnnce.ac.in
https: www.doi.org/10.37314/JJEM.SP0273
Indexed in International Scientific Indiexing (ISI)
Impact factor: 1.395 for 2021-22
Published on: 08 December 2024

VOICE BASED SQL QUERY GENERATOR USING NLP

ABHILASH S[1], SUNITHA G P[2],
Department of Master of Computer Applications
JNN College of Engineering, Shivamogga
abhibalaji32507@gmail.com[1], sunithagpise@gmail.com[2]

ABSTRACT

A voice command-driven SQL query generator is tailored for banking systems, aiming to enhance user engagement and operational efficiency. Traditional SQL query creation poses challenges for non-technical staff due to its technical requirements. Our solution simplifies this process by translating voice commands into SQL queries, enabling users to access and manage data effortlessly. Advanced speech recognition and natural language processing technologies decode user intentions accurately. The NLP engine interprets context and semantics, while the voice recognition module transcribes user inputs. Leveraging Vanna AI, the query generator dynamically generates precise SQL queries to interact with the SQLite database. The prototype exhibits impressive accuracy in handling complex queries, presenting significant benefits for financial operations.

Keywords:SQL,Query

1. INTRODUCTION

The modern lives are being drastically changed by artificial intelligence (AI), particularly with the help of developments like deep learning, which is driving advances in computer vision, natural language processing(NLP), and gaming. SQL query generators can be Formulating using natural language processing (NLP), which is essential for text interpretation.AI and NLP integration in banking has produced powerful voice-based SQL query tools that make it simple for users to Convert natural language questions into structured SQL. On allowing users to retrieve client data and identify fraud using spoken instructions, Thesesystems maximize data access. This intuitive, hands-free technologyimproves productivity accessibility while enabling users to make well-informed decisions. All things considered, voice-based SQL query generators make more comprehensiblebanking procedures for fast response, Facilitating smooth database interactions and making database interactions understandable even for non-SQL users. The introduced system constructs the SQL final query, runs it in accordance with the final query type (SELECT, UPDATE, DELETE), trains the

model for accuracy, and forecasts querytypes.

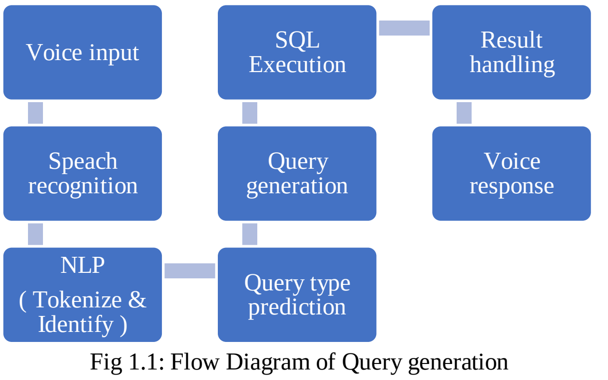


Fig 1.1: Flow Diagram of Query generation

The flow diagram represents a system where voice input undergoes speech recognition and NLP tokenization. The system projects the query type, generates SQL, executes it, handles the results, and provides a voice response back to the user.

2. PRIOR ART

This section extensively studies existing works on text-based query generators, describing several notable efforts. It compares these existing solutions with the proposed approaches, highlighting key differences and improvements. The analysis aims to show how the proposed methods enhance and build upon current technologies, offering more efficient and effective solutions for text-based query generation.

Niranjan Gowda, et al Explored AI's rapid evolution in NLP and deep learning, with practical applications like image recognition and chatbots, and highlighted privacy, security, and bias challenges [2]. Ftoon Kedwan (2022) described an NLIDB system for translating NLQs into SQL, emphasizing lexical, syntactic, and semantic analysis, with processes including tokenization, lemmatization, stemming, and syntactic role tagging using Python's "speech recognition" library, and WordNet for semantic analysis [1]. Aditya Sawant (2022)**:** Presented an AI model using natural language system can formulate sql queries voice instructions, aiding users without SQL expertise and those with RSI, using NLP and an LSTM neural network, achieving 93.47% accuracy [3]. Yuanfeng Song and Raymond Chi-Wing Wong (2022) Presented a Voice Query System, a voice-driven database querying tool using NLQs, translating NLQs into SQL via cascaded and E2E approaches, and enhancing non-technical user accessibility [4]. Anisha T. S., Rafeeque P. C and Reena Murali (2019) Introduced an NLI system that translates NLQs into SQL using a deep neural network, leveraging the Spider dataset, and employing Seq2Seq models with Glove word embeddings [5]. Pooja Nivrutti Chaudhri, Rutuja Vijay Kadam, and Namrata Vishnu Kamble (2019)**:**ntroduced a voice-controlled personal assistant using NLP and machine learning to translate English queries into SQL, aiding non-technical users [6]. Amey Baviskar, Akshay Borse, Eric White, and Umang Shah (2017)**:** Detailed an NLP system translating English queries to SQL using semantic grammar, employing NLTK, Stanford POS tagger, and MySQL [7]. Pooja Nivrutti Chaudhari, Rutuja Vijay Kadam, and Namrata Vishnu Kamble (2020) Presented an NLP-based system for translating English queries into SQL using semantic grammar, enabling non-technical users to interact with databases using NLP [8]. Prasun Kanti Ghosh, Saparja Dey, and Subhabrata Sengupta: Outlined an NLP system using the "Levels of Language" model, facilitating automatic SQL query formation from natural language input [9]. Tanzim Mahmud, K. M. Azharul Hasan, Mahtab Ahmed, and Hla Ching Chak**:** Discussed a rule-based approach for NLP-based query processing in databases, simplifying computer interaction for general users [10].

Table 1: Comparison of Proposed and Existing Works

Particular	Voice-Based	Text-Based
Input method	Uses speech recognition	Uses typed text input
Ease of Use	Users can speak naturallywithout knowing SQL syntax.	Difficult to type long inputs
speed	Faster to generate queries by speaking.	Typing out queries can be slow, especially for complex.
Natural Language	Provides immediate feedback to correct errors via conversation.	syntax errors &misinterpretations, requiring manual corrections.

- The scope of the proposed work is limited only for simple SQL queries
- The queries as well as execution of complex queries will be extended

3. PROPOSED METHODOLOGY

The difficulty of allowing users to query databases using plain language without having prior knowledge of SQL is addressed by our suggested technique. The system works in many steps. First, it converts user input into text format by using speech recognition. after then, tokenizationwhich divides sentences into individual words kept in a listbegins text processing. In order to focus the question, stop words like "I" and "you" are subsequently eliminated. To improve syntactic and semantic analysis for precise query interpretation, parts of speech tagging is used. Utilizing assessed information, The system then verifies and creates the SQL query that needs to be run. An interface is

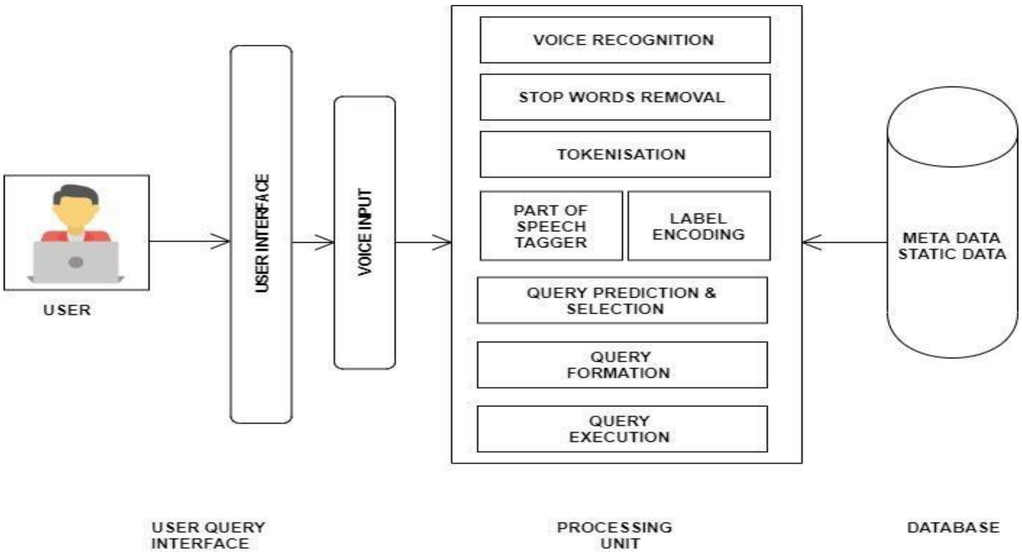
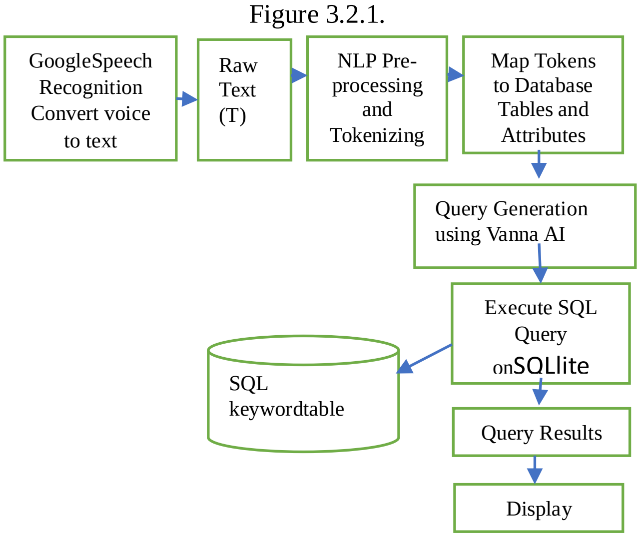


Fig 3.1 System Architecture

used to visualize the data that has been retrieved from the database to the customer.enabling effective data retrieval according to user needs. By making natural language interaction simpler, this method seeks to democratize database access and close the gap for non-technical users.voice SQL query generators are easier, faster, hands-free, understand natural language better, and reduce errors more effectively than text-based query generators

3.2 SYSTEM FLOW

The system generates SQLite SQL commands from user-typed sentences, as illustrated in Figure 3.2.1.



1. Voice Input:
 - The user speaks a command into the system.
2. Convert Speech to Text:
 - Google Speech Recognition converts the spoken words into text.
3. Text Processing:
 - Filter Relevant Information: Keeps only the important parts of the text.
 - Tokenize: Breaks the text into smaller pieces
 - Remove Unnecessary Words: Removes common words and irrelevant information.
4. Map Tokens to Database:
 - Matches the cleaned tokens to the relevant parts of the database (tables and attributes).
5. Generate SQL Query:
 - Vanna AI creates an SQL query based on the identified components and conditions.
6. Execute Query:
 - Runs the SQL query on an SQLite database hosted on XAMPP.

7. Retrieve Results:
   - Gets the results from the executed query.
8. Display Results:
   - Shows the query results to the user.

3.3 ALGORITHM

The following are the steps involved in converting voice into an SQL Query

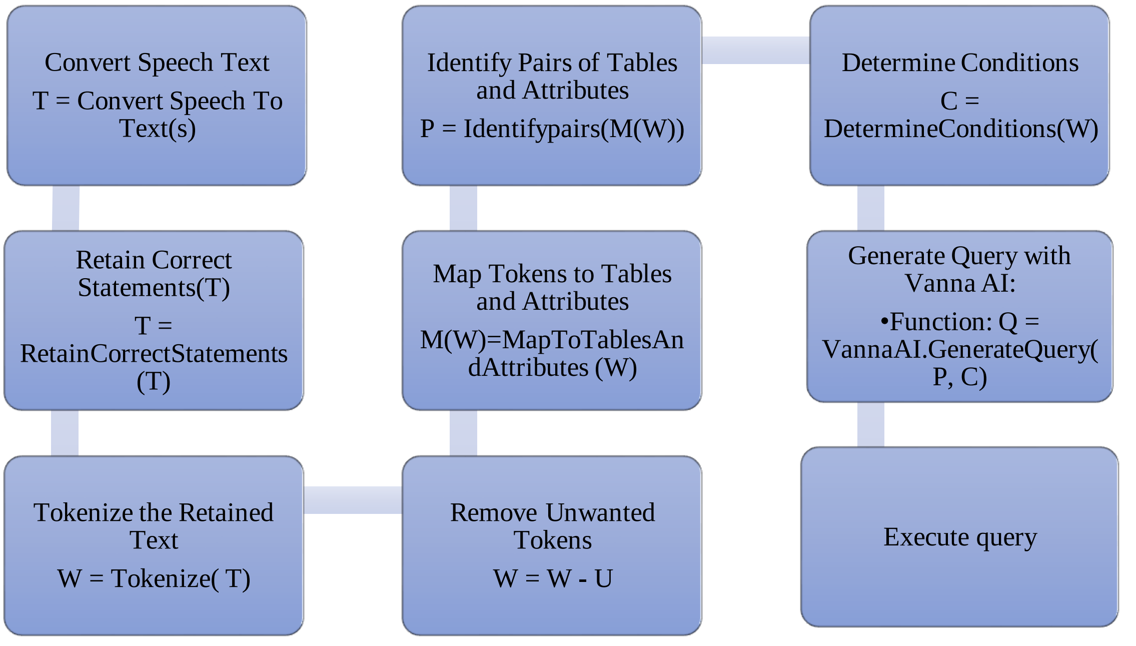


Fig 3.1.1 Algorithm of Voice Based Query Generation

The system starts by converting spoken language into text format, ensuring the accuracy of this transcription through verification processes. Once verified, the text undergoes tokenization and normalization to standardize its structure. Unwanted tokens are removed to streamline the text. The cleaned tokens are then mapped to the relevant database schema, which involves identifying corresponding table-attribute pairs. With these mappings, the system establishes specific query conditions based on the content. It proceeds by generating an SQL query that aligns with these conditions. Finally, the system validates the generated SQL query to ensure it is syntactically correct and functionally accurate.

Pre-processing Text

The initial step for preparing Indonesian command phrases involves pre-processing text. This includes tasks such as case folding, filtering, word tokenizing, stemming, and Stopword estimation, Presented in Figure3.1.2.

- The system pre processes the text after the user enters the command in Indonesian.
- It determines if the keywords match SQL commands and transcribes vocal commands into text.
- Tokenization divides the command into words, while filtering compares tokens against a table of phrases to detect special symbols like "=", which represents "equal," and "<" for "less than.
- The system displays for <>
- The system displays for greater than
- It is used for aggregate functions using group by
- It is used for where clause
- It is used for having clause
- It determines nested queries
- It determines DDL commands
- It identifies DML commands
- The system then displays the SQL command syntax by virtue of the input command in Indonesian.
- The system identifies DCL commands
- The system uses all set operations

- Tokenization divides the command into words, while filtering compares tokens

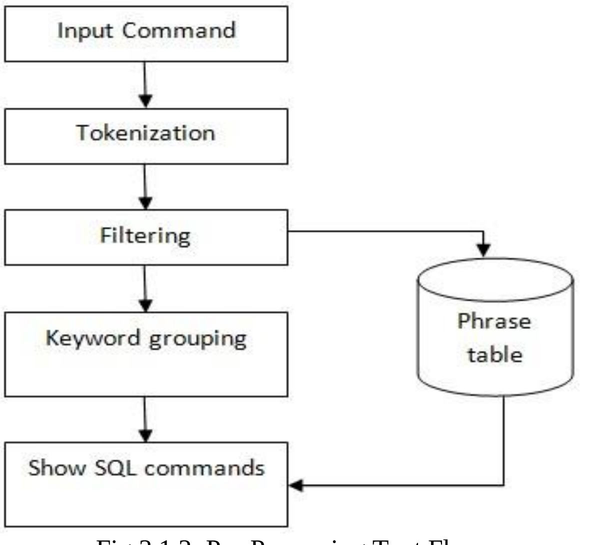


Fig 3.1.2: Pre-Processing Text Flow

4. RESULT AND ANALYSIS

Processing phases include tokenization, keyword detection, grouping, and testing are Portion ofthe query generating algorithm. This section discusses the products of various procedures.

4.1The Procedure for Tokenization Tokenization involves dividing a sentence into individual words, simplifying analysis. For example, "Convert speech to text" would be tokenized into ["Convert", "speech", "to", "text"]. This process is crucial for handling natural language processing tasks as it breaks down text into manageable parts.

Table1 showcases the outcomes of tokenization, illustrating how sentences are divided for further analysis.

Table 1: Example of the Tokenizing Process

Command Sentence	Word Index Order	Word	Category
Show account holders with balance above 10000	0	Show	Command
	1	Account holder	Other
	2	with	Other
	3	balance	Condition
	4	Above	Condition
	5	10000	Value

4.2 Keyword Grouping Process

In this process, tokenized words are categorized into SQL command groups and translated into SQL syntax. The grouping includes four steps:keyword group analysis, table and column analysis, SQL command analysis, and SQL syntax mapping.

1. Keyword Group Analysis

Analyzing root words from the tokenizing process involves categorizing them into "command" (e.g., SELECT), "value" (e.g., 10000), "condition" (e.g., WHERE),or "other" (e.g., table names like employees). This supportsin structuring SQL queries by clearly definingactions, criteria, and data to be retrieved

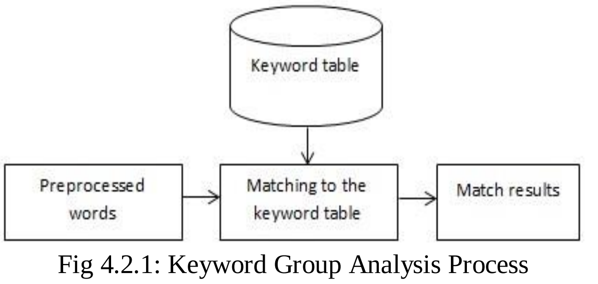


Fig 4.2.1: Keyword Group Analysis Process

Table 2: Keyword Analysis Results

Basic Word	Keyword Group
Show	All
Accounts	Entity
With	Command
Balance	Attribute
Over	Condition
1000	Value
Belonging	Command
To	Preposition
John Doe	Name

2. Table and Column Analysis

Identifies words that are not keywords to determine their type for further processing.

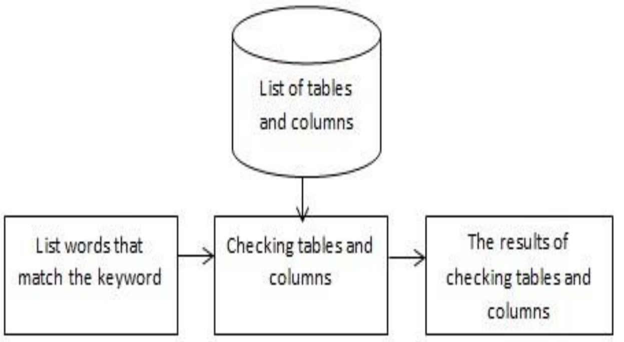


Fig 4.2.2: Table and Column Analysis Process

3. Identifying SQL Commands

- Determines whether the command belongs to the DDL or DML group.
- Identifies each word in the command as a table, field, command, condition, or value.
- Figure 7: Identification Process Flow

Table 3: Example of the Identification Process

Basic Word	Keyword Group	SQL Command
Show	Command	SELECT
All	Command	SELECT
Accounts	Table	Accounts
With	Condition	WHERE
Balance	Column	Balance
Over	Condition	>
1000	Value	1000
Belonging	Condition	AND

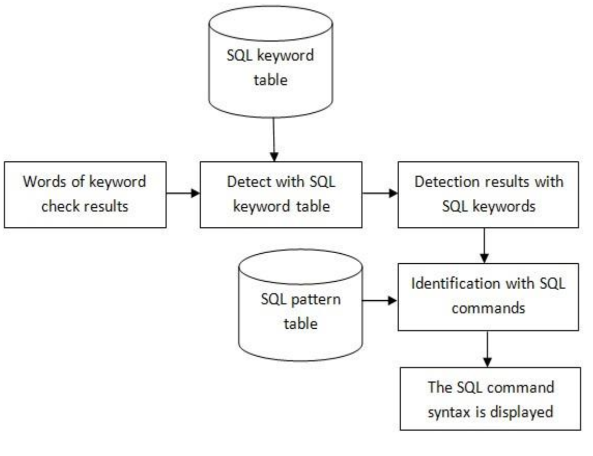


Figure 4.2.3. Identification process flow

4. SQL Command Syntax Mapping

Basic word	SQL command identify	Keyword group
Show	Select	Command
All	*	Table
Employee	employees	Table
Have	Where	Condition
Salary	salary	Condition
Above	>	Value
10000	10000	Value

- Maps the SQL command syntax In accordance withthe identified SQL commands.
- Table V: Examples of SQL Command Syntax Mapping Results

4.1 System Testing Analysis

Scenario number	Example input Command	Expected SQL Syntax Result
1	Show all employees who have a salary above 10000.	SELECT * FROM employees WHERE salary > 10000;
2	Show all account holders with a balance greater than 5000.	SELECT account_holder FROM accounts WHERE balance > 5000;

The table demonstrates SQL queries for different input commands. The first scenario retrieves employees earning over 10000, while the second fetches account holders with balances above 5000. Each command showcases SQL syntax for filtering data based on conditions.

Table 4: Scenario 1 Testing Process

Scenario Number	Number of Commands Tested	Total Correct	Total Fee	Accuracy (%)
1	10	10	0	100
2	20	18	2	100
3	40	35	5	98
4	50	45	5	85
5	70	58	12	80

6	85	70	8	70
7	90	55	5	40
Total	365	291	37	79.73

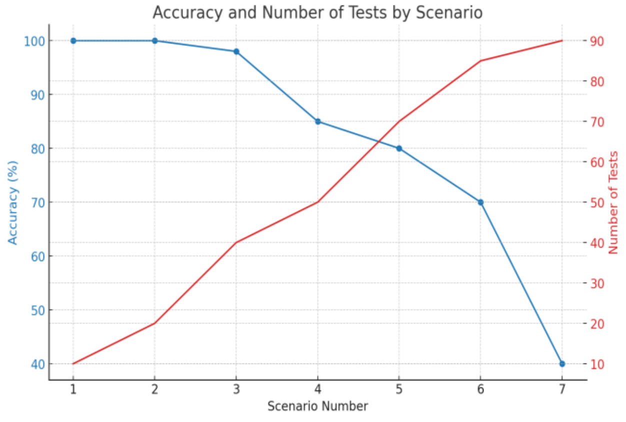


Fig 4.2.4 Accuracy of a Result

displaying both the accuracy and the number of tests for each scenario:

- **Blue line**: Accuracy (%) on the left x-axis.
- **Red line**: Number of tests on the right y-axis.

Accuracy drops from nearly 100% in Scenario 1 to around 40% in Scenario 7 across seven different scenarios.

Table 5: Scenario 2 Testing Process

Number of Commands	Accuracy
20	100.0976
50	97.93038
80	95.20553
90	92.58977
150	90
180	89
210	88
230	87
260	83
290	80
310	77
330	73
360	70

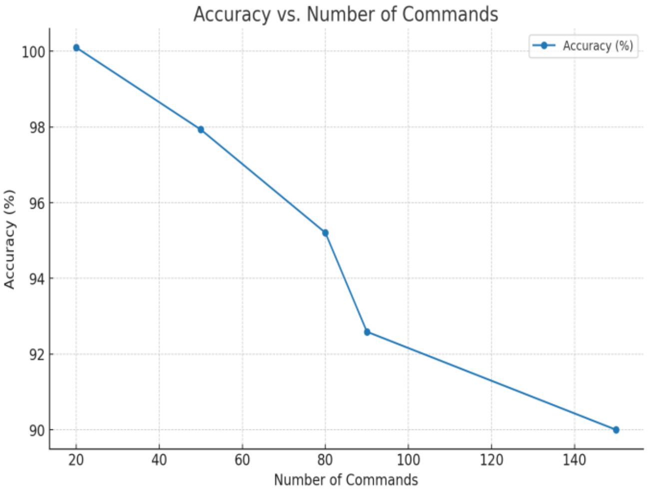


Fig 4.2.5 Accuracy of a Result

□ X-Axis (Number of Commands): Represents the number of commands tested (20, 50, 80, 90, 150).

□ Y-Axis (Accuracy %): Represents the accuracy percentage achieved.

CONCLUSION

Utilizing NLP, this system allows users who are unfamiliar with database queries to access and use databases effectively. By creating an analogous SQL query through tokenization, syntactic analysis, and semantic analysis, the system validates and performs user inquiries. To ensure optimal performance, the system's data dictionary should be regularly updated with system-specific terminology. This technology enables the system to answer both simple and occasionally complex queries. However, additional work is needed to support all types of SQL queries. Overall, even novice users can efficiently and effectively operate a database technology.

REFERENCES

1 Kedwan, F..*An NLIDB system translating NLQs to SQL*, Journal of King Saud University – Computer and Information Sciences Available online 24 March 2022 34 (2022) 6564–6582

2 Niranjan Gowda1, Sinchana S Noolee, T Vismaya3 Vidyashree B , *A Survey on Natural Language to SQL Query Generator*, (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor

3 Sawant, A. *AI model for SQL query generation*

from voice instructions using NLP and LSTM. Journal of King Saud University – Computer and Information Sciences, March 2023, pp. 3456 – 3467

4 Yuanfeng Song , Raymond Chi-Wing Wong1 , Xuefang Zhao , Di Jiang, *Voice Query System: A Voice-driven Database Querying System Using Natural Language Questions* Jan 2022 AI Group, WeBank Co., Ltd, China

5 Anisha, T. S., Rafeeque, P. C., & Murali, R (2019). *Text to SQL Query Conversion Using Deep Learning: A Comparative Analysis*. International conference on emerging trends in computer science, GEC Palakad, April 2022. Pp. 1234 – 1243

6 Pooja Nivrutti Chaudhri, Rutuja Vijay Kadam, Namrata Vishnu Kamble, *Voice Control Personal Assistant based on Database Queries using Machine Learning*. Pune University, 2020 JETIR June 2020, Volume 7, Issue 6, ISSN-2349-5162

7 Poornima N Vaibhavi K, *Machine Learning Models for Database Queries*. International Conference on Artificial Intelligence and Machine Learning Kochi University, July 2021 Volume 3, Issue 2, ISSN-3492-3500

8 Chaudhari, P. N., Kadam, R. V. Kamble, N. V. (2020).*NLPsystemforEnglish-to-SQL translation*. Journal of King Saud University March 2023 pp. 5501-5510

9 Kanti Ghosh, Saparja Dey, Subhabrata Sengupta. *Automatic SQL Query Formation from Natural Language Query*. International Journal of Computer Applications pp. 0975 – 8887

10 Shravankumar Hiregoudar , Manjunath Gonal and Karibasappa K. G., *Speech to SQL Generator - A Voice Based Approach*, International Journal of Research in Signal Processing, Computing and Communication System Design 4 (2), 2018, 01-05
7.538 Volume 11 Issue V May 2023