Baghdad Science Journal

Volume 23 | Issue 6 Article 29

6-26-2026

Complex Question Analysis in Voice Commands Using Generate SQL for Database Searching

P. Prabaswara
Ministry of Tourism of Indonesia, Indonesia, pdyanp@gmail.com

D. Wardani
Department of Data Science, Faculty of Information Technology and Data Science, Universitas Sebelas Maret, Surakarta, Indonesia, dww_ok@uns.ac.id

How to Cite this Article

Prabaswara, P. and Wardani, D. (2026) "Complex Question Analysis in Voice Commands Using Generate SQL for Database Searching," *Baghdad Science Journal*: Vol. 23: Iss. 6, Article 29.
DOI: https://doi.org/10.21123/2411-7986.5343

BAGHDAD SCIENCE JOURNAL 2026;23(6):2352–2363

Scan the QR to view the full-text article on the journal website

RESEARCH ARTICLE

Complex Question Analysis in Voice Commands Using Generate SQL for Database Searching

P. Prabaswara [1], **D. Wardani** [2,*]

[1] Ministry of Tourism of Indonesia, Indonesia
[2] Department of Data Science, Faculty of Information Technology and Data Science, Universitas Sebelas Maret, Surakarta, Indonesia

ABSTRACT

The definition of a complex question is a long sentence with many focal points that may have many answers, involve many entities, or use non-standard linguistic constructs. Complex questions can be obtained from voice commands. It will help some people with blindness or visual impairments. Voice commands have become a mandatory feature of applications, including the basic one to access databases. In this work, the Generate Structured Query Language Configuration (GSC) approach was developed to answer complex questions in the Indonesian language. The non-learning approach was chosen as the learning process needs costly computation and is inappropriate for low-resource languages. GSC can be plugged into any database system without any learning process and is implemented on Android devices. GSC succeeded in generating about 93.33% of all the questions. An accuracy of 80.01% was achieved based on experiments with 196 questions submitted to the Mondial database. The accuracy of the truth in the 114-question experiment using the IMDB database was 78.07%. The fundamental framework of GSC does not have a very high score but is still promising. Indeed, this work must be elaborated on shortly by adding a few methods to handle some limitations.

Keywords: Assistive technology, Complex query, Database, SQL, Voice question answering

Introduction

Almost all activities that require visual interaction are challenging for people with vision impairments, including accessing web-based systems that typically rely on screen and keyboard input. Unlike search engines, searching for data in databases requires structured input, which can be particularly challenging for people with Visual Impairment (VI). Therefore, enabling database access through voice commands can be beneficial for people with vision impairments. Previous work has emphasized that technologies incorporating electronic information must consider the needs of people with vision impairments[1] and all users of computer systems.[2] One everyday activity that remains challenging is database searching, which is commonly performed in universities and other organizations. In this context, questions or search inputs can be formulated as voice commands to improve accessibility. Voice-based questions are often expressed as long sentences.[3,4] Some applications, such as search engines, have enhanced their features with voice commands, which are especially helpful for people with vision impairments.

This work presents its findings in Indonesian because the proposed approach addresses specific grammatical issues of the language. Complex questions in the Indonesian language differ from those in other languages. A single question may have one primary focus among multiple possible focuses, may produce multiple answers, and may involve multiple entities.[5,6] Questions can employ non-standard language structures or slang. Indonesian slang can relate to syntax or word order, and this work focuses on the latter. To express a question, complex word order can be produced in multiple ways. For example, the

Received 17 August 2025; revised 24 February 2026; accepted 7 March 2026.
Available online 26 June 2026

* Corresponding Author.
E-mail addresses: pdyanp@gmail.com (P. Prabaswara), dww_ok@uns.ac.id (D. Wardani).

https://doi.org/10.21123/2411-7986.5343

question "Negara apakah di Benua Asia yang memiliki luas kurang dari seratus ribu kilometer persegi?" (Which country on the Asian continent has an area of less than one hundred thousand square kilometers?) can also be expressed *as "Negara yang di Asia yang luasnya di bawah seratus ribu kilometer, apa ya?"* (A country in Asia with an area below one hundred thousand square kilometers, what is that?). These examples illustrate complex questions that involve relationships between substances or events and require in-depth subject knowledge.[7,8] From a database perspective, complex questions can be answered using structured data through complex Structured Query Language (SQL) queries. Such queries typically involve multiple keywords, such as SELECT, JOIN, WHERE, AND, and OR, which allow relationships across two or more tables. However, using complex SQL is not accessible to everyone.[9,10]

Several studies have investigated Indonesian question-and-answer (QA) systems in practice.[11,12] These studies describe a QA framework consisting of three components, one of which is the Address Analyzer. In this component, a rule-based framework processes address terms such as "what," "who," "where," and "how much," and generates answers based on predefined word patterns. Yeo[13] proposed a machine learning approach to search structured health data using natural language input. In this work, complex questions are decomposed into simple factoid sub-queries, whose answers are then used as input for pattern learning and prediction. Machine learning is effective for analyzing complex questions[14,15] and for enhancing Natural Language Interfaces to Databases (NLIDB), enabling systems to adapt to new conditions. Other studies have successfully converted natural language into SQL through pattern matching and semantic matching.[16,17] In these approaches, SQL queries are generated using predefined rules and data dictionaries. The information lexicon contains semantic descriptions of attributes and relationships. The process typically involves tokenization, removal of unnecessary words, element classification, and query construction.

Question Answering System (QAS) classification[18] provides a foundation for further research and development, including machine learning–based, semantic knowledge–based, ontology-based, and pattern-matching approaches. Some languages are better supported than others, depending on the maturity and availability of resources. Several studies on Indonesian QAS[19,20] have developed open-domain QA frameworks capable of handling both factoid and non-factoid questions using monolingual approaches such as machine learning, named entity recognition, pattern matching, and semantic analysis. Factoid questions require precise answers (e.g., "who," "where," "when"), whereas non-factoid questions require descriptive or narrative answers, often associated with "how" and "why." Indonesian QA for non-factoid questions has applied tokenization, stop-word removal, and stemming, showing that non-stem keywords produce better accuracy than stem-based keywords. Other approaches have employed hyponym relations[21,22] and keyword extraction to translate queries. These integrated methods handle complex queries using antonym relations and narrower concepts, achieving results comparable to those of complex SQL queries. Additional efforts have focused on answering spoken questions,[23] including the development of various tools and APIs.[24] One study proposed using natural language to query databases directly.[25]

Comparative studies have analyzed the advantages and disadvantages of natural language approaches for database interaction, including pattern matching, syntax parsing, intermediate query generation, ontologies, and machine learning.[26,27] In these approaches, complex questions are decomposed into simpler sub-queries, and the resulting answers are used for pattern training and prediction. The semantic metadata of databases is often used to build dictionaries. Other approaches have applied natural language mapping and machine learning to handle complex questions.[28]

This work contributes by presenting DSearch, an Android-based system for querying structured databases using complex Indonesian voice commands. It further contributes an analysis of the linguistic characteristics of complex Indonesian spoken questions and their implications for database searching. A rule-based Generate SQL Configuration (GSC) approach is proposed to transform natural language queries into executable complex SQL. The proposed system improves database accessibility, particularly for users with vision impairments. Finally, this work contributes a foundational framework for future NLIDB systems that can be extended with more advanced techniques.

Materials and methods

Natural language to SQL (NL2SQL) is a concept that aims to translate a natural language query or question into Structured Query Language (SQL).[16] Meanwhile, the details of the method can be varied. As each language has a different grammar for complex questions, low-resource languages like the Indonesian language cannot; this first work utilizes the simple approach that considers the rule-based approach. The approach's main idea is to obtain the SQL template from a natural language query from

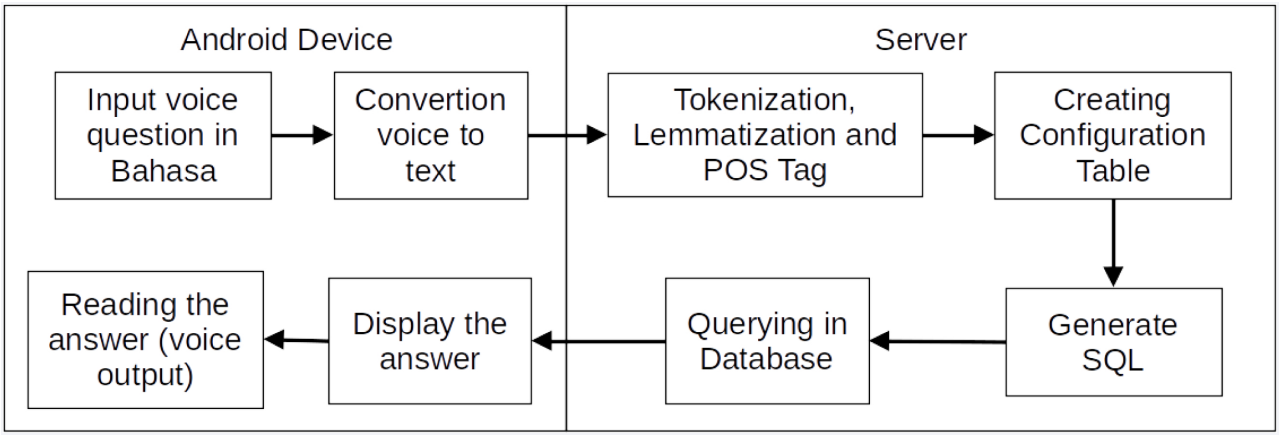

Fig. 1. The system architecture of DSearch.

a voice command. SQL is a matching process based on rules. As the rules are strict and there is a slight variation, the approach will elaborate on the rules to obtain the goal by implementing algorithms. This work develops an Android application using the Java programming language and utilizes existing tools. Fig. 1 shows the system's flow from the beginning of answering complex questions through voice commands to the formation of answers.

The whole process is implemented using several algorithms. An important algorithm is to build the configuration of the table, which will be shown here and described in the flow in Fig. 2. The input is a complex question on an Android device in the form of voice commands about things in Mondial and IMDB. An example of asking with Voice Command (in the Indonesian language) is: *"Negara apa ya yang benuanya Asia dan luas negaranya lebih besar dari 1919440-kilometer persegi?"* (What country is located on the continent of Asia, and the area of the country is more significant than 1919440 square kilometers?). Questions are in the form of voice commands, then converted into text. This step implements tokenization, including lemmatization and POS Tagging. The result will be used to fill the rule in the next step. Table 1 shows an example result of POS Tagging. An example of the outcome of Converting Voice Commands to Text can be seen in Fig. 3. Fig. 2 is implemented in the Algorithm Building Configuration Table —an example of the obtained generated SQL from the configuration table, as shown in Table 2.

Algorithm Building Configuration Table, in brief, is explained as follows:

1) Find a Table. Find candidate tables in the token list based on POS tags that can detect table names by referring to Table 1. These candidate tables are compared with tables in the database using text similarity to get the most similar table. If the table is found, add it to the configuration table on a new row and go to step (2). If it is not found, then the token is not a table.
2) Find Columns. Find candidate columns in the token list based on POS tags that can detect column names by referring to Table 1. These candidate columns are compared with columns in the database using text similarity to get the most similar columns. Column candidates can be in front of or behind the table token. If the column is found, add it to the configuration table in line with the step (1) table. If it is not found, the token is not a column, so continue with step (3).
3) Find an Operator. Find operator candidates in the token list based on POS tags capable of detecting operators by referring to Table 1. Operator candidates are behind the table token. If the operator is found, add it to the configuration table in line

Table 1. Example of tokenization results and POS tagging.

Token	POS Tag
Negara	NNO
apa	PRI
ya	INT
yang	PRR
benua	NNO
nya	PRK
Asia	NNP
dan	CCN
luas	ADJ
negara	NNO
nya	PRK
lebih	ADV
besar	ADJ
dari	PPO
1919440	NUM
km^2	NUM

Table 2. The example of obtained configuration table (in Indonesian language).

Code	Table	Column	Operator	Value	Type
negara1 (country1)	negara (country)	nama (name)			SELECT
benua1 (continent1)	benua (continent)	nama (name)		Asia	WHERE
negara1 (country1)	negara (country)	luas (wide)	>	1919440	WHERE
negara1 (country1)	negara (country)	kode (code)			JOIN
meliputi1 (consist1)	meliputi (consist)	negara (country)			JOIN
meliputi1 (consist1)	meliputi (consist)	benua (continent)			JOIN
benua1 (continent1)	benua (continent)	nama (name)			JOIN

with the step (1) table. If it is not found, the token is not an operator and continues with step (4).

4) Find a value. Find candidate values in the token list based on POS tags that can detect values by referring to Table 1. Candidate values are behind the table token. If the value is found, add it to the configuration table in line with the step (1) table.
5) Repeat steps (1), (2), (3), and (4). Repeat steps (1), (2), (3), and (4), and place them on a new line until all tokens have been analyzed.
6) Find Implicit Values and Tables. If there is a value for which the table cannot be found, search for the entity (table) using the entity analyzer from the TextRazor API. Because the TextRazor API cannot handle Indonesian, questions must be converted into English using the Google Translate API. Add the table and value to the configuration table as a new row if they are found.
7) Find Relationships for Each Table. Find related tables and columns using SQL based on the relationships formed in the database in the form of Primary Keys, Foreign Keys, and Table Reference relationships. Then, add them to the configuration table as a new row.
8) Add Unfilled Columns. If there is a row in the configuration table whose columns have yet to be filled in, check whether there are tables in the database with the same table and column names using text similarity. Hence, it will get the most similar columns. If there are any, then fill in the configuration table with those columns. If none are found, fill in the configuration table with a NAME column.
9) Correct or fill in the unfilled operators. If an operator is in the configuration table as a word, change it to an operator symbol based on a specific rule. If a row has a value and does not have an operator, fill in the operator with the default operator as the equal symbol (=).
10) Add Code. Add Code to the configuration table as a table name plus a unique number in the corresponding row.
11) Add Type. Add the Type to the configuration table with the following conditions: a) SELECT is the first row of the configuration table. b) WHERE is a row in the configuration table that has codes, tables, columns, operators, and values. c) JOIN is a row in the configuration table with Code, tables, and columns.
12) Improve Values. If a value in the configuration table does not exist in the database, look for the closest value among all the values using text similarity.

After the configuration table is obtained, it will generate SQL. Generally, the process is to manage each column in the configuration table using the SQL value. Table 3 is the template that is preserved.

SELECT DISTINCT *negara1.nama*
FROM *negara1*
INNER JOIN *meliputi1* ON *negara1.kode=meliputi1.negara* INNER JOIN *benua1* ON *meliputi1.benua=benua1.nama*
WHERE *benua1.nama= "Asia"* OR *benua1.nama* LIKE *"Asia"* AND *negara1.luas > 1919440*

After generating SQL, the next step is to submit the query to the database. Example Search results for the input question are India, Kazakhstan, Russia, Saudi Arabia, China, and Australia. Generating SQL is used to obtain the table configuration. After the results are found, they are displayed on the Android device, as shown in Fig. 3. Australia is one of the answers, even though we know it is not a country on the Asian continent. According to the database, 10% of Australia's territory is on the Asian continent. The Mondial database's schema makes Australia available as an answer as well.

Results and discussion

The experiment is conducted in two steps: (i) the evaluation of the technical aspects of the system and (ii) the evaluation of the prototype by the VI users. The first experiment aims to measure the accuracy of the proposed approach. Meanwhile, the second experiment aims to get feedback from VI users. In the first experiment, Table 4 shows examples of test scenarios

Algorithm: Building Configuration Table

```
Input: token of question (t), entity in question (e)
Output: configuration table (ct)
Step:
Begin
i ←0, k←0
for each t do
    if isTableOrColumn(t[i] then
        ct ←findSimilarWordInTable(t[i])
        if ct NOT NULL then
                insertToConfigurationTable(ct, k, 1)
    if ct[k][1] then
        if isRelatedColumn(i-1,i,t) then
        insertToConfigurationTable(t[i-1], k, 2)
    j ←t+1
    if k NOT 0 then
        if < sizeOf(t) AND NOT isPreposition(t[j]) AND NOT isTableOrColumn(t[j])
        AND isTable(t[j]) then
                if sRelatedColumn(t[j],t[i] AND NOT isSet(ct[k][4])) then
                insertToConfigurationTable(t[j],k,2)
    if NOT isSet(ct[k][3]) AND isOperator(t[j]) then
        insertToConfigurationTable(t[j],k,3)
    if NOT isSet(ct[k][4]) then
        if j<sizeOf(t) then
                if isTableOrColums(t[j]) AND NOT isTable(t[j]) OR isValue(t[j]) then
                    if sizeOf(word) NOT 0 then
                        if t[j-1][postag]!=t[j] OR t[j][postag]=NUM then
                        word←word+t[j]
                j ←j+1
        if word NOT '' '' then
                insertToConfigurationTable(word,k,4)
        k ←k+1, i ←j+1
    if isSet(e) then
        for each e do
                for type ∈ e do
                type←stringToLower(type)
    if isTable(type) then
        index ← finding index ∈ ct
        if index > then
                if NOT isSet(ct([index][4])) then
                        insertToConfigurationTable(e, index, 4)
                        index ← sizeOf(configurations)
    insertToConfigurationTable(type,index,1)
    insertToConfigurationTable(type,index,4)
for config at ∈ ct do
    Input column 1 at config to array ct
    ct ←findunique row at ct
    i ←0, j ←0
    if i < sizeOf(ct) then
    if j < sizeOf(ct) then
        relationTable ← find relationship ct i to ct j
        k ←1
        if k≤ sizeOf(relationshipTable)/2 then
                index ← sizeOf(ct)
                insertToConfigurationTable(reationshipTable[ct k],index,1)
                insertToConfigurationTable(relationshipTable[column k],index,2)
        k ←k+1, j ←j+1
    i ←i+1, j ←i+1
    for config ∈ ct do
        insertToConfigurationTable(generateCode(ct[i][1]),i,0)
    if NOT isSet(ct[i][2]) AND (i=0 OR isSet(ct[i][4])) then
        insertToConfigurationTable(generateColumn(ct[i][1],i,2))
    if isSetconfigurationTable[i][4] then
        insertToConfigurationTable(generateOperator(isSet(ct[i][3]),ct[i],[3]:null),i,3)
        value ← findSimilarWordInValue(configurationTable[i][4])
        if value NOT null then
                insertToConfigurationTable(value,i,4)
                insertToConfigurationTable(generateType(i,ct[i]),i,5)
End
```

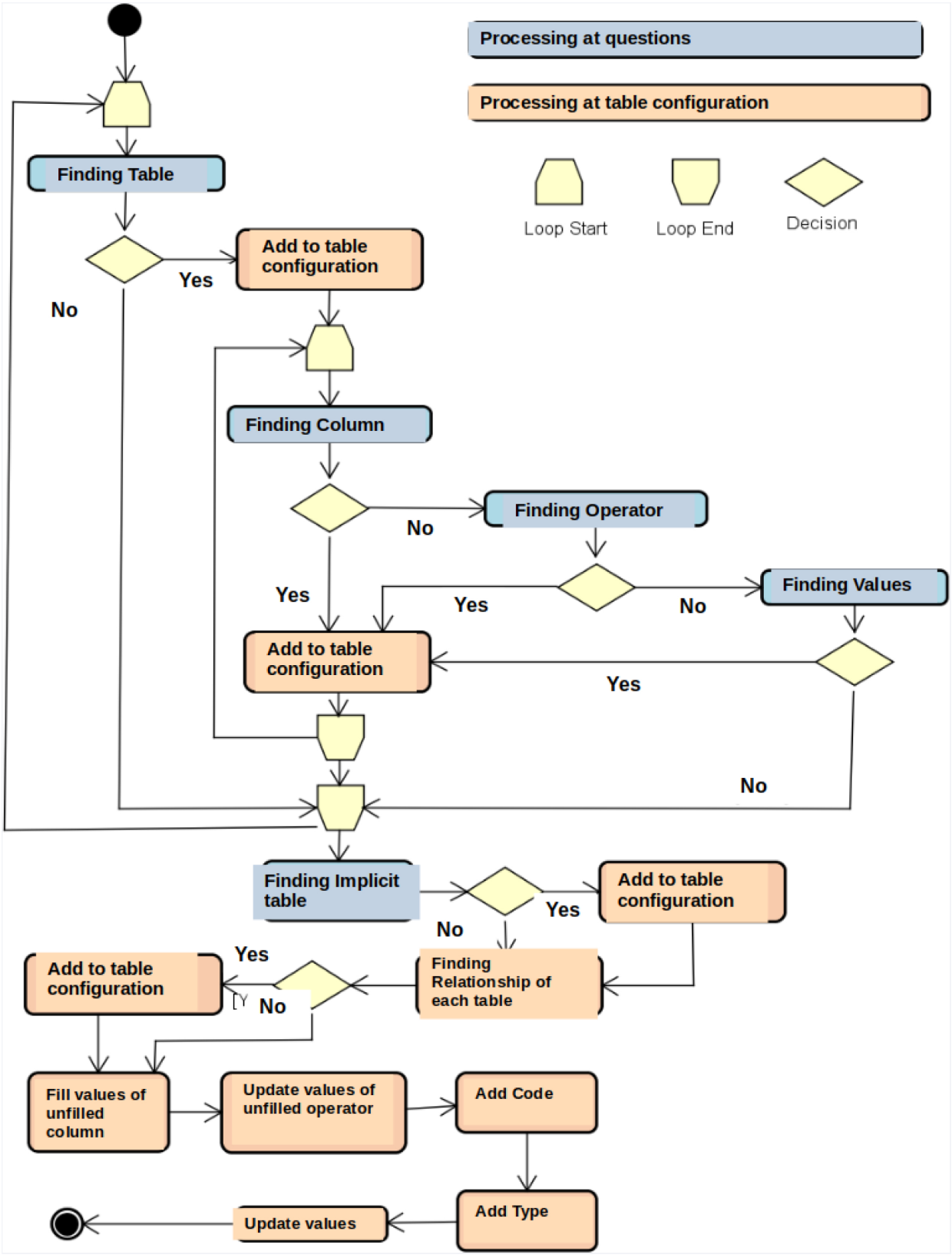

Fig. 2. The flowchart of algorithm in building configuration table.

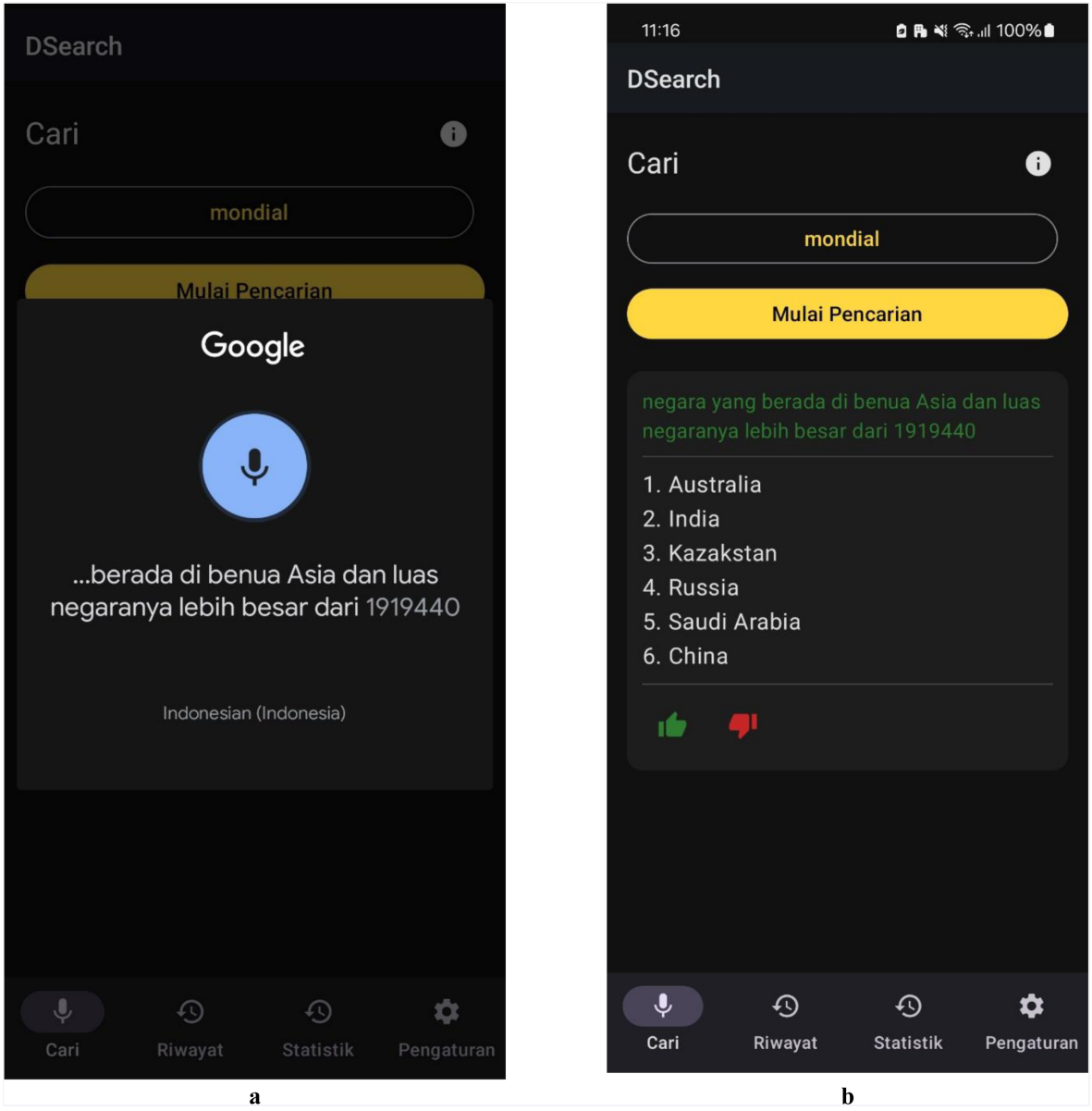

Fig. 3. The running step (a). Input the voice question (in Indonesian language), (b) the result of converting it into text, and an example of displaying and reading answers in DSearch.

Table 3. The template SQL for string and number answers.

Template 1	Template 2
SELECT DISTINCT kode(1).kolom(1) FROM tabel(1) kode(1) INNER JOIN tabel(N + 1) kode(N + 1) ON kode(N).kolom(N)=kode(N + 1).kolom(N + 1)... WHERE kode(N).kolom(N)operator(N)nilai(N) OR kode(N).kolom(N) LIKE “nilai(N)”) AND..	SELECT DISTINCT kode(1).kolom(1) FROM tabel(1) kode(1) INNER JOIN tabel(N + 1) kode(N + 1) ON kode(N).kolom(N)=kode(N + 1).kolom(N + 1)... WHERE kode(N).kolom(N)operator(N)nilai(N) AND..

that run concurrently and how conditions from previous test processes affect subsequent test processes. It combines each step’s success (S) or failure (F). The experiment resulted in trial statuses (1) through (32). The experiment starts with analyzing complex questions and ends with evaluating the correctness of the answers. Experiments are performed using two datasets, Mondial and IMDB. Table 5 also shows the

Table 4. Example of the experiment scenario. There are 32 scenarios where the combination succeeded (S) or failed (F) for five features.

NoCode	Complex Question Analysis	Generate SQL	SQL Execution on Database	Answer Appearance	Evaluation
1	S	S	S	S	S
7	F	S	F	F	S
17	F	S	S	S	S
23	F	S	F	F	S
32	F	F	F	F	F

Table 5. Write the table title here.

No	Database	No Question	Total Question
1	Mondial	54	212
2	IMDB	32	121

number and the total number of questions. The entire number of questions shows that the same address may contain different expressions. A significant number of NoCode values are codes 1, 4, 8, 17, 18, 20, and 24.

Table 6 shows some of the results of the experiments performed. The same question code indicates that the questions have the same focus but are in a different order. From Table 6, the first five questions (on the Mondial dataset) are the same. Therefore, this work considers that voice questions can vary in word choices and style, either formal or non-formal. One question, but in multiple forms of voice questions. The same condition is shown in the rest of the questions (on the IMDB dataset). The questions *"film yang ada di Indonesia"* (films in Indonesia) and *"Tolong carikan daftar film apa aja yang tayang di Indonesia"* (please find a list of films showing in Indonesia) have the same meaning. This work can handle this challenge.

The results and discussion should be written as clearly and as detailed as possible. Authors may separate or merge the results with the discussion.

The experiment results in Table 7 show that out of 212 questions on the Mondial data, 196 (92.45%) were answered, and 114 (94.21%) of the 121 questions on the IMDB data. It indicates that almost all questions have been answered. Other questions could have been improved in the generation process. Therefore, the average for generating SQL is 93.33%. Meanwhile, the accuracy of each data point reached 80.10% and 78.07% on Mondial and IMDB, respectively. The following is a summary of the limitations of GSC:

(i) Text similarity cannot find data that is similar between data in Indonesian and English, or the data does not exist in the database,
(ii) The table (entity) in the question is far from the column, operator, and value. It happens if the order of words in a question is unique,
(iii) The table in the database does not have a default column name that is the same as the table name or does not have a column name that is needed,
(iv) Incorrect POS tagging. It happens and influences the wrong step after preprocessing.
(v) Errors in sorting relations between tables, causing the JOIN order to be reversed in SQL.
(vi) There are multiple interpretations of an entity. Example (in the Indonesian language): "*Negara mana yang memiliki Laut Jawa*" (Which country owns the Java Sea). The Java Sea can be considered the Java Sea and the Java Island.

In the second experiment, there were ten VI respondents to test the application prototype. The small number of respondents is due to only a few agreeing to test the application. Moreover, the VI community has a small number (less than 20) of members. The measure is related to a quality measurement that provides four classes, with values 1 to 4 (a higher value means better quality). Table 8 shows the details of the result. Respondents input the voice question, which differs from the questions in the first experiment but is in the same domain. The result based on respondents' opinions can be narrated as the prototype of the proposed approach being valuable and easy to use. However, the respondents expect better results from their questions. Respondents also agree that the prototype has a fast duration from the time the input voice question is given until the time they get the answer. From all the results of the experiments, it can be concluded that this approach is promising for use in assisting VI people. The other plus point of this work is that GSC is plugged into any database system. However, a few aspects could be improved, such as combining grammar-based rules with a machine-learning-based approach that can handle various scenarios. The work should conduct experiments on more databases to measure the robustness of this approach. Additionally, as a prominent work in the Indonesian language, this work does not compare itself with other works, and it also highlights crucial notes that should be addressed in future work.

Table 6. Example of obtained results from a similar question, the same focus but a different expression.

NoCode	Voice Question (in Indonesian Language)	Obtained SQL
1	*Apa nama danau di Asia yang punya luas Danau lebih besar dari 3000 kilo* (What is the name of a lake in Asia that has a lake area greater than 3000 kilos?)	SELECT DISTINCT *danau1.nama* FROM *danau danau1* INNER JOIN *geo_danau geo_danau1* ON *danau1.nama=geo_danau1.danau* INNER JOIN *negara negara1* ON *geo_danau1.negara=negara1.kode* INNER JOIN *meliputi meliputi1* ON *negara1.kode=meliputi1.negara* INNER JOIN *benua benua1* ON *meliputi1.benua=benua1.nama* WHERE *danau1.luas* > 3000 AND (*benua1.nama="Asia"* OR *benua1.nama* LIKE *'%Asia%'*)
1	*Danau apa sih di Asia yang luas Danau nya lebih dari 2000 km2* (What Lake in Asia has a lake area of more than 2000 km2?)	SELECT DISTINCT *danau1.nama* FROM *danau danau1* INNER JOIN *geo_danau geo_danau1* ON *danau1.nama=geo_danau1.danau* INNER JOIN *negara negara1* ON *geo_danau1.negara=negara1.kode* INNER JOIN *meliputi meliputi1* ON *negara1.kode=meliputi1.negara* INNER JOIN *benua benua1* ON *meliputi1.benua=benua1.nama* WHERE *danau1.luas* > 2000 AND (*benua1.nama="Asia"* OR *benua1.nama LIKE '%Asia%'*)
1	*Danau apa ya di Asia yang luas Danau nya lebih besar dari 10000* (What Lake in Asia is bigger than 10,000 lakes?)	SELECT DISTINCT *danau1.nama* FROM *danau danau1* INNER JOIN *geo_danau geo_danau1* ON *danau1.nama=geo_danau1.danau* INNER JOIN *negara negara1* ON *geo_danau1.negara=negara1.kode* INNER JOIN *meliputi meliputi1* ON *negara1.kode=meliputi1.negara* INNER JOIN *benua benua1* ON *meliputi1.benua=benua1.nama* WHERE *danau1.luas* > 10000 AND (*benua1.nama="Asia"* OR *benua1.nama* LIKE *'%Asia%'*)
1	*Sebutkan Danau apa aja yang ada di Asia tapi luas Danau nya lebih besar dari 6000* (Name any lake in Asia but the area of the lake is greater than 6000)	SELECT DISTINCT *danau1.nama* FROM *danau danau1* INNER JOIN *geo_danau geo_danau1* ON *danau1.nama=geo_danau1.danau* INNER JOIN *negara negara1* ON *geo_danau1.negara=negara1.kode* INNER JOIN *meliputi meliputi1* ON *negara1.kode=meliputi1.negara* INNER JOIN *benua benua1* ON *meliputi1.benua=benua1.nama* WHERE *danau1.luas* > 6000 AND (*benua1.nama="Asia"* OR *benua1.nama* LIKE *'%Asia%)'*
1	*Danau apa ya yang ada di Eropa tapi luas danaunya lebih dari 2000 km2* (What lakes are there in Europe, but the lake area is more than 2000 km2)	SELECT DISTINCT *danau1.nama* FROM *danau danau1* INNER JOIN *geo_danau geo_danau1* ON *danau1.nama=geo_danau1.danau* INNER JOIN *negara negara1* ON *geo_danau1.negara=negara1.kode* INNER JOIN *meliputi meliputi1* ON *negara1.kode=meliputi1.negara* INNER JOIN *benua benua1* ON *meliputi1.benua=benua1.nama* WHERE *danau1.luas* > 2000 AND (*benua1.nama="European"* OR *benua1.nama* LIKE *'%European%'*)
1	*Beritahu saya film apa aja sih yang ada di India* (tell me what films are there in India)	SELECT DISTINCT *film1.film* FROM *film film1* INNER JOIN *negara negara1* ON *film1.idfilm=negara1.idfilm* WHERE (*negara1.negara="India"* OR *negara1.negara* LIKE *'%India%'*)
1	*Film apakah yang ada di Kanada* (What movies are in Canada)	SELECT DISTINCT *film1.film* FROM *film film1* INNER JOIN *negara negara1* ON *film1.idfilm=negara1.idfilm* WHERE (*negara1.negara="Canada"* OR *negara1.negara* LIKE *'%Canada%'*)
4	*Ada film apa aja sih di Malaysia* (what films are there in Malaysia?)	SELECT DISTINCT *film1.film* FROM *film film1* INNER JOIN *negara negara1* ON *film1.idfilm=negara1.idfilm* WHERE (*negara1.negara="Malay"* OR *negara1.negara* LIKE *'%Malay%'*)
4	*Film yang ada di Indonesia* (films in Indonesia)	SELECT DISTINCT *film1.film* FROM *film film1* INNER JOIN *negara negara1* ON *film1.idfilm=negara1.idfilm* WHERE (*negara1.negara="Indonesian"* OR *negara1.negara* LIKE *'%Indonesian%'*)
4	*Tolong carikan daftar film apa aja yang tayang di Indonesia* (Please find a list of films showing in Indonesia)	SELECT DISTINCT *film1.film* FROM *film film1* INNER JOIN *negara negara1* ON *film1.idfilm=negara1.idfilm* WHERE (*negara1.negara="Indonesian"* OR *negara1.negara* LIKE *'%Indonesian%'*)
4	*Tolong carikan daftar film apa aja yang terbit di Indonesia* (Please find a list of films published in Indonesia)	SELECT DISTINCT *film1.film* FROM *film film1* INNER JOIN *negara negara1* ON *film1.idfilm=negara1.idfilm* WHERE (*negara1.negara="Indonesian"* OR *negara1.negara* LIKE *'%Indonesian%'*)

Table 7. The results of prototype's evaluation.

No	Database (no of question)	S	F
1	Mondial (196)	157 (80,10%)	39 (19,90%)
2	IMDB (114)	89(78,07%)	25 (21,93%)

Table 8. The result of respondent's evaluation.

Measurement	Score 1	Score 2	Score 3	Score 4
Helpful	0%	0%	0%	100%
Easy to use	0%	0%	305	70%
Satisfy	0%	10%	40%	50%
Speed	0%	0%	30%	70%

Conclusion

This work has enabled the analysis of complex questions as input for structured data searches using voice commands with the Generate SQL Configuration approach. The idea is implemented in an Android-based application named DSearch. The question analysis process is carried out by creating a configuration

table based on the discovery rules of the database query component in the question. By asking 196 complex questions for the Mondial Database and 114 questions for the IMDB Database, an accuracy of 80.10% was obtained for the Mondial Database and 78.07% for the IMDB Database. Several of these questions are developments from the initial questions made in other forms, such as by changing the order of words, sentences, or language. Future investigations are anticipated to enable the preparation of complex questions with the address center, either within the center or at the end of a sentence, in the form of Indonesian voice commands. The approach can discover proportional words, discover inquiry components, and build queries using machine learning. Expansion, moreover, creates more varieties of inquiries to handle a greater number of questions. Respondents' feedback returns to the conclusion that the prototype of the proposed approach is helpful and promising in assisting people with vision impairments. Future work will focus on improving the method to handle a broader range of query intentions by utilizing scalable machine learning methods or a combination of rule-based and machine learning models. This future work aims to overcome the limitations of the current work.

Acknowledgment

Thank you to Josephine Angelia Pramudya for the contributions to managing the user testing and the product poster.

Authors' declaration

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are ours. Furthermore, any Figures and images that are not ours have been included with the necessary permission for republication, which is attached to the manuscript.
- No animal studies are present in the manuscript.
- No human studies are present in the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee at the University of Sebelas Maret.

Authors' contributions statement

P. P contributed significantly to the initial development phase of the project, focusing on designing the algorithm and translating it into a functional codebase. This included formulating the core logic, implementing the necessary computational structures, and conducting preliminary experiments to validate the algorithm's effectiveness prior to deployment. Meanwhile, D.W. played a crucial role in both the development and post-deployment stages. Her contributions included co-designing the algorithm, preparing and organizing the dataset, and drafting the manuscript to document the methodology and findings. Additionally, she was responsible for conducting thorough testing after deployment, ensuring the algorithm performed reliably in real-world conditions. Together, their collaborative efforts ensured the project's success from design through to implementation and evaluation.

Funding statement

This work was supported by Kemdiktisaintek Grant on Research Commercialization - Model and Prototype Testing under contract number 2315.1/UN27.22/PT.01.03/2025.

References

1. Senjam SS, Manna S, Bascaran C. Smartphones-based assistive technology: Accessibility features and apps for people with visual impairment, and its usage, challenges, and usability testing. Clin Optom (Auckl) 2021 Nov;13:311–322. https://doi.org/10.2147/OPTO.S336361.
2. Arslantas TK, Gul A. Digital literacy skills of university students with visual impairment: A mixed-methods analysis. Educ Inf Technol (Dordr) 2022 May;27(4):5605–5625. https://doi.org/10.1007/s10639-021-10860-1.
3. Sa N, Yuan X (Jenny). Examining User Perception and Usage of Voice Search. Data Inf Manag 2021 Jan;5(1):40–47. https://doi.org/10.2478/dim-2020-0046.
4. Keyvan K, Huang JX. How to Approach Ambiguous Queries in Conversational Search: A Survey of Techniques, Approaches, Tools, and Challenges. ACM Comput Surv 2023 Dec;55(6):1–40. https://doi.org/10.1145/3534965.
5. Deng Y, Zhang W, Xu W, Shen Y, Lam W. Nonfactoid Question Answering as Query-Focused Summarization With Graph-Enhanced Multihop Inference. IEEE Trans Neural Netw Learn Syst 2024 Mar;35(8):11231–11245. https://doi.org/10.1109/TNNLS.2023.3258413.
6. Lan Y, He G, Jiang J, Jiang J, Zhao WX, Wen J-R. Complex Knowledge Base Question Answering: A Survey. IEEE Trans Knowl Data Eng 2023 Nov;35(11):11196–11215. https://doi.org/10.1109/TKDE.2022.3223858.
7. Zhu Y, Wang X, Chen J, Qiao S, Ou Y, Yao Y, *et al.* LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities. World Wide Web 2024 Sep;27(5):58. https://doi.org/10.1007/s11280-024-01297-w.
8. Li C, Wang Y, Wu Z, Yu Z, Zhao F, Huang S, *et al.* MultiSQL: A Schema-Integrated Context-Dependent Text2SQL Dataset with Diverse SQL Operations. In: Findings of the Association for Computational Linguistics ACL 2024. Association for Computational Linguistics: Stroudsburg, PA, USA, 2024 Aug, pp 13857–13867. https://doi.org/10.18653/v1/2024.findings-acl.823.

9. Taipalus T. The effects of database complexity on SQL query formulation. J Syst Softw 2020;165:110576. https://doi.org/10.1016/j.jss.2020.110576.
10. Miedema D, Fletcher G, Aivaloglou E. Expert Perspectives on Student Errors in SQL. ACM T Comput Educ. 2023 Dec;23(1):1–28. https://doi.org/10.1145/3551392.
11. Hanifah AF, Kusumaningrum R. Non-Factoid Answer Selection in Indonesian Science Question Answering System using Long Short-Term Memory (LSTM). Procedia Comput Sci 2021 Jan;179:736–746. https://doi.org/10.1016/j.procs.2021.01.062.
12. Pramana R, Prasojo RE. Improving Low-Resource Question Answering with Cross-Lingual Data Augmentation Strategies. In: 2022 10th International Conference on Information and Communication Technology (ICoICT). IEEE, 2022 Aug, pp 110–115. https://doi.org/10.1109/ICoICT55009.2022.9914847.
13. Yeo H. A Machine Learning Based Natural Language Question and Answering System for Healthcare Data Search using Complex Queries. In Conf Conference: IEEE International Conference on Big Data (Big Data). 2018. pp 2467–2474. https://doi.org/10.1109/BigData.2018.8622448.
14. Rogers A, Gardner M, Augenstein I. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. ACM Comput Surv 2023 Feb;55(10):1–45. https://doi.org/10.1145/3560260.
15. Liu Y, Pu H, Sun D-W. Efficient extraction of deep image features using convolutional neural network (CNN) for applications in detecting and analysing complex food matrices. Trends Food Sci Technol 2021 Jul;113:193–204. https://doi.org/10.1016/j.tifs.2021.04.042.
16. Kim H, So B-H, Han W-S, Lee H. Natural language to SQL: Where are we today?. Proc VLDB Endow. 2020;13(10):1737–1750. https://doi.org/10.14778/3401960.3401970.
17. Fu H, Liu C, Wu B, Li F, Tan J, Sun J. CatSQL: Towards Real World Natural Language to SQL Applications. Proc VLDB Endow. 2023;16(6):1534-1547. https://doi.org/10.14778/3583140.3583165.
18. Roy PK, Saumya S, Singh JP, Banerjee S, Gutub A. Analysis of community question-answering issues via machine learning and deep learning: State-of-the-art review. CAAI Trans Intell Technol 2023 Mar;8(1):95–117. https://doi.org/10.1049/cit2.12081.
19. Loukachevitch NV. Automatic Methods for Extracting Taxonomic Relationships from Texts. Pattern Recog. Image Anal 2023 Sep;33(3):398–406. https://doi.org/10.1134/S1054661823030276.
20. Suhartono D, Majiid MRN, Fredyan R. Towards automatic question generation using pre-trained model in academic field for Bahasa Indonesia. Educ Inf Technol (Dordr) 2024 Nov;29(16):21295–21330. https://doi.org/10.1007/s10639-024-12717-9.
21. Lezama-Sánchez AL, Tovar Vidal M, Reyes-Ortiz JA. An Approach Based on Semantic Relationship Embeddings for Text Classification. Mathematics 2022 Nov;10(21):4161. https://doi.org/10.3390/math10214161.
22. Pinto O, Gole S, Srushti HP, Madasamy AK. Relation Extraction: Hypernymy Discovery Using a Novel Pattern Learning Algorithm. SN Comput Sci 2023 Sep;4(6):730. https://doi.org/10.1007/s42979-023-02161-w.
23. Zhao Z, Jiang Y, Liu H, Wang Y, Wang Y. LibriSQA: A Novel Dataset and Framework for Spoken Question Answering with Large Language Models. In: IEEE Trans Artif Intell. 2024;1–12. https://doi.org/10.1109/TAI.2024.3427069.
24. O'Brien K, Liggett A, Ramirez-Zohfeld V, Sunkara P, Lindquist LA. Voice-controlled intelligent personal assistants to support aging in place. J Am Geriatr Soc 2020 Jan;68(1):176–179. https://doi.org/10.1111/jgs.16217.
25. Maddigan P, Susnjak T. Chat2VIS: Generating Data Visualizations via Natural Language Using ChatGPT, Codex and GPT-3 Large Language Models. IEEE Access. 2023;11:45181-45193. https://doi.org/10.1109/ACCESS.2023.3274199.
26. Grossman Liu L, Grossman RH, Mitchell EG, Weng C, Natarajan K, Hripcsak G, *et al.* A deep database of medical abbreviations and acronyms for natural language processing. Sci Data 2021 Jun;8(1):149. https://doi.org/10.1038/s41597-021-00929-4.
27. Torregrosa J, Bello-Orgaz G, Martínez-Cámara E, Ser J Del, Camacho D. A survey on extremism analysis using natural language processing: Definitions, literature review, trends and challenges. J Ambient Intell Humaniz Comput 2023 Aug;14(8):9869–9905. https://doi.org/10.1007/s12652-021-03658-z.
28. Zhao L, Alhoshan W, Ferrari A, Letsholo KJ, Ajagbe MA, Chioasca E-V, *et al.* Natural Language Processing for Requirements Engineering. ACM Comput Surv 2022 Apr;54(3):1–41. https://doi.org/10.1145/3444689.

تحليل الأسئلة المعقّدة في أوامر الصوت باستخدام توليد استعلامات SQL للبحث في قواعد البيانات

ب. باراموديتايا[1]، د. واردانى[2]

[1]قسم المعلوماتية، كلية تكنولوجيا المعلومات وعلوم البيانات، جامعة سبيلاس مارات، سوراكارتا، إندونيسيا.
[2] قسم علم البيانات، كلية تكنولوجيا المعلومات وعلوم البيانات، جامعة سبيلاس مارات، سوراكارتا، إندونيسيا.

الخلاصة

يُعرَّف السؤال المعقّد بأنه جملة طويلة تحتوي على عدة نقاط محورية قد تكون لها إجابات متعددة، أو تتضمن عدة كيانات، أو تستخدم تراكيب لغوية غير معيارية. ويمكن الحصول على الأسئلة المعقّدة من خلال أوامر الصوت، مما يساعد بعض الأشخاص من ذوي الإعاقة البصرية أو المكفوفين. وقد أصبحت الأوامر الصوتية ميزة أساسية في التطبيقات، بما في ذلك استخدامها للوصول إلى قواعد البيانات. في هذا العمل، تم تطوير منهجية توليد استعلامات لغة الاستعلامات المهيكلة (SQL) المُهيّأ (GSC) للإجابة عن الأسئلة المعقّدة باللغة الإندونيسية. وقد تم اختيار النهج غير القائم على التعلم الآلي لأن عملية التعلم تتطلب حسابات مكلفة وغير مناسبة للغات منخفضة الموارد. يمكن دمج نظام GSC في أي نظام قواعد بيانات دون الحاجة إلى عملية تعلم، كما يمكن تنفيذه على أجهزة أندرويد. نجح نظام GSC في توليد حوالي 93.33% من جميع الأسئلة. وحقق دقة بلغت 80.01% بناءً على تجارب شملت 196 سؤالاً تم تقديمها إلى قاعدة بيانات Mondial. كما بلغت دقة صحة النتائج في تجربة مكونة من 114 سؤالاً باستخدام قاعدة بيانات IMDB نسبة 78.07%. وعلى الرغم من أن الإطار الأساسي لـ GSC لا يحقق درجات عالية جداً، إلا أنه لا يزال واعداً. ومن المؤكد أن هذا العمل يحتاج إلى تطوير قريب من خلال إضافة بعض الأساليب لمعالجة بعض القيود.

الكلمات المفتاحية: التكنولوجيا المساعدة، استعلامات معقدة، قواعد البيانات، SQL، الإجابة على الأسئلة الصوتية.