VoiceQuerySystem: A Voice-driven Database Querying System Using Natural Language Questions

Yuanfeng Song[1,2], Raymond Chi-Wing Wong[1], Xuefang Zhao[2], Di Jiang[2]
[1]The Hong Kong University of Science and Technology, Hong Kong, China [2]AI Group, WeBank Co., Ltd, China
{songyf,raywong}@cse.ust.hk

ABSTRACT

With recent development in natural language processing (NLP) and automatic speech recognition (ASR), voice-based interfaces have become a necessity for applications such as chatbots, search engines, and databases. In this demonstration, we introduce VoiceQuerySystem, a voice-based database querying system that enables users to conduct data operations with natural language questions (NLQs). Different from existing voice-based interfaces such as SpeakQL or EchoQuery, which restricts the voice input to be an exact SQL or follow a pre-defined template, VoiceQuerySystem attempts to achieve data manipulation via common NLQs, and thus does not require the user's technical background in SQL language.

The underlying techniques in VoiceQuerySystem is a new task named *Speech-to-SQL*, which aims to understand the semantic in speech and then translate it into SQL queries. We explore two proposed approaches - the cascaded one and the end-to-end (E2E) one towards speech-to-SQL translation. The cascaded method first converts the user's voice-based NLQs into text by a self-developed ASR module, and then conducts downstream SQL generation via a text-to-SQL model (i.e., IRNet). In contrast, the E2E method is a novel neural architecture named *SpeechSQLNet* designed by us, which converts the speech signals into SQL queries directly without the middle medium as text. Extensive experiments and demonstrations validate the rationale of the speech-to-SQL task and the effectiveness of the proposed SpeechSQLNet model. To the best of our knowledge, this is the first system that provides a voice-based querying functionality on DBMS from common NLQs.

CCS CONCEPTS

• Information systems → Structured Query Language; **• Human-centered computing → Accessibility systems and tools**;

KEYWORDS

speech-to-SQL, relational database, SQL query generation, voice-based interface, speech-driven querying system

ACM Reference Format:
Yuanfeng Song, Raymond Chi-Wing Wong, Xuefang Zhao, Di Jiang. 2022. VoiceQuerySystem: A Voice-driven Database Querying System Using Natural Language Questions. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3514221.3520158

1 INTRODUCTION

With recent development in natural language processing (NLP) and automatic speech recognition (ASR), voice-based interfaces have become a new trend for applications such as chatbots, search engines, and databases. Since a large volume of data is stored in the relational database, it would be very helpful for non-technical users to conduct data analysis with the voice-based interfaces using natural language questions (NLQs) rather than SQL queries.

In the community, there are already a substantial amount of studies on voice-based interfaces for databases. For example, Utama *et al.* [9] designed an EchoQuery system to support querying the database with voice-based commands. However, these systems usually restrict the spoken query to be an exact SQL query (e.g., "`Select Salary From Employees Where Name Equals John`" by SpeakQL [4]) or strictly follow some pre-defined templates (e.g., "`What is the {Aggregation} {Columns(s)} of {Table(s)}?`" by EchoQuery). None of them achieved translating common NLQs (i.e., questions without being restricted by any template or SQL grammar) into SQL queries, which is an easier and flexible way for user-database interaction that requires limited SQL background.

To achieve the aforementioned goal, this demonstration introduces VoiceQuerySystem, a voice-based database querying system that attempts to handle NLQs on DBMS. VoiceQuerySystem is built on a new task named *Speech-to-SQL* together with a constructed dataset named *SpeechQL* [6] proposed by us, which aims to understand the information conveyed by human speech and directly convert it into SQL queries. The success of VoiceQuerySystem relies on the seamless integration of multiple ASR and NLP technologies. The backbone SQL generation techniques in VoiceQuerySystem can be categorized into two proposed approaches - the cascaded one and the end-to-end (E2E) one. The cascaded method first converts the user's voice questions into text by a self-developed ASR module, and then conducts downstream SQL generation via a text-to-SQL model like EditSQL [13] or IRNet [1]. In contrast, the E2E method in VoiceQuerySystem is a novel neural architecture named *SpeechSQLNet* [6] designed by us, which consists of advanced speech encoder, schema encoder, SQL-aware decoder, and pre-training mechanisms towards accurate SQL query generation from common NLQs. Serving as a bridge between end-users and the DBMS systems, VoiceQuerySystem can not only help users with a non-technical background to manipulate the data, but also significantly improve the efficiency in interacting with DMBS. We expect that VoiceQuerySystem will inspire more research on integrating state-of-the-art NLP and ASR techniques to improve database usability.

2 RELATED WORK

We briefly survey the related work from two closely related areas, text-to-SQL and voice-driven querying systems.

Text-to-SQL. Text-to-SQL has drawn great attention from the research communities since it provides a text-based interface for the database. Existing studies generally belong to two approaches: the rule-based one and the learning-based one. The classical rule-based approach includes studies such as QUICK [12], SINA [5], and NLQ/A [14]. With the popularity of deep neural networks (DNNs), methods such as Seq2SQL [15], EditSQL [13], and IRNet [1] were developed with advanced neural network structures working in an E2E style. Commonly-used public benchmark datasets includes Spider [11] and WikiSQL [15].

Voice-driven Querying Systems. Voice-driven querying systems cover a wide range of applications from AI-powered assistants (e.g., Alexa, Siri, and Cortana) and voice-based search engines (e.g., Google, Baidu, Bing) to recent SQL-related systems such as Echo-Query [2]. In the database area, speech-driven SQL-related systems have also been extensively studied. For example, EchoQuery [2] designs a system to translate the voice input into SQL queries. SpeakQL [4] proposes a system to support a subset of SQL grammar which helps the users to manipulate the system with a speech-based interface. TalkSQL [3] implements the speech-based interface through a three-step pipeline: that is, first requires the user to input a voice query, then translates this query into SQL query, and finally execute the query to give the results. CiceroDB-Zero [8] provides a voice-based interface to help the users to explore large data sets. However, differing from these existing studies, SpeechSQLNet [6] is the first that synthesize SQL queries from NLQs expressed in human speech with an E2E neural network.

3 SYSTEM ARCHITECTURE

We are ready to introduce the details of the proposed VoiceQuerySystem, from the following aspects - the task, system overview, the two speech-to-SQL approaches used, and performance analysis.

3.1 The Speech-to-SQL Task

Formally, the training corpus $\mathcal{D}$ is composed of M instances, denoted as $\mathcal{D} = \{\mathbf{d}^1, \cdots, \mathbf{d}^M\}$, where $\mathbf{d}^i$ ($i \in \{1, \cdots, M\}$) represents the i-th instance. We ignore the superscript i for clarity. Each instance $\mathbf{d}$ contains a speech-based NLQ x, its schema s of the corresponding database, and the target SQL query y. The *Speech-to-SQL* problem focuses on translating an unseen speech-based NLQ x' into the desired SQL query y', with restriction from the schema s' of the corresponding database. In our scenario, the database includes a collection T_x of tables, a collection C_x of columns for each table in T_x. The schema greatly affects the desired SQL query, even for the same NLQ. To promote the further development of this task, we construct a benchmark dataset named *SpeechQL* [6] by piggy-backing the widely-used text-to-SQL datasets, where the speech NLQs are generated by a text-to-speech (TTS) [7] component.

3.2 VoiceQuerySystem Overview

Figure 1 gives the system overview and pipeline of the VoiceQuerySystem. The system can be roughly divided into three layers - *Interface*, *Model*, and *Data*. The interface layer enables users to

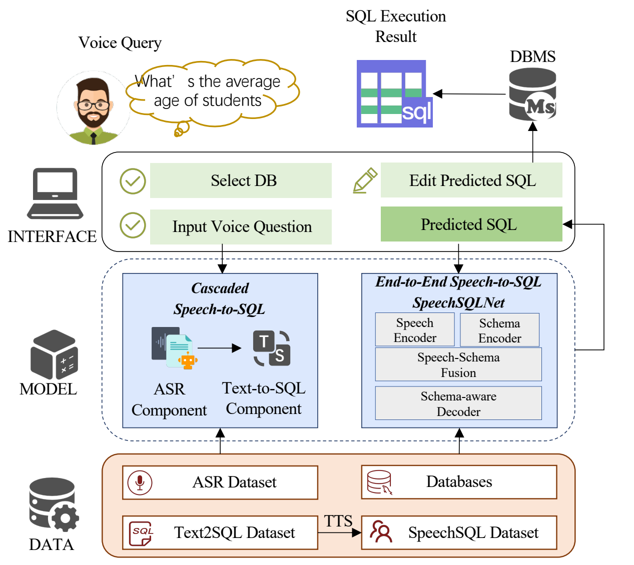


Figure 1: Architecture overview of the VoiceQuerySystem, including three layers - Interface, Model, and Data. The core Speech-to-SQL techniques belongs to two categories: the cascaded one and the E2E one (SpeechSQLNet).

interact with the VoiceQuerySystem, i.e., input the voice query, select the corresponding database, check the predicted SQL queries, and view the SQL execution results. The model layer provides the backbone techniques for Speech-to-SQL translation in VoiceQuerySystem, which can be categorized into two approaches - the cascaded one and the E2E one. The cascaded method first converts the user's voice questions into text by a well-trained ASR module, and then conducts downstream SQL generation via a text-to-SQL model (i.e., IRNet [1]). In contrast, the E2E method is a novel self-designed neural architecture named *SpeechSQLNet* which directly converts the speech signals into SQL queries without the middle medium as text. The data layer covers the datasets (i) Spider and WikiSQL for constructing the text-to-SQL models, (ii) SpeechQL for constructing the speech-to-SQL models, and (iii) Databases where the predicted SQL will be executed with.

3.3 The Cascaded Approach

The cascaded approach is a straightforward solution that combines an ASR component with a well-trained text-to-SQL component. In VoiceQuerySystem, we deploy a self-developed ASR module (i.e., Kaldi "Chain" model and a trigram LM) trained on around 8000-hour labeled data. The text-to-SQL model aims to translate the NLQs into SQL queries, with existing studies such as Seq2SQL [15], EditSQL [13], and IRNet [1]. In our scenario, we directly employ an advanced text-to-SQL model - IRNet due to its good performance. In addition to IRNet, any other text-to-SQL models can also be employed here, and in our experiments, we also employ a vanilla Seq2Seq model similar to [15] as a baseline to conduct performance comparison and we refer this baseline as Seq2SQL.

The cascaded approach is quite straightforward, but it suffers from error-compounding problems (i.e., small ASR errors leads to larger SQL generation errors) which greatly affect its performance.

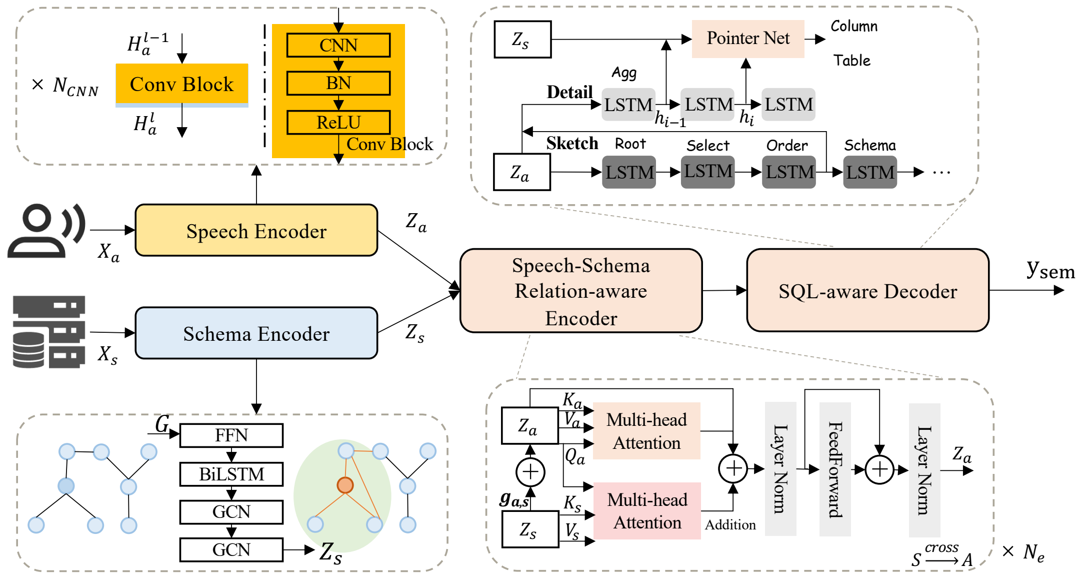


Figure 2: The network structure of our proposed SpeechSQLNet model, which is comprised of a speech-encoder, a schema-encoder, a speech-schema relation-aware encoder, and a SQL-aware decoder.

3.4 The End-to-End Approach

Directly synthesizing SQL queries from speech signals is hard due to the huge modality gap between the two modalities of speech NLQ and SQL queries. Our proposed SpeechSQLNet model consists of four main components, namely a *speech-encoder*, *a schema-encoder*, *a speech-schema relation-aware encoder*, and *a SQL-aware decoder*. The speech-encoder uses a convolutional neural network (CNN)-based architecture to convert the speech signals into hidden representations. Meanwhile, the schema, which greatly affects the desired SQL, is first converted into a graph structure and then encoded into hidden representation by a GNN-based encoder to preserve its structural information. The speech-schema relation-aware encoder aims to identify the references of the tables and columns in the NLQ and then fuses a joint representation for the speech and the schema. The SQL-aware decoder is inspired by the common text-to-SQL architectures [1, 10], which first predicts an intermediate abstract syntax tree (AST) using SemQL [1], and then convert the SemQL into SQL query. The overall structure of the proposed model is illustrated in Figure 2. To further promote the performance, we design two pre-training mechanisms to bridge the modality gap between speech and text representations. The first pre-training task learns to predict if an item from the schema is mentioned by a speech NLQ, while the second one tries to encode then decode the speech and the schema inputs respectively to enforce them to map into the same hidden space. A introduce and comprehensive evaluation of SpeechSQLNet can be found in [6].

3.5 Performance Analysis

We also conduct some comparison among four methods belonging to two approaches, namely **ASR + Seq2SQL**, **ASR + IRNet**, **SpeechSeq2Seq**, and **SpeechSQLNet**. The SpeechSeq2Seq model is a vanilla Seq2Seq network that directly converts speech signals into SQL queries. We adopt the widely-used metric - exact match accuracy [1, 11] as our main indicator. All these models are trained on the same training set from our constructed SpeechQL dataset, and then the accuracies of all these models on the SpeechQL validation and testing datasets are presented in Table 1.

Table 1: Performance of different Speech-to-SQL methods.

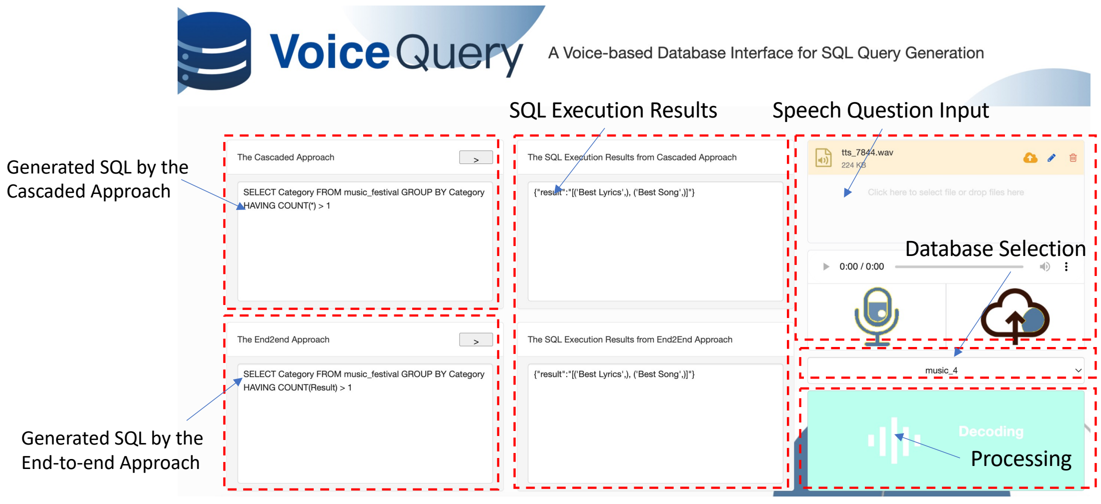
Method	Validation Set	Testing Set
ASR + Seq2SQL	0.0236	0.0195
ASR + IRNet	0.4264	0.4373
SpeechSeq2SQL	0.0700	0.0695
SpeechSQLNet	**0.5355**	**0.5395**

The performance of the cascaded approaches is not competitive compared with the E2E ones, due to the error-compounding problem. The SpeechSQLNet could alleviate this problem since it naturally retains the rich linguistic information in the speech with an advanced network structure, and is optimized globally in an E2E-style to reduce the errors. The dominating performance of the E2E models compared with the cascaded ones validates the necessity of exploring the E2E approach that bypasses the text for the speech-to-SQL problem. What is more, compared with a vanilla Seq2Seq model (i.e., SpeechSeq2Sql), the designed SpeechSQLNet could greatly improve the performance, proving the effectiveness of the designed network structure and pre-training mechanisms.

4 DEMONSTRATION OVERVIEW

This demonstration aims to provide an interactive system for users to learn the working mechanisms of the speech-to-SQL techniques introduced above. Figure 3 gives a screenshot of the user interface of the VoiceQuerySystem, which mainly includes the following three functionalities.

Speech Question Input: Currently VoiceQuerySystem supports audio NLQs input using pre-recorded audio files, and next we will include real-time NLQs recorded by the microphone provided in the interface.


VoiceQuery
A Voice-based Database Interface for SQL Query Generation
SQL Execution Results
Speech Question Input
Generated SQL by the Cascaded Approach
The Cascaded Approach
>
The SQL Execution Results from Cascaded Approach
tts_7844.wav
224 KB
Click here to select file or drop files here
Database Selection
0:00 / 0:00
The End2end Approach
>
The SQL Execution Results from End2End Approach
music_4
Decoding
Processing
Generated SQL by the End-to-end Approach

```
SELECT Category FROM music_festival GROUP BY Category
HAVING COUNT(*) > 1
```

```
{"result":"[('Best Lyrics',), ('Best Song',)]"}
```

```
SELECT Category FROM music_festival GROUP BY Category
HAVING COUNT(Result) > 1
```

```
{"result":"[('Best Lyrics',), ('Best Song',)]"}
```

Figure 3: The user interface of our proposed VoiceQuerySystem.

Database Selection: Since the desired SQL is greatly affected by the schema, the system requires the user to select the corresponding database that the user wants to execute the target SQL query.

Predicted Query: After the speech NLQ and the database schema are fixed, the user need to click the "Decoding" button in the interface. Then the VoiceQuerySystem will generate the desired SQL query. The result in the top left area is from the cascaded approach, while the one from the lower left area is from the E2E (i.e., SpeechSQLNet) one. The system also supports the user to modify the generated SQL query, in case there are some errors.

Execution Result Display: If the SQL query and the database are correctly set, a further click on the "▹" button will trigger the execution of the query on the selected database, and the querying result will be displayed on the webpage.

Case Study: We also list a case in Table 2 to give vivid illustration. The results given by the ASR engine, the cascaded approach (i.e., ASR + IRNet), and the E2E approach (i.e., SpeechSQLNet) are displayed. We can see that the E2E approach correctly predicts the desired SQL query but the cascaded one fails for this case.

Table 2: A SQL Example Returned by VoiceQuerySystem

Input	Speech NLQ	return the number of music festivals of each category
	Selected DB & Schema	music_4: artist(Artist_ID, Artist, Age, Famous_Title, Famous_Release_date); volume(Volume_ID, Volume_Issue, Issue_Date, Weeks_on_Top, Song, Artist_ID); music_festival(ID, Music_Festival, Date_of_ceremony, Category, Volume, Result)
Expected Output	Target SQL	select category, count(*) from music_festival group by category
Output	ASR Result	return the number of music festivals of each category
	Cascaded Approach	select count(music_festival) from music_festival
	E2E Approach	select category, count(*) from music_festival group by category

5 CONCLUSION

In this demonstration, we show a novel VoiceQuerySystem to achieve voice-based database querying using common NLQs. Our contributions can be summarized as (i) we propose a new task named *Speech-to-SQL* with a constructed dataset *SpeechQL* to promote this field; (ii) we explore two approaches (i.e., the cascaded one and the E2E one) to validate the rationale of the proposed speech-to-SQL task; (iii) we design the first E2E-style neural network - *SpeechSQLNet* with an advanced network structure that achieves direct speech-to-SQL conversion; (iv) we develop the VoiceQuerySystem to demonstrate SQL translation from voice-based NLQs.

Acknowledgements. The research of Yuanfeng Song and Raymond Chi-Wing Wong is supported by PRP/026/21FX.

REFERENCES

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

[2] Gabriel Lyons, Vinh Tran, Carsten Binnig, Ugur Cetintemel, and Tim Kraska. 2016. Making the case for query-by-voice with echoquery. In *SIGMOD*.

[3] George Obaido, Abejide Ade-Ibijola, and Hima Vadapalli. 2021. TalkSQL: A Tool for the Synthesis of SQL Queries from Verbal Specifications. In *IMITEC*.

[4] Vraj Shah, Side Li, Kevin Yang, Arun Kumar, and Lawrence Saul. 2019. Demonstration of SpeakQL: Speech-driven Multimodal Querying of Structured Data. In *SIGMOD*.

[5] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer. 2015. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics* (2015).

[6] Yuanfeng Song, Raymond Chi-Wing Wong, Xuefang Zhao, and Di Jiang. 2022. Speech-to-SQL: Towards Speech-driven SQL Query Generation From Natural Language Question. *arXiv*.

[7] Paul Taylor. 2009. *Text-to-speech synthesis*. Cambridge university press.

[8] Immanuel Trummer. 2020. Demonstrating the voice-based exploration of large data sets with CiceroDB-zero. *VLDB* (2020).

[9] Prasetya Utama, Nathaniel Weir, Carsten Binnig, and Ugur Cetintemel. 2017. Voice-based data exploration: Chatting with your database. In *SCAI*.

[10] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *ACL*.

[11] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *EMNLP*.

[12] Gideon Zenz, Xuan Zhou, Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. 2009. From keywords to semantic queries—Incremental query construction on the Semantic Web. *Journal of Web Semantics* (2009).

[13] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. In *EMNLP-IJCNLP*.

[14] Weiguo Zheng, Hong Cheng, Lei Zou, Jeffrey Xu Yu, and Kangfei Zhao. 2017. Natural language question/answering: Let users talk with the knowledge graph. In *CIKM*.

[15] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv*.