The VLDB Journal
https://doi.org/10.1007/s00778-024-00837-0

SPECIAL ISSUE PAPER

Speech-to-SQL: toward speech-driven SQL query generation from natural language question

Yuanfeng Song[1,2] · Raymond Chi-Wing Wong[1] · Xuefang Zhao[2]

Received: 28 January 2023 / Revised: 2 December 2023 / Accepted: 28 December 2023

Abstract
Speech-based inputs have been gaining significant momentum with the popularity of smartphones and tablets in our daily lives, since voice is the most popular and efficient way for human–computer interaction. This paper works toward designing more effective speech-based interfaces to query the structured data in relational databases. We first identify a new task named *Speech-to-SQL*, which aims to understand the information conveyed by human speech and directly translate it into structured query language (SQL) statements. A naive solution to this problem can work in a cascaded manner, that is, an automatic speech recognition component followed by a text-to-SQL component. However, it requires a high-quality ASR system and also suffers from the error compounding problem between the two components, resulting in limited performance. To handle these challenges, we propose a novel end-to-end neural architecture named *SpeechSQLNet* to directly translate human speech into SQL queries without an external ASR step. SpeechSQLNet has the advantage of making full use of the rich linguistic information presented in speech. To the best of our knowledge, this is the first attempt to directly synthesize SQL based on common natural language questions in spoken form, rather than a natural language-based version of SQL. To validate the effectiveness of the proposed problem and model, we further construct a dataset named *SpeechQL*, by piggybacking the widely used text-to-SQL datasets. Extensive experimental evaluations on this dataset show that SpeechSQLNet can directly synthesize high-quality SQL queries from human speech, outperforming various competitive counterparts as well as the cascaded methods in terms of exact match accuracies. We expect speech-to-SQL would inspire more research on more effective and efficient human–machine interfaces to lower the barrier of using relational databases.

Keywords Speech-to-SQL · SQL query generation · Speech-driven querying system · AI/NLP/Speech for database

1 Introduction

Nowadays, the vast majority of data in our lives is stored in relational databases, and an essential tool for accessing this data is through structured query language (SQL) commands [33]. However, SQL has a rather complex grammar and long learning curve, and the difficulty of mastering SQL blocks many non-technical users to use SQL. To facilitate these users to perform data queries, automatically generating SQL queries from natural language questions (NLQs), or text-to-SQL, has been extensively studied in natural language processing (NLP) and database communities recently [3, 25, 34, 43, 44, 53, 64, 100]. For example, Gkini *et. al* published a paper [25] in SIGMOD'21 which systematically summarizes popular text-to-SQL methods and natural language interfaces (NILs) for databases. Affolter *et. al* [3] conducted a comprehensive survey in VLDBJ about the progress in NLIs for databases, especially about recent neural-based approaches.

Compared with the text-based input, it is widely believed that the voice-based interface is a much easier and efficient way for human-computer interaction. According to the user study in [65], the voice-based interface could allow users to compose SQL queries considerably faster by up to 6.7x com-

A VoiceQuerySystem using the techniques described in this draft was published in SIGMOD'22 [70].

✉ Xuefang Zhao
zhaoxf17@163.com

Yuanfeng Song
songyf@cse.ust.hk

Raymond Chi-Wing Wong
raywong@cse.ust.hk
http://www.cse.ust.hk/raywong/

[1] The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China

[2] AI Group, WeBank Co., Ltd, Shenzhen, China

Published online: 16 February 2024

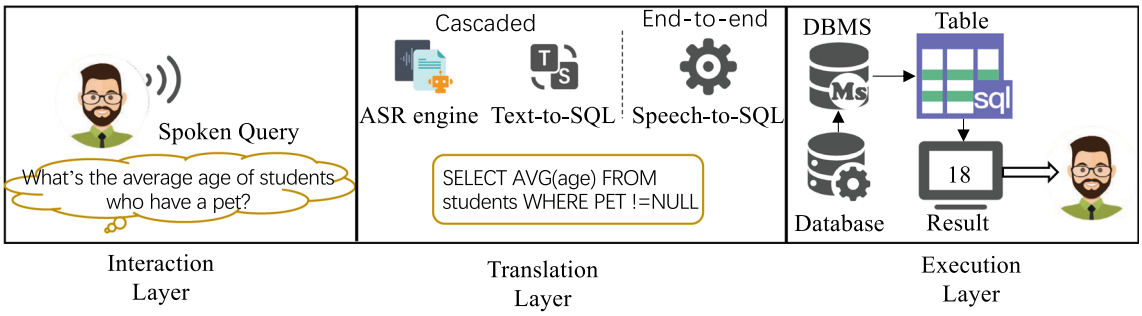


Fig. 1 Speech-driven database querying using Speech-to-SQL; two kinds of underlying techniques, the cascaded approach and the end-to-end approach, are explored in this study. Text is shown only for illustrating the speech content, which is not available in real scenarios

pared to typing on a text-based interface in a tablet device. Meanwhile, with the popularity of smart mobilephones and tablets in our daily lives, dozens of voice-based applications have emerged in the market in the past decades, such as voice-based search engines [4], voice-based AI assistants and chatbots (e.g., Siri, Xiaoice, Google Home and Cortana) [58] and voice-based databases [65].

In parallel, studies already exist for building voice-based interfaces for database systems in the research community [65, 66, 77, 78]. For example, *SpeakQL* implements a voice-based query interface for structured data that enables users to input SQL with speech. Utama et al. [78] designed a system named *EchoQuery*, which also supports users to query the database with voice. However, these studies usually restrict the voice query to be an NL-based version of SQL or its variants with a limited subset of SQL grammar, and thus, they still require the users to have professional background and skill in SQL language. For example, SpeakQL requires the query to be an exact SQL statement such as "`Select Salary From Employees Where Name Equals John`", and EchoQuery requires that the basic query request should be in the form like "`What is the {Aggregation}{Columns(s)} of {Table(s)}?`". As such, none of them achieves translating common flexible speech-based NLQs (i.e., questions without being restricted by any template) into SQL queries, a harder yet more valuable task widely believed to be a more efficient and straightforward way of human–machine interaction that lowers the barrier of using relational databases.

In this paper, we work toward designing more effective voice-based interfaces that attempt to handle common NLQs to manipulate the structured data stored in the relational database. Toward this goal, we first present a new task named `speech-to-SQL`, which aims to understand the information conveyed by human speech and translate it into the corresponding SQL query, as shown in Fig. 1. A naive solution to this task can work in a *cascaded* fashion, namely first converting speech signals into their corresponding transcripts with an automatic speech recognition (ASR) component, and then conducting downstream SQL generation by a text-to-SQL component. While this approach can alleviate the problem to a certain extent, it suffers from the *error compounding problem*. That is, the ASR module produces myriad forms of errors in the recognized transcripts which brings a big technical challenge for the subsequent text-to-SQL conversion. In our analysis, we found that existing text-to-SQL models are not robust to these ASR errors. For example, according to our experiment, a 33% ASR error rate would cause more than 36% accuracy reduction for the downstream text-to-SQL conversion.

Motivated by the above-mentioned issues, we aim to explore the *end-to-end neural approach* that does not require an external ASR step for speech-to-SQL in this work. The benefits of such an end-to-end neural model for speech-to-SQL are as follows, (i) *Potentially good performance*: the end-to-end approach could potentially alleviate the error compounding problem since the whole network enjoys a *single* objective function which is the same as the SQL generation objective, while each component of the cascaded approach employs a different optimization objective; (ii) *Less complexity and efforts*: the end-to-end approach does not require to construct *separate* linguistic components and assemble them together, which frees the engineers from the burden of constructing and maintaining of the high-quality ASR and the text-to-SQL components; (iii) *Faster speed*: since the end-to-end models are usually more compact than cascaded models, it could process the query much faster, which is important since this speech-driven application is usually deployed in a mobile environment with restrict resources.

To validate the rationale of this problem and the following possible end-to-end models, we first construct a benchmark dataset named `SpeechQL`, by piggybacking the public text-to-SQL datasets - WikiSQL [103] and Spider [96]. Then, we design an advanced neural architecture, namely `SpeechSQLNet`, to directly explore the semantics presented in the speech and synthesize the corresponding SQL queries. In particular, SpeechSQLNet seamlessly inte-

grates speech encoder, graphical neural network (GNN) and transformer as its backbones to conduct speech parsing for unlabeled speech data without an ASR step as a premise, and the whole network is optimized in an end-to-end fashion. There are two challenges. The first challenge is the huge modality gap (or representation difference) between speech and text. The second challenge is the schema-linking problem (identifying the references of tables and columns in the speech queries). As such, we design two novel pre-training mechanisms, *speech–sentence* and *speech–item* pre-training, to address these two challenges, respectively. The speech–sentence pre-training is based on an auto-encoder to force the two modalities from speech and text to map into the *same* hidden space. The speech–item pre-training aims to identify the references of tables and columns in the speech queries, which could meanwhile also reduce the modality gap. Experimental evaluations on this proposed dataset show that SpeechSQLNet could synthesize high-quality SQL queries directly from speech. Asides from the experimental evaluations, the techniques describes in this paper are further validated by VoiceQuerySystem [70], a voice-based database querying system that enables users to conduct data operations with speech-based NLQs.

While artificial intelligence for the database (*AI for DB*) has become a hot direction and drawn significant attention from the database community recently [46, 105], a few studies have been conducted on advanced natural language processing (NLP), especially speech technologies, for database topics. Speech-to-SQL will become a popular task toward more effective and intelligible speech-driven human–machine interfaces for relational databases. We hope this work will also inspire more following studies in *NLP/Speech for DB* direction to promote the development of the database area.

In a nutshell, the contributions of the work can be summarized as follows.

- We propose a kind of new query interface with its corresponding task, namely speech-to-SQL, that lowers the barriers of using SQL and relational databases. To promote further research in this task, we also construct a benchmark dataset called SpeechQL. This new task and benchmark dataset would promote the development of speech-driven interfaces for improving the usability of traditional databases (Sect. 3).
- We explore two approaches, the cascaded one and the end-to-end one (i.e., SpeechSQLNet), to synthesize the SQL queries from speech. Compared with the cascaded one, SpeechSQLNet has the advantage of reducing the error compounding problem by optimizing in an end-to-end style, achieving a better performance compared with the cascaded methods. To the best of our knowledge, we are the first in the literature to validate that it is possible to synthesize SQL from an common human question in speech and bypasses text. This would open a new research direction of novel end-to-end neural architectures toward SQL generation from speech NLQs (Sect. 4 & Sect. 5).
- To bridge the challenging modality gap between the two modalities of speech and text, we propose a novel speech–sentence pre-training mechanism that employs an autoencoder-based framework to map the speech and text into the same hidden space. Furthermore, to handle the schema-linking problem, we propose another speech–item pre-training mechanism to empower the modal's capability in identifying the refereed items in the speech query. Since these two mechanisms are intended for reducing the modality gap between the two inputs, speech NLQ and schema, they would be potentially applied to any following end-to-end neural networks for speech-to-SQL (Sect. 6).
- We conducted extensive experiments on this proposed benchmark dataset, compared with several strong baselines [28, 74, 79, 100, 103]. Experimental results show that SpeechSQLNet can significantly outperform not only other end-to-end baselines [74, 79] but also improve the cascaded models, such as an advanced model—IRNet [28], by up to 10.22% in terms of query match accuracy (Sect. 7).

The rest of this paper is organized as follows. We first introduce some preliminaries in Sect. 2. Then we give the detailed problem setup in Sect. 3, following by the introduction of the naive cascaded approach in Sect. 4. The proposed SpeechSQLNet, including the model structure as well as its components, is illustrated in Sect. 5. The specially designed training mechanism is discussed in Sect. 6. Experimental evaluations are then presented in Sect. 7, followed by a comprehensive review of the related work in Sect. 8. Finally, we conclude the work with discussion especially on this large language model (LLM) era in Sect. 9.

2 Preliminaries

We introduce some preliminary knowledge about sequence-to-sequence structure (Seq2Seq) and transformer structure, which helps the understanding of the concept used.

Sequence-to-Sequence Structure. Seq2Seq [17] refers to a series of neural architectures that map a given sequence of elements (e.g., words and speech signals) into another sequence. Many tasks such as machine translation and ASR enjoy a common Seq2Seq structure and can be solved under the Seq2Seq framework. Take English-to-Chinese machine translation as an example. The input sequence of a Seq2Seq network designed for this task is the source language (i.e., English) and the output sequence is the target language (i.e.,

Chinese). The network usually consists of an encoder and a decoder, where the former takes the input sequence and converts it into some hidden representations, and the latter maps these hidden representations into the target sequence. The common choice of the encoder and decoder can be a recurrent neural network (RNN) [50] or long short-term memory (LSTM) [27]. The speech-to-SQL problem also enjoys a Seq2Seq structure and can also be considered to be a special case of Seq2Seq conversion. Thus, we also employed several basic Seq2Seq network architectures [74, 103] as baselines for performance comparison in this work.

Transformer-based Structure. As a special case of the Seq2Seq structure [17], Ashish et al. [79] proposed a neural network named *Transformer* that shows promising performance in various NLP tasks such as machine translation and dialogue systems. A novel architecture named *multi-head self-attention* was designed, which is formally defined as

$$\texttt{MultiHead}(Q, K, V) = [h_1; h_2; \cdots; h_n]W_0, \tag{1}$$

where [;] represents the concatenation operation, Q is the query, K is the key, V is the value, and $W_0 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ is learnable parameters. n is the number of heads, d_{model} is dimensionality of the model, and each h_i is obtained from an attention module, defined as

$$h_i = \texttt{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{2}$$

$$\texttt{Attention}(Q_i, K_i, V_i) = \texttt{softmax}(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i, \tag{3}$$

where $d_k = \lceil \frac{d_{\text{model}}}{n} \rceil$, d_k is the dimensionality of queries Q_i, $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ are the parameters.

The transformer consists of multiple stacked encoder and decoder layers. The encoder contains several layers consisting of a self-attention module followed by a position-wise feed-forward layer, defined as

$$\texttt{FFN}(X) = \texttt{max}(0, XW_1 + b_1)W_2 + b_2, \tag{4}$$

where $\texttt{FFN}(\cdot)$ refers to a feed-forward network, $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$ and $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$ are the weight matrices, $b_1 \in \mathbb{R}^{d_{ff}}$ and $b_2 \in \mathbb{R}^{d_{model}}$ are the corresponding bias, d_{ff} is the dimensionality of the inner-layer, and $X \in \mathbb{R}^{d_{model} \times d_{model}}$ is the input matrix. To further capture the sequential information, a positional encoding (PE) [79] mechanism is further employed, which mathematically defined as

$$\texttt{PE}_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}}), \tag{5}$$

$$\texttt{PE}_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}}), \tag{6}$$

where pos is the token position in a sequence and i is the dimension index.

Table 1 The statistics of the proposed SpeechQL Dataset

Statistic	Train	Validation	Test	Total
No. of Instances	31431	1100	2200	34731
No. of Databases	13199	1058	2023	13601
Length of Speech	37.43 h	1.33 h	2.61 h	41.37 h
Avg NLQ Length	11.85	11.97	11.83	11.85
Avg SQL Length	9.1	9.05	9.03	9.09
Vocab Size	–	–	–	15908

3 Problem setup

We are ready to give the formal definition of the speech-to-SQL problem and the benchmark dataset constructed to evaluate the rationale of the proposed problem and possible models.

3.1 Speech-to-SQL problem

Suppose that we have a speech corpus $\mathcal{D}$ 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. The superscript i is ignored for simplicity in the following discussion. Each training instance $\mathbf{d}$ is in the format of $\{x, y, S\}$, where x is a speech recording expressing the NLQ, y is its translation in SQL query, and S is the schema of the corresponding database on which y will be executed. The *speech-to-SQL* problem aims to learn a model which can translate an unseen question-schema pair $\{x', S'\}$ to its corresponding SQL query y'. Specifically, the schema S includes the collection T_x of N tables where $T_x = \{t_i\}_{i=1}^N$, the collection $C_{x,i}$ of L_i columns where $C_{x,i} = \{c_{i,j}\}_{j=1}^{L_i}$ for each table $t_i \in T_x$ and L_i is the number of columns in table t_i for $i \in [1, N]$, and a set F_x of foreign key–primary key column pairs, where each pair $(c_f, c_p) \in F_x$ refers to a relation between a foreign key c_f and a primary-key c_p where the keys are from two different tables. We use C_x to include all the columns and $C_x = \{C_{x,i}\}_{i=1}^N$.

Figure 2 gives two example for a vivid illustration. In the first example, the user expresses their data requirement by orally asking a question "`Please find the average age of students who do not have a pet?`" to the speech-based interface of the databases. Given the schema of the database as "`student (sid, sname, age), pet (pid, sid, pname)`" and the above speech (without the corresponding transcript), the speech-to-text task aims to directly synthesize the desired SQL query "`SELECT AVG(age) FROM student WHERE sid`

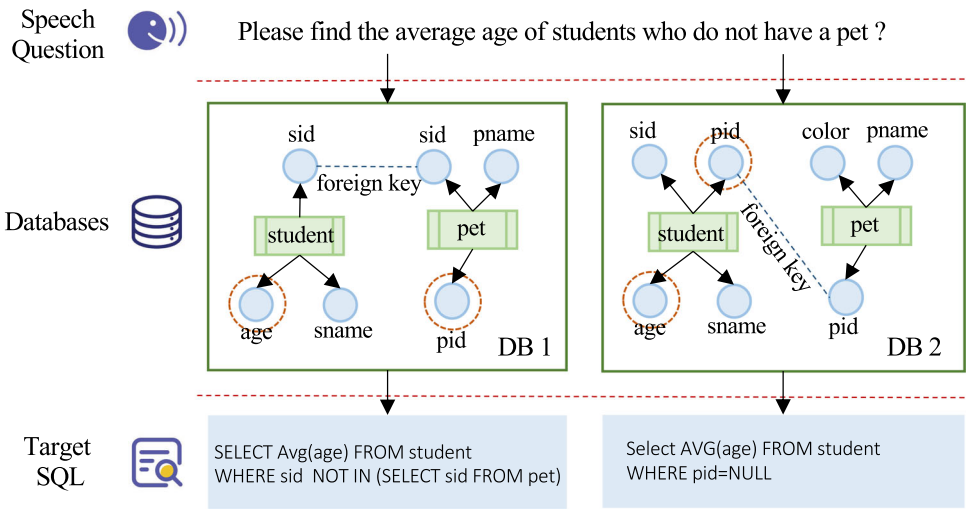


Fig. 2 Two examples of the speech-to-SQL problem, showing that different databases (schemas) can result in different desired SQL queries, even for the same speech question. Thus, schema is also essential as a part of the input for the potential speech-to-SQL models, which is the same for the existing text-to-SQL problem. However, in practical systems like VoiceQuerySystem [70], from the perspective of no-technical background users, the schemas are usually preloaded by the systems, rather than defined/required from the end user side

`NOT IN (SELECT sid FROM pet)`". Finally, the DBMS could execute the generated SQL and display the final results to the user. For the second example, the same NLQ is queried by the user on another database with schema "`student (sid, pid, sname, age), pet (pid, color, pname)`", and the desired SQL would be "`SELECT AVG (age) FROM student WHERE pid = NULL`" correspondingly, to guarantee the correctness of the final results. From the perspective of a potential model, the schema is an essential part of the input in the speech-to-SQL problem, which is consistent with the setting of existing text-to-SQL problem and studies [11, 28, 42, 94, 96]. However, this does not mean that we require the non-technical background users to give the schema when we deploy the potential speech-to-SQL models in a speech-driven querying system. Actually, in systems like VoiceQuerySystem [70], the schema is usually preloaded by the systems, rather than required from the end user side.

3.2 Speech-to-SQL dataset

Since speech-to-SQL is a new task, there are currently no existing datasets in the literature that are suitable for evaluating its performance. Following the same practice of using text-to-speech (TTS) technology [22] to generate the spoken captions in other speech-driven applications like the speech-to-image task [47, 83, 84], the spoken implicit discourse relation recognition task [51] and the speech translation task [87], we create a new dataset based on public text-to-SQL datasets, namely WikiSQL [103] and Spider [96], by converting the textual statement into speech using a TTS module.

There are several text-to-SQL datasets in the text-to-SQL field [72, 96, 103]. Spider dataset [96] is a small-scale dataset for cross-domain text-to-SQL evaluation annotated by human experts. It contains 8,625 training instances, 1,034 validation instances and 2,147 test instances, where one instance corresponds to $\{x, y, S\}$ containing x as the text-based NLQ, y as the SQL and S as the schema of the corresponding database. One of the reasons that Spider is popular in the text-to-SQL field is that it contains complicated queries from diverse databases, ranging from restaurants and scholars, to academics. The WikiSQL dataset [103] is another popular dataset in the text-to-SQL task. Compared with the Spider, WikiSQL only covers simple queries in form of `aggregate-where-select` structure, but the size of the WikiSQL dataset is much larger.

In our experiments, we mainly created our dataset based on Spider [96] and WikiSQL [103]. Since speech-based applications usually require much larger training datasets to obtain well-trained neural models compared with text-based neural applications, we try to obtain more training instances based on the Spider dataset with a method similar to [94]. In particular, we first extract some templates (including both SQL queries and NLQs) from the Spider dataset, then we choose some new databases from other sources (i.e., WikiSQL), and finally, we fill the templates with information (i.e., columns) from these new databases to generate new instances. Then, we merge the generated dataset with the original Spider. This step results in a dataset that is much larger compared with the original Spider, which is more suitable for exploring speech-driven methodologies and applications.

For the TTS module, we apply FastSpeech 2 [62] to generate the Mel spectrogram from text. Then, we synthesize the raw waveform as a speech recording from the Mel spectrogram using MelGAN [38], with a stable and efficient filter bank - pseudo quadrature mirror filter bank (Pseudo-QMF) [52]. Finally, we obtain a new dataset which we named *SpeechQL*. The detailed statistics of this dataset can be found in Table 1. Then, we randomly split the dataset into training, validation and testing sets. Following the common practice in evaluating the quality of the generated speech records [88],

we also randomly sample 50 records to manually check the intelligibility rate and it achieves a very high accuracy (near 100%). It is noted that using TTS to generate the speech recording is a commonly used and widely accepted approach to create large-scale datasets in evaluating speech-driven tasks due to its high intelligibility rate (e.g., at least 98% for commercialized cloud speech services [88]) and mature of existing TTS technologies, as used in [47, 51, 83, 84, 87].

4 A naive baseline: the cascaded approach

A naive solution to the speech-to-SQL task can work in a cascaded fashion, namely first converting speech signals into their corresponding transcripts with an ASR system and then conducting downstream text-to-SQL conversion. In this section, we mainly discuss this cascaded solution from its main components: ASR in Sect. 4.1 and text-to-SQL in Sect. 4.2.

4.1 Automatic speech recognition

Mathematically, the hybrid ASR component [60] transcribes the a user's voice NLQ into a textual output, which can be expressed as follows:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} (\log P_{LM}(\mathbf{w}) + \lambda \log P_{AM}(\mathbf{a}|\mathbf{w})), \tag{7}$$

where λ is a trade-off parameter. P_{AM} is an acoustic model (AM), evaluating how sounds combine to produce word sequence, and P_{LM} is a language model (LM), picking the word sequence that has the largest probability in human language perspective [68, 69].

The ASR component plays a critical role in the cascaded approach for speech-to-SQL, but constructing a SQL domain ASR is hard since the NLQs contain many domain-specific words, especially in the database context. These words can easily be misrecognized. For example, in a real case from SpeechQL dataset, the original NLQ is "`Which Video has a PSIP Short Name of rt?`". It is as easily recognizable as "`What video has a safe short name of artie`", even by the widely used ASR engines provided by Google,[1] Microsoft AI Cloud,[2] or Baidu Cloud.[3] The reason for this phenomenon is obvious. That is, the trained data used to construct their ASR model usually lack SQL domain-specific data. Hence, it would be better and flexible if we could construct an ASR model by ourselves with a SQL domain dataset.

To alleviate the above-mentioned problem, we turn to a two-step (i.e., pre-training and fine-tuning) approach to build a reliable and accurate ASR model dedicated to the SQL domain. Specifically, the ASR model is pre-trained with various English datasets from general domains including Ted Talks [29, 63] and LibriSpeech [57], then the pre-trained ASR model is further fine-tuned with the small-scale dataset in SQL domain to adapt to the database domain. For the ASR system, we utilize the Kaldi "Chain" model [60] as the AM component and employ a trigram LM [71] as the LM component. The DNN component of the "Chain" model is a Time Delay Neural Network (TDNN) [82]. To further improve the performance, an advanced rescoring mechanism named L2RS [68, 69], integrated with BERT [20], is also adopted in the N-best rescoring step.

We employ the widely used metric, word error rate (WER), as the main indicator for the performance of the constructed ASR model. WER is formally defined as

$$\text{WER} = \frac{S + D + I}{T}, \tag{8}$$

where S, D and I represent the number of word substitutions, deletions and insertions, respectively, and T is the total number of words in the ground-truth transcript. From the definition, we can conclude that WER reflects the quality of the ASR decoding result, and the lower the value, the better the decoding result. Overall, we achieve a WER of 34.451%, a relatively low error rate in this specific SQL domain. It is worth mentioning that this well-tuned ASR model makes the cascaded approaches very strong baselines since parallel studies in the speech-to-image task such as [84] employ the ASR model with a far high WER (around 50%) as their baseline.

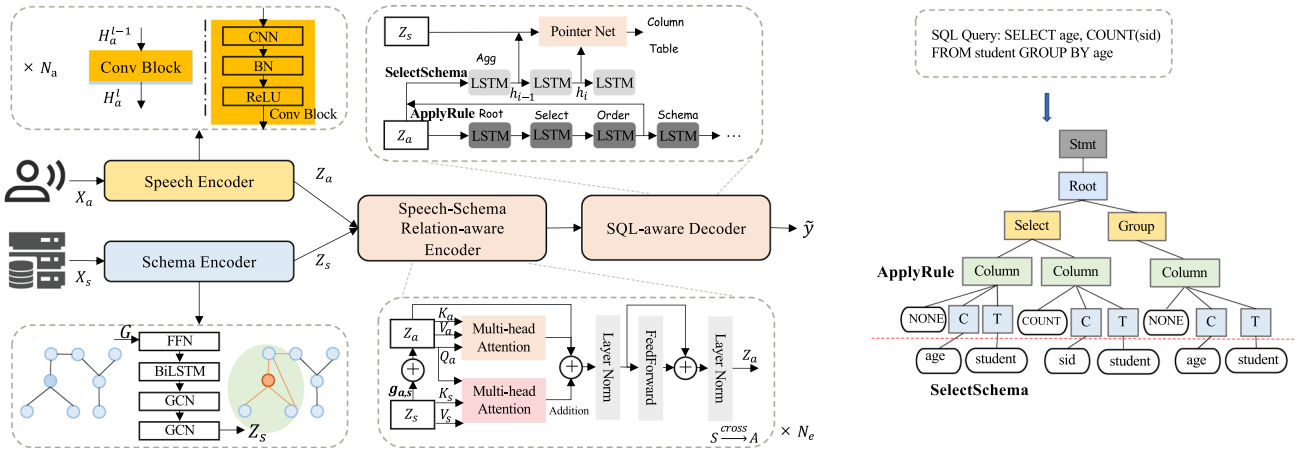
4.2 Text-to-SQL conversion

The text-to-SQL model aims to convert the NLQs into SQL queries, with previous studies such as Seq2SQL [103], SeqNet [89], EditSQL [100], IRNET [28], and so on. In our implementation, we directly construct text-to-SQL models with existing techniques on our datasets. The existing text-to-SQL model is usually trained based on a clean dataset. That is, the NLQs contain no errors. However, in our scenario, the NLQ is recognized by an ASR module, which produces myriad forms of errors in recognized transcriptions that further introduce the *error compounding* problem, a technical challenge for the subsequent text-to-SQL conversion. To improve the performance, we then explore the possibility of end-to-end speech-to-SQL conversion in Sect. 5.

[1] https://cloud.google.com/speech-to-text

[2] https://azure.microsoft.com/cognitive-services

[3] https://ai.baidu.com/tech/speech/asr



(a) Network Structure

(b) An example of converting SQL query into Abstract Structure Tree

Fig. 3 (a) The network structure of the proposed SpeechSQLNet model, including a speech encoder, a schema encoder, and a SQL-aware decoder. (b) An example of representing SQL query with abstract structure tree (AST). The output of the SpeechSQLNet model is an AST $\tilde{y}$ in SemQL [28], which is further converted into the desired SQL query y

5 Our proposed end-to-end model: speechSQLNet

Directly synthesizing SQL queries from speech signals is hard due to the huge modality gap between the two modalities of speech signals (i.e., audio modality) and SQL (i.e., programming language modality). To deal with this problem, SpeechSQLNet first uses a speech encoder to convert the speech signals into hidden representations. In the meantime, the schema, which greatly affects the desired SQL, is also converted into hidden features by a GNN-based encoder to preserve its structural information. Finally, the speech embedding, together with the schema features, is fused to synthesize the corresponding SQL query with semantic consistency. The overall structure of the proposed model is illustrated in Fig. 3. In this section, we detail the proposed SpeechSQLNet model from its four main components: *Speech Encoder* (Sect. 5.1), *Schema Encoder* (Sect. 5.2), *Speech–Schema Relationship-aware Encoder* (Sect. 5.3) and *SQL-aware Decoder* (Sect. 5.4).

5.1 Speech encoder

5.1.1 Raw feature extraction from speech signals

Theoretically, it should be possible to synthesize SQL directly from the digitized waveform, due to the strong modeling capability of existing DNNs. However, human speech signals are highly variable, and the objective of conducting feature extraction is to reduce the *variabilities*. Specifically, the typical variability that we would like to eliminate includes the effect of the periodicity or pitch, the amplitude of excitation signal and fundamental frequency [14]. Inspired by the common practice in ASR [60, 68, 91], we extracted 96 log-scaled Mel-band energies from speech x using sliding windows of 1024 samples ($\approx 46ms$), with 512 overlaps and the Hamming windowing function, to serve as inputs $X_a \in \mathcal{R}^{l_a \times 96}$ for speech encoder, where l_a is the length of the speech hidden space.

5.1.2 Speech embedding from CNN-based architecture

After extracting the feature embeddings from the digitized waveform, we design a speech encoder that employs a convolutional neural network (CNN)-based structure to preserve the continuity of speech. Specifically, the network is composed of a series of convolutional blocks, each of which is stacked by a CNN module [37] and a batch normalization (`BatchNorm`) [32] module. The process of each block can be denoted as:

$$H_a^{(l)} = f(\texttt{BatchNorm}^{(l)}(\texttt{CNN}^{(l)}(H_a^{(l-1)}))), \tag{9}$$

$$H_a^{(0)} = X_a, \tag{10}$$

where $l \in \{1, \ldots, N_a\}$ is the layer number of stacked blocks, $H_a^{(l)} \in \mathbb{R}^{l_a^{(l)} \times d_a^{(l)}}$ is the hidden representation of the speech features, delivered from the l-th layer and sent to the $(l+1)$-th layer, $l_a^{(l)}$ and $d_a^{(l)}$ are the length and the dimensionality of the l-th layer speech hidden representation, respectively. $f(\cdot)$ is an activation function and in our network, we choose the popular `ReLU` function [26]. Lastly, we obtain a speech embedding $Z_a \in \mathbb{R}^{l_a^{(N_a)} \times d_a^{(N_a)}}$ by the speech encoder as

$$Z_a = H_a^{(N)}, \tag{11}$$

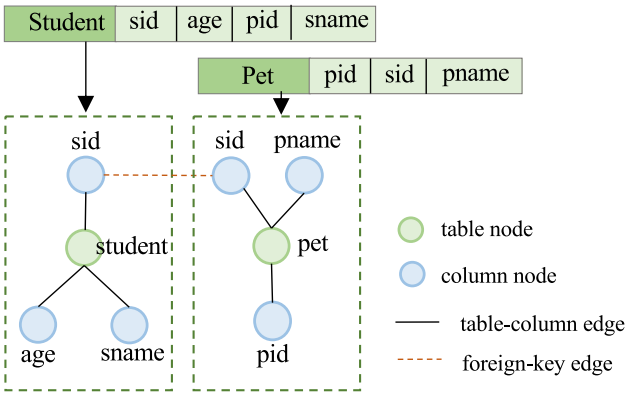


Fig. 4 An example of converting database schema into graph

where N_a is the number of layers in speech stacked blocks.

5.2 Schema encoder

Even for the same NLQ, it has been proven that the schema of the database greatly influences the structure of the desired SQL query [11]. As such, we need to preserve the schema information from both the semantic and the structural perspectives. In SpeechSQLNet, we first convert the database schemas into graphs and then employ the GNN-based architecture as the schema encoder.

5.2.1 Converting database schema into graph

To leverage the structure information, we define an *undirected* graph $G = (V, E)$ to represent the schema. An example with two tables is shown in Fig. 4. Specifically, each node $v_i \in V$ is either a *table node* $t \in T_x$ or a *column node* $c \in C_x$, and each edge $e_{i,j} = (v_i, v_j) \in E$ is designed based on the database relations. There are two types of edges in the graph, and each of them represents a way how columns and tables correlate to one another, including table-column (edge between table $t_i \in T_x$ and its column $c_j \in C_{x,i}$) and foreign key (edge between a key and its foreign key linking two different tables). Since that items with the same name contribute to the feature construction and SQL decoder in the same way, the columns with the same name, even in a different table, are represented by the same node. For each node v, we could obtain its neighborhood $N(v) = \{u \in V | (u, v) \in E\}$. Finally, the graph G is serving as inputs X_s for the schema encoder.

5.2.2 Converting the NL into initial node embedding

To retain the semantic information, instead of randomly initializing the embedding of each node in the graph, we adopt an LSTM-based network to generate the initial embedding. For a node $v_i \in V$ which may contains more than one tokens (e.g., 'Student_ID' consists of two tokens 'Student' and 'ID'), each token in the node is first converted into an embedding vector denoted by embed(v_i), and then a bidirectional LSTM (BiLSTM) is employed to convert the variable-length node tokens' embedding into fixed-length hidden state vectors. The output vectors of the forward and backward LSTM are concatenated together, followed by a feed-forward network (FFN) to construct the node embedding $h_{v_i} \in \mathbb{R}^{d_{model}}$. Then, the initial embedding of the schema graph, denoted as $H_s \in \mathbb{R}^{l_s \times d_{model}}$ which includes l_s nodes, is calculated as follows.

$$h_{v_i} = \texttt{FFN}(\texttt{BiLSTM}(embed(v_i))), \tag{12}$$

$$H_s = [h_{v_1}, \ldots, h_{v_{l_s}}]^T. \tag{13}$$

5.2.3 GCN-based schema encoder

Given the schema in the form of a graph, we would like to capture the overall structure and the detailed connections for each node. As such, we use a GNN-based encoder to embed the schema graph G into a hidden space representation. Its main idea is to update embedding h_v for each node v by aggregating its own features h_v and all of its neighbors' features, each denoted by $h_u \in \mathbb{R}^{d_{model}}$, where $u \in N(v)$, as shown in Eq. (14).

$$h_v = f(\Theta^T \sum_{u \in \{N(v) \cup v\}} \bar{A}_{u,v} h_u), \tag{14}$$

where $f(\cdot)$ is an activation function (ReLU), Θ is a learned parameter and $\bar{A}_{u,v}$ is correlation coefficient between node u and node v computed by the adjacency matrix [36]. In practice, we apply a two-layered GCN to obtain the final schema embedding, namely two-hop neighbors are considered. This is because that a two-layered GCN ensures that a column node embedding (e.g., node 'age' from Fig. 4) can obtain the information from its table (e.g., 'Student' table) and nodes in the same table (e.g., nodes 'sid' and 'sname'). The final schema embedding $Z_s \in \mathbb{R}^{l_s \times d_{model}}$ is obtained by

$$Z_s = \texttt{GCN}(\texttt{GCN}(H_s)). \tag{15}$$

5.3 Speech–schema relation-aware encoder

In traditional text-to-SQL tasks, schema-linking, which identifies references of tables and columns in the NLQs, could greatly improve the accuracy of the models [42]. While it is trivial to identify the mentioned database keywords in an NLQ (e.g., simple keywords matching), it is quite hard to assign the linking information between the speech and database in advance since identifying database keywords from the speech NLQ (without an external ASR component) requires much more effort [5]. As such, we design an

advanced encoder to predict the relation information between speech and database automatically.

5.3.1 Self-learned schema linking

The goal of schema-linking is to recognize the items (table, column, and database values) mentioned in the speech NLQ and then enhance the speech embedding by incorporating the embeddings from these correlated items. This module takes the speech embedding Z_a and the schema embedding Z_s as input and then outputs an *advanced hybrid speech embedding*. Concretely, a feed-forward network is applied to the speech and schema embedding firstly, which aims to map them into a *common* hidden space. To learn a speech representation that considers its related schema items, we further perform an attention mechanism [79] over the speech frames and the schema items. A *semantic similarity score*, denoted as $g_{a,s} \in \mathbb{R}^{l_a \times l_s}$ (l_a is short for $l_a^{(N_a)}$, which denotes the length of speech in the output layer), and $g_{a,s}^{i,j}$ is calculated between the i-th speech frame and the j-th schema nodes to serve as a schema-linking relevance score, and then the weighted sum of the schema embedding denoted by $C_a \in \mathbb{R}^{l_a \times d_{model}}$ is concatenated to the original speech embedding. Mathematically, the whole process is represented as follows.

$$g_{a,s} = \frac{Z_a (Z_s)^T}{||Z_a|| \cdot ||Z_s||}, \tag{16}$$

$$C_a = g_{a,s} Z_s, \tag{17}$$

$$Z_a = Z_a + C_a. \tag{18}$$

5.3.2 Multi-modal transformer-based fusion

To leverage both multi-modal and long-term temporal relationships among different modalities (i.e., audio and text), we learn a joint representation for the speech and the schema by a multi-modal transformer-based encoder. Inspired by the vanilla transformer structure [79], the encoder is composed of a stack of basic blocks, each of which consists of a multi-head attention mechanism, a fully connected feed-forward network and a layer normalization [6]. However, different from the basic transformer structure, we add two attention modules in our multi-modal transformer, the first of which is a self-attention (SA) module designed for learning the relationships inherent in one modality, and the latter of which is a cross-modal attention (CA), which is proposed to fuse the latent relation information among different modalities.

Due to space limitations, we only illustrate the generation process of the improved speech embedding here. We first employ the SA module over the speech embedding to obtain $Y_a^{SA} \in \mathbb{R}^{l_a \times d_{model}}$, then perform the cross-modal attention between the speech and the schema to obtain $Y_a^{CA} \in \mathbb{R}^{l_a \times d_{model}}$. Furthermore, the sum of Y_a^{SA} and Y_a^{CA} is used to represent the *multi-modal speech output* Y_a. Finally, layer normalization and feed-forward network, which has been proved effective in improving the performance in prior work [6], are also incorporated into the encoder. After the above-mentioned steps, we could obtain a new embedding for the speech, denoted as Z_a. Mathematically, the whole process can be represented as:

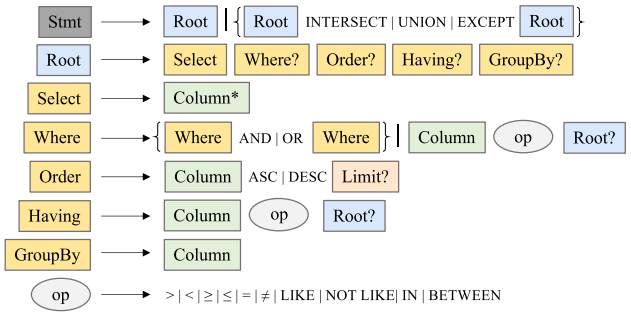


Fig. 5 The grammar of SemQL

$$Y_a^{SA} = \texttt{MultiHead}(Q, K, V), Q = K = V = Z_a, \tag{19}$$

$$Y_a^{CA} = \texttt{MultiHead}(Q, K, V), Q = Z_a, K = V = Z_s, \tag{20}$$

$$Y_a = Y_a^{\text{SA}} + Y_a^{\text{CA}}, \tag{21}$$

$$Y_a = \texttt{LayerNorm}(Y_a + Z_a), \tag{22}$$

$$Z_a = \texttt{LayerNorm}(\texttt{FFN}(Y_a) + Y_a), \tag{23}$$

where $\texttt{MultiHead}(\cdot)$ means the multi-head mechanism mentioned in Sect. 2, $\texttt{LayerNorm}(\cdot)$ is the layer normalization, and $\texttt{FFN}(\cdot)$ refers to the feed-forward network. We repeat this process N_e times to obtain the final updated speech embedding Z_a. Similar to the generation process of the improved speech embedding Z_a (Eq. (19) to (23)), we could generate the improved schema embedding Z_s.

5.4 SQL-aware decoder

Since SQL is a programming language with clear and strict grammar, it would be better to encode this grammar information as a prior to guide the generation process.

Following the practice in text-to-SQL [28], in our implementation, we also choose the grammar of SemQL (described in Fig. 5) to represent each SQL query as an *Abstract Structure Tree* (AST), with an example shown in Fig. 3(b). Our decoder is adapted from a grammar-based structure commonly used in the text-to-SQL task [28], which employs an LSTM architecture to synthesize SemQL AST $\tilde{y}$ by choosing a sequence of *actions*. An action can either be applying a production rule (e.g., "$Root \rightarrow Select + Group$" in Fig. 3(b)) or selecting a schema item (e.g., "$Column \rightarrow COUNT$" and "$Column \rightarrow C \rightarrow age$" in Fig. 3(b)), which corresponds

to a branch selection operation in the AST. Mathematically, the generation process of a SemQL query sequence $\tilde{y}$ can be formalized as:

$$p(\tilde{y}|x, S) = \prod_{i=1}^{T} p(a_i|x, S, a_{<i}), \tag{24}$$

where x is the speech-based NLQ, S is the schema, a_i means an action applied at Step i, $a_{<i}$ are all the previous actions before step i, and T is the number of all the actions to predict $\tilde{y}$. The actions involved in the generation process of Eq. (24) is further categorized into two types: (i) *ApplyRule*: applying a production rule to the current grammar tree till finishing the SQL sketch; (ii) *SelectSchema*: selecting a column or table item from the schema to complete the SQL query.

5.4.1 ApplyRule

The objective of the *ApplyRule* step is to construct a context-free grammar tree of the SQL query [28]. In every step, we select the most probable branch given the previous route and employ an LSTM-based structure to simulate the process. At each prediction step i, the LSTM state is updated based on the previous state $h_{i-1} \in \mathbb{R}^{d_{model}}$, the previous action embedding $a_{i-1} \in \mathbb{R}^{d_{action}}$ (d_{action} is the dimensionality of the action embedding), the previous action type embedding $n_{i-1} \in \mathbb{R}^{d_{type}}$ (d_{type} is the dimensionality of the action type embedding), and previous context representation of LSTM c_{i-1}. Inspired by [48], a Luong attention mechanism is then applied to obtain the intermediate representation u_i based on current state h_i and the final speech embedding Z_a (Eq. (26) and (27)). Finally, given u_i, the probability of selecting a rule is calculated by Eq. (28).

$$h_i = \texttt{LSTM}([a_{i-1}; n_{i-1}; c_{i-1}], h_{i-1}), \tag{25}$$

$$c_i = \texttt{Softmax}(h_i^T W_a Z_a^T) Z_a, \tag{26}$$

$$u_i = \texttt{tanh}([h_i; c_i] W_u + b_u), \tag{27}$$

$$p(\tilde{y}_i = a_i|x, S, a_{<i}) = \texttt{Softmax}(\texttt{tanh}(u_i^T W_p + b_p)), \tag{28}$$

where $W_a \in \mathbb{R}^{d_{model} \times d_{model}}$, $W_u \in \mathbb{R}^{2d_{model} \times d_{model}}$ are the trainable weight matrices, $W_p \in \mathbb{R}^{d_{model} \times n_a}$ (n_a is the number of actions and n_a=46 in our setting), $\texttt{Softmax}(\cdot)$ is the softmax function, and [;] refers to concatenation operation. $b_u \in \mathbb{R}^{d_{model}}$ and $b_p \in \mathbb{R}^{n_a}$ are trainable bias, and the initial state h_0 is obtained by a max-pooling operation of the final speech embedding Z_a.

5.4.2 SelectSchema

To fill in the specific item in the target SQL, another LSTM-based structure is also employed. In this part, we need to predict the operation (e.g., `max`, `min` and `count`) and the item (column or table) involved in the speech given the schema. The schema varies in every case and the desired items are also not fixed, which is quite different from the *ApplyRule* part. As such, we further employ the pointer network [80] to handle this issue. The probability of selecting a schema item a_i is calculated as follows,

$$p(\tilde{y}_i = a_i|x, S, a_{<i}) = \texttt{Softmax}(u_i^T W_s Z_s^T), \tag{29}$$

where $W_s \in \mathbb{R}^{d_{model} \times d_{model}}$ is a trainable weight matrix. In particular, the candidate tables items/nodes can only be the ones connecting the selected column items/nodes.

6 Model training

Even though we have designed several modules to fuse the information from different modalities, there still exists a large gap between the speech modality and the text modality. To further minimize this modality gap, we propose a two-step framework (i.e., pre-training and fine-tuning) to train this network. The network first conducts model pre-training (Sect. 6.1 and Sect. 6.2) to align a common hidden space for each pair of speech and its corresponding transcript sentence and then conducts fine-tuning (Sect. 6.3) based on the datasets of speech–SQL pairs. It should be noticed that the pre-training step is optional, depending on the availability of the transcripts. Furthermore, the dataset used for the pre-training step is not necessarily required in the SQL domain.

6.1 Speech–sentence pre-training

As shown in Fig. 6, we design an autoencoder (AE)-based framework to match speech and text sentence with two different AEs. Semantic representations for different modalities are mapped by a speech autoencoder (SAE) and a text autoencoder (TAE), respectively, and these two representations are forced to be close to each other. We employ the same encoder like the one mentioned in Sect. 5.1 for the SAE, and although the encoder mentioned in Sect. 5.2 is designed for database schema, we could still use its NL embedding part (Sect. 5.2.2) here to construct the TAE. By doing this, the network parameters of TAE and SAE after this pre-training step can be reused in the following fine-tuning phase. The decoders of the SAE and the TAE have reversed modules of their encoders according to the design rules of AEs. The loss $\mathcal{L}$ of the network is composed of three parts: a reconstruction loss $\mathcal{L}_a$ of speech input, a reconstruction loss $\mathcal{L}_s$ of transcript input, and a contrastive loss $\mathcal{L}_p$ between the speech and text pairs, which can be formulated as follows.

$$\mathcal{L}(\tilde{X}_a, \tilde{X}_s) = \mathcal{L}_a + \mathcal{L}_s + \mathcal{L}_p, \tag{30}$$

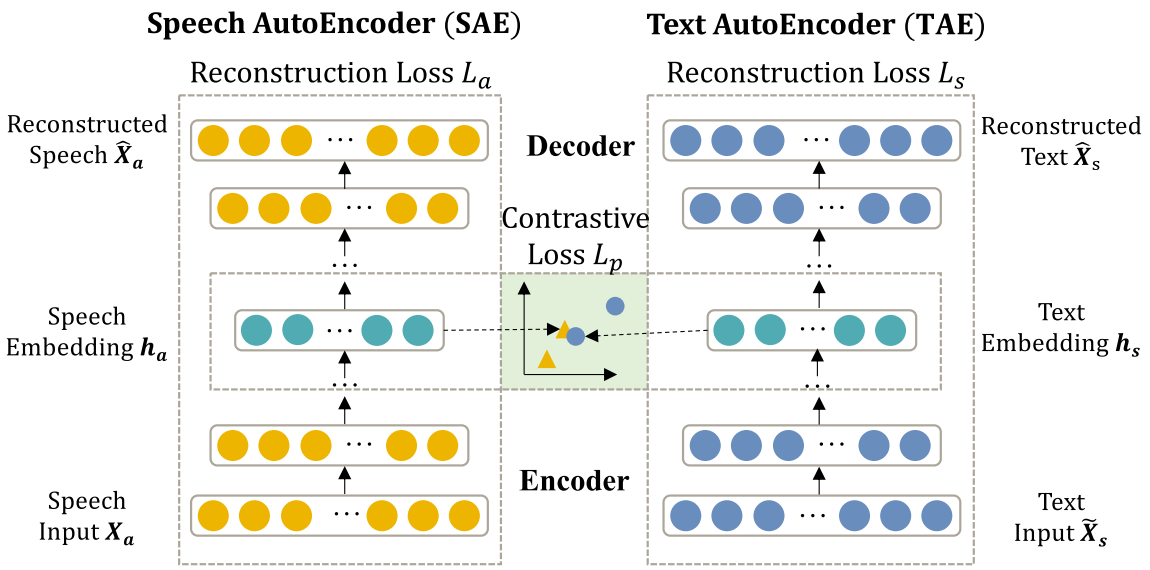


Fig. 6 The network structure in the speech–sentence pre-training step

$$\mathcal{L}_a = D_{KL}(\tilde{X}_a||\hat{X}_a), \tag{31}$$

$$\mathcal{L}_s = D_{KL}(\tilde{X}_s||\hat{X}_s), \tag{32}$$

$$\mathcal{L}_p = -\log \frac{exp(sim(h_a^b, h_s^b))}{\sum_{i[i\neq b]} \exp(sim(h_a^b, h_s^i))}, \tag{33}$$

$$sim(h_a^b, h_s^b) = \frac{(h_a^b)^T h_s^b}{\|h_a^b\| \cdot \|h_s^b\|}, \tag{34}$$

where $\hat{X}_a$ and $\hat{X}_s$ are the reconstructed outputs by the AEs, $D_{KL}(\cdot||\cdot)$ is a KL divergence used to measure the reconstruction loss, and $\tilde{X}_a$ and $\tilde{X}_s$ are the inputs of speech and transcript. We use two new notations $\tilde{X}_a$ and $\tilde{X}_s$ (rather than X_a and X_s) to emphasize the fact that the inputs of speech and transcript used in this pre-training step are not restricted to the speech-based NLQ and schema from the speech-to-SQL dataset. Any common ASR training data, which is more abundance in amount, can be used here. For contrastive loss, suppose a mini-batch $G = \left\{(\tilde{X}_a^b, \tilde{X}_s^b)\right\}_{b=1}^{N_b}$ with N_b examples is training simultaneously, and h_a^b and h_s^b are the intermediate representation of an example b where h_a^b (h_s^b) is produced by a max-pooling operation from $\tilde{Z}_a$ ($\tilde{Z}_s$). The loss is designed to reduce the distance between the same text and speech pair and enlarge the distance between different text and speech pair. After the AEs are trained, embeddings extracted from the speech and the schema encoder can be regarded as close enough to each other.

6.2 Speech–item pre-training

The objective of the speech–item pre-training phase serves as the same objective of the schema-linking step in the text-to-SQL task, that is, to explicitly identify if an item from the database schema is refereed by the speech NLQ. Given the speech query and an item from the schema, their embeddings are first extracted by the speech encoder and the text encoder trained in the above-mentioned pre-training phase. Then a cosine-similarity following by Sigmoid function is employed to predict the existence as Eq. (35).

$$\hat{\mathbf{y}}_\mathbf{f} = \texttt{Sigmoid}(W_f \texttt{sim}(Z_a, h_s) + b_f)), \tag{35}$$

where $W_f \in \mathbb{R}^{l_a}$ and $b_f \in \mathbb{R}$ are trainable parameters and l_a is dimensionality of the speech hidden space.

The training dataset for this pre-training step can easily be obtained from the original speech-to-SQL dataset. For any instance $\{x, y, S\}$ in the speech-to-SQL dataset, we could compose multiple new training instances, each of which can be denoted as $\{x, c, y_f\}$, where c is a column item from the schema S and the label y_f is equal to 1 if c appears in SQL y and is equal to 0 otherwise. We could obtain a new dataset D' with N' instances. Finally, we employ the cross-entropy loss in this pre-training step, defined as

$$\mathcal{L}(x, y_f) = -\sum_{i\in D'} y_f^i \log(\hat{y}_f^i) + (1 - y_f^i)\log(1 - \hat{y}_f^i) \tag{36}$$

where $\hat{y}_i \in \{0, 1\}$ is the predicted result.

6.3 Model fine-tuning

After above-mentioned pre-training phases, we obtain trained weights for the speech encoder and the schema (text) encoder. During this fine-tuning step, we will first load the weights by the pre-training step in advance if they exist and then train

the model globally with speech-to-SQL data, maximizing the log-likelihood of ground true action sequences defined as:

$$\mathcal{L} = \max \sum_{(x,y,S)\in\mathcal{D}} \left[\sum_{a_i \in ApplyRule} \log p(\tilde{y}_i = a_i | x, S, a_{<i}) + \sum_{a_i \in SelectSchema} \log p(\tilde{y}_i = a_i | x, S, a_{<i}) \right]. \tag{37}$$

The whole network is trained in an end-to-end style with stochastic gradient descent methods such as Adam [35].

7 Experiment

In this section, we evaluate the performance of the proposed model in terms of quantitative metrics. We first describe the experimental setup, including the datasets used in Sect. 7.1, baselines in Sect. 7.2, implementation details in Sect. 7.3 and evaluation measurements in Sect. 7.4. Then, we present the results in Sect. 7.5 to demonstrate the effectiveness of our proposed models by comparing them with these baselines.

7.1 Datasets

We use the same training set of the SpeechQL dataset to train all the models, tune their parameters with the same validation set and finally evaluate the performance on the same testing set. For the cascaded baselines, the ASR model is pre-trained with English datasets from general domains including Ted Talks [29, 63] and LibriSpeech [57], and then the pre-trained ASR model is further fine-tuned with the small-scale dataset in SQL domain (i.e., SpeechQL) to adapt to the database domain. For our design SpeechSQLNet approach, the speech–sentence pre-training is also conducted with the LibriSpeech [57] dataset.

7.2 Baselines

There is currently no literature that achieves direct SQL generation from common speech NLQs. Existing text-to-SQL approaches assume the availability of the transcripts for each speech, either labeled by human annotators or recognized by an ASR system. In our experiments, we compare carefully designed methods for this problem, which can be roughly categorized into the *cascaded* approach and the *end-to-end* approach. For the cascaded approach, the discussed well-trained ASR system in Sect. 4.1 is first used to decode the transcripts for each speech, and then a text-to-SQL model is adopted to generate the SQL queries. Specifically, we adopt the following four cascaded baselines.

- *ASR-Seq2SQL*: Seq2SQL [103] is the first DNN-based approach to solve text-to-SQL problem. The model is a straightforward Seq2Seq neural network, which takes the text as the input and the SQL statement as the output.
- *ASR-SQLNet*: SQLNet [89] employs a more refined slot filling strategy. It takes advantage of the fact that most of the queries in the WikiSQL dataset can be represented in a standard "`SELECT _ FROM _ WHERE _`" format.
- *ASR-IRNet*: IRNet [28] is an advanced text-to-SQL model that improves the above two methods by representing the SQL statement with the SemSQL grammar in an abstract syntax tree (AST) format. This representation captures the structure information of the SQL generation problem and achieves much better results than the previous methods.
- *ASR-EditSQL*: EditSQL [100] is another advanced text-to-SQL model that performs very well on SQL generation task. This model could reuse previously generation SQL results from adjacent NLQs, which makes it a very strong baseline for performance comparison.

For the end-to-end approach, we design and implement the following three baselines. It should be noted that these two end-to-end baseline models can not only work for speech-to-SQL, but also be used as end-to-end speech-to-code models.

– *SpeechSeq2SQL*: This baseline employs a vanilla Seq2Seq model directly trained on the speech-to-SQL dataset that converts the speech signals into its corresponding SQL. The encoder and the decoder are both Bi-LSTM, and we name this method SpeechSeq2SQL.
– *Transformer*: Transformer has shown quite promising performance on various NLP tasks from machine translation [18, 39] and dialogue system [40, 101], to ASR [99, 104]. In our experiment, we also implement the transformer structure according to [79], with the speech NLQs as the input and the SQL query as the output.
– *SpeechSQLNet*: This method is the end-to-end model that we designed in this work, which employs novel encoders, SQL-aware decoder and pre-training mechanisms that are dedicated to this speech-to-SQL task.

7.3 Implementation details

The speech encoder is composed of a six-layer CNN module with 1 input channel and 128 output channels, which takes speech features of the same size by resampling and zero-padding as needed. The schema encoder takes word embedding as inputs, which is initialized with a BiLSTM with 512 hidden units, and the outputs are produced by a two-layer GCN. The transformer has a two-layer encoder, with head size, feed-forward size and hidden size set to be

4, 1024 and 512, respectively. The dimensionality of action embedding and type embedding in the SQL-aware decoder are all set to 12. In addition, the pre-training models keep the same size as the above settings. During the training period, the learning rate, decay rate, dropout rate and batch size of the network are set to 0.0001, 0.8, 0.3 and 256, respectively. All the experiments were conducted on a server with a 72 Intel Core Processor (Xeon), 314 GB memory, Tesla K80 GPU and CentOS.

7.4 Evaluation metrics

Two widely used metrics in SQL generation task are used in our experiments to validate the effectiveness of the proposed model.

- *Query-Match Accuracy: This metric evaluates the percentage of matches between the generated SQL with the ground truth. To alleviate the negative effect of condition order, we further decouple each SQL statement into several parts: "select column", "aggregator" and "condition". Instead of directly comparing the SQL string, we take the condition part as a set and compare each element in the set.*
- *Average Time Per Query (TPQ): This metric evaluates the average inference time cost of a query by various methods, and it reflects how fast a model processes the speech query.*

7.5 Experimental results

7.5.1 Comparison of accuracy

The accuracies of our proposed model together with the baseline on the validation and test datasets are presented in Table 2, from which we could conclude the following observations.

Some end-to-end methods can outperform the cascaded methods by a large margin (e.g., ASR-Seq2SQL vs SpeechSeq2SQL), proving the necessity of exploring approaches that bypass the text for the speech-to-SQL problem. The end-to-end approach has the advantage of naturally retaining the rich linguistic information in the speech, and conducting global optimization to reduce the errors. However, their capabilities of capturing this linguistic information are not equally powerful. The performance of the baseline SpeechSeq2SQL method is less powerful than other end-to-end methods, and the reason is obvious. The SpeechSeq2SQL does not consider the specificity of the SQL generation problem and thus has a limited ability in both understanding the information converted in the speech and the database schema. The transformer-based structure could improve the performance compared with the basic SpeechSeq2SQL model, which is consistent with previous studies [39, 101]. Among all these methods, the proposed method, SpeechSQLNet, uses a powerful speech encoder and a schema encoder to extract meanings from the speech and database schema, and thus, it could beat all compared methods. Furthermore, the SQL-aware decoder can guide the generation process with the SQL grammar. This set of experiments verifies the rationale of the proposed speech-to-SQL problem and the effectiveness and necessity of the end-to-end approach.

Table 2 Performance of different speech-to-SQL models

Model	Validation Query Acc.	TPQ	Test Query Acc.	TPQ
ASR-Seq2SQL	0.0236	0.093	0.0195	0.089
ASR-Transformer	0.0273	0.096	0.0259	0.094
ASR-IRNet	0.4345	0.146	0.4500	0.140
ASR-EditSQL	0.4973	0.128	0.5116	0.125
SpeechSeq2SQL	0.0700	0.003	0.0695	0.003
Transformer	0.0755	0.003	0.0709	0.004
SpeechSQLNet	0.5355	0.070	0.5395	0.070

For the cascaded approach, we also included the detailed ratio of errors made by the first stage (i.e., ASR) and the second stage (i.e., text-to-SQL) in Table 3. Specifically, the errors made in the first stage is obtained by calculating the percentage of instances that ASR (of the ASR-X model) creates a cascading error among all instances that the whole ASR-X model creates an error. For example, suppose that there are N instances that the whole ASR-X model creates an error. Let M be the instances that the text-to-SQL module in the ASR-X model, taking the original text of the speech input, generates no error. Note that we know that the reason why these M instances have errors is the introduction of the error of the ASR module. Thus, the error rate of the first stage is equal to M/N. Similarly, the errors made in the second stage means the percentage of instances that text-to-SQL (of the ASR-X model) creates an error among all instances that the whole ASR-X model creates an error. Note that we know that for each of these $(N - M)$ instances, the text-to-SQL module in the ASR-X model, taking the original text of the speech input, generates an error. Thus, the error rate of the second stage is equal to $(N - M)/M$. We can see that the cascaded approaches do not perform very well and are not that competitive. The main reason is that they suffer from the error compounding problem between components, i.e., speech recognition errors leading to following SQL conversion errors, besides the errors made by the text-to-SQL module. From Table 3, we can see that for some cascaded baselines like ASR-IRNet and ASR-EditSQL, the errors caused by the ASR step account for more than 60% of all the errors.

Table 3 The ratio of errors from the two stages for the cascaded speech-to-SQL baselines

Model	Validation		Test	
	ASR stage	Text-to-SQL stage	ASR stage	Text-to-SQL stage
ASR-Seq2SQL	4.84%	95.16%	3.99%	96.01%
ASR-Transformer	6.54%	93.46%	7.19%	92.81%
ASR-IRNet	60.36%	39.64%	60.63%	39.37%
ASR-EditSQL	64.73%	35.27%	62.85%	39.37%

Table 4 The detailed accuracy of the speechSQLNet model

Data source	Validation	Test
Spider DB source	0.1667	0.1538
WikiSQL DB source	0.5416	0.5470

Table 5 Ablation study results: the effect of each network component

Model	Validation query acc.	Test query acc.
SpeechSQLNet	0.5355	0.5395
w/o GCN	0.5209	0.5159
w/o Linking	0.5273	0.5177
w/o Fusion	0.4600	0.4568
w/o SQL	0.3227	0.3250

We also split the results on our dataset from the database sources (i.e., from WikiSQL databases and Spider databases) and the results are shown in Table 4. The reasons for this unbalanced prediction rate between these two kinds are: (i) Even though the templates of these "WikiSQL DB Source" instances are coming from Spider, simple SQL queries from Spider are easier to use to generate correct new instances than complicated queries. Thus, the overall difficulty level of these queries from the "WikiSQL DB Source" is easier than the queries from the "Spider DB Source"; (ii) The training set contains more newly generated instances with "WikiSQL DB Source". Thus, the trained model is better in these cases, resulting in a higher prediction rate. From the result, we must admit that compared with text-to-SQL, which has achieved great development in recent years, speech-to-SQL still has a large room to improve. An important reason is that limited studies have been conducted on this valuable task and we hope this study would draw more attention from the community to this task. We believe that with more effort being put in, the performance of speech-to-SQL will also be greatly improved.

7.5.2 Ablation studies: the effect of each network component

In this section, we performed ablation studies (Table 5) to show the contribution of each component in SpeechSQLNet.

Table 6 Ablation study results: the effect of the pre-training mechanisms

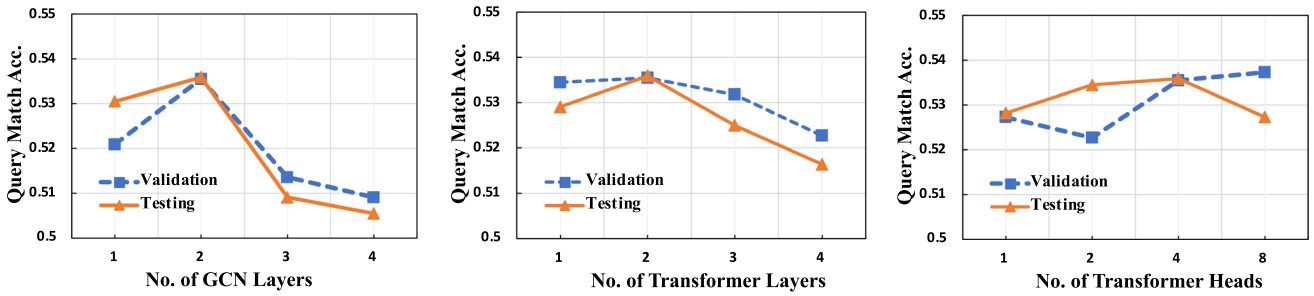
Model	Validation query acc.	Test query acc.
SpeechSQLNet	0.5355	0.5395
w/o SSPT	0.5136	0.5023
w/o SIPT	0.4982	0.5105
w/o Both	0.4655	0.4573

Specifically, we first evaluate the SpeechSQLNet with all the components as the baseline. Each model is then represented by the name(s) of the components that it removed or replaced. To evaluate the effectiveness of the schema encoder, we replace it with a vanilla RNN-based encoder (**w/o GCN**). For the schema-linking component (introduced in Sect. 5.3.1), we compared it with a baseline without it (**w/o Linking**). For the transformer-based fusion part, we remove it and name the baseline **w/o Fusion**. Finally, for the SQL-aware decoder, we replace it with a basic RNN-based decoder (**w/o SQL**).

Compared with the model without the schema-linking part, the performance decreases remarkably, up to 4.21% relative accuracy reduction. The significant reduction demonstrates the effectiveness of our proposed schema-linking mechanism in addressing the speech-to-SQL task. Other modules show similar conclusions. For example, the fusion mechanism brings around 18.10% relative improvement, while the SQL-aware decoder shows around 66.00% relative improvement. However, compared with a vanilla RNN encoder, the GNN-based schema encoder only shows a 4.57% relative improvement. The main reason is that the schema encoder mainly handles the complicated queries across tables, but most of the cases in our dataset involve SQL queries on single tables (from WikiSQL). However, a further case study with multiple tables in Sect. 7.5.6 shows the necessity of involving a GNN-based schema encoder in handling these complicated queries.

7.5.3 Ablation studies: the effect of the pre-training mechanisms

We also conduct another set of ablation studies to show the effectiveness of the two proposed pre-training mechanisms.



(a) GCN Layer (b) Transformer Layer (c) Transformer Head

Fig. 7 The hyper-parameter study of the SpeechSQLNet model

Table 7 The effect of our proposed pre-training mechanism (i.e., sentence-item pre-training, SIPT) on the text-to-SQL and the cascaded approach

Model	Validation query acc.	Test query acc.
IRNet	0.7709	0.7764
+ SIPT	0.7700	0.7745
ASR-IRNet	0.4345	0.4500
+ SIPT	0.4064	0.4232

Each model is still represented by the name(s) of the components that is/are removed, namely **w/o SSPT**, **w/o SIPT** and **w/o Both**, referring to trained models without the *s*peech-*s*entence *p*re-*t*raining mechanism, without the *s*peech-*i*tem *p*re-*t*raining mechanism, and without these two pre-training mechanisms, respectively.

As shown in Table 6, these two pre-training mechanisms together bring around 17.98% relative query match accuracy improvements in the test set, showing the necessity and effectiveness of involving these pre-training mechanisms to align the semantic representation for speech and text. Specifically, the speech–sentence pre-training mechanism solely brings about 7.41% relative query match accuracy improvement in the test set, while the speech–item pre-training mechanism solely shows a 5.68% relative query match accuracy improvement. Lastly, it is noted that, unlike the training step, the labeled data used in the pre-training step are not restricted to the SQL domain, and any labeled ASR dataset can be used.

Compared with text-to-SQL, speech-to-SQL is a much more complicated and harder task due to the huge modality gap between the two inputs (i.e., speech modality and schema (text) modality). Then, we design these two pre-training mechanisms which are proved to be effective for our speech-to-SQL model. An interesting problem would be if these two mechanisms are also effective for the text-to-SQL models (the cascaded approach). The SSPT mechanism aims to reduce the huge modality gap. For text-to-SQL, since the two inputs (i.e., question and schema) are already from the same text modality and could easily be mapped into the same hidden space (e.g., by Glove embedding [59]), our first mechanism, SSPT, cannot be used on text-to-SQL models. The second mechanism, SIPT, serves the goal of reducing the modality gap and meanwhile also enhances the schema-linking part between the speech-based NLQ and the schema. Theoretically speaking, it could also be customized for the text-to-SQL models and the cascaded approach. However, compared with speech-to-SQL, the schema-linking part is much easier for the text-to-SQL task since the question and the schema are both from the same text modality. In the text-to-SQL task, a string matching between the NLQ and the schema is proven to be good enough as a schema-linking step [28], and existing text-to-SQL models usually already contain their schema-linking sub-component. As a result, our designed SIPT would barely improve these text-to-SQL models when directly applied to them for the schema-linking purpose. This could also be validated by another set of experiments in Table 7, where the performance of the popular text-to-SQL model - IRNet (w/o ASR) keeps almost the same (i.e., 0.7764 vs 0.7745) after this pre-training mechanism. Furthermore, we notice that the performance of the cascaded approach (ASR-IRNet) with the SIPT mechanism is worse than the baseline, which indicates that the cascaded approach is not robust to the ASR errors (i.e., error compounding problem) and SIPT enlarges the effect of the ASR errors for the cascaded approach since it relies on the correctness of the text.

7.5.4 Hyper-parameter study

In this section, we study the performance variation affected by the parameters such as the number of GCN layers, the number of heads and the number of transformer layers. The results are shown in Fig. 7.

As shown in Fig. 6, a large number of GCN layers does not always lead to better performance. The performance reaches a peak when the number of layers is set to 2 and then decreases when the number exceeds this value. Previous studies on GNN [16] have also shown similar observations. The reason is obvious. That is, the learning capacity of the model

Table 8 Two SQL examples generated by various cascaded and the end-to-end methods

Case	Case 1	Case 2
Speech Query	What is the lowest number of draws with more than 1 byes?	What is the average number of votes of representatives from party “Republican"
Schema	wimmra(Wimmera_fl,wins,byes,losses, draws,against)	election(election_id, representative_id, date, votes, vote_percent, seats, place), representative(representative_id, name, state, party, lifespan)
Ground Truth SQL	select min(draws) from wimmera where byes > 1	select avg(t1.votes) from election as t1 join representative as t2 on t1.representative_id = t2.representative_id where t2.party = 1
ASR Result	what is the lowest number of drawers with more than one bites	what is the average number of votes of representatives from party republican
ASR-Seq2SQL [103]	select min(no) from 2008 where long > 1	select avg(#) from united where party = 1
ASR-Transformer [79]	select min(jake) from 2008 where runner > 1 and pass_def > 1	select avg(round) from list where nation = 1 and party = 1
ASR-IRNet [28]	select min(*) from wimmera where draws > 1	select avg(t1.votes) from election as t1 join representative as t2 on t1.representative_id = t2.representative_id where t2.party = 1
ASR-EditSQL [100]	select min(draws) from wimmera where wimmera_fl = 1	select avg(votes) from representative where party = 1
SpeechSeq2SQL [74]	select min(draws) from imperfect where byes > 1	select avg(votes) from list where party = 1
Transformer [79]	select min(draws) from golden where byes > 1	select avg(votes) from list where difference =1
SpeechSQLNet (Our Proposed Model)	select min(draws) from wimmera where byes > 1	select avg(t1.votes) from election as t1 join representative as t2 on t1.representative_id = t2.representative_id where t2.party = 1

increases with the number of the layers, but, too many layers require more data to train [45]. Then, for the other two parameters - the number of transformer layers and the number of heads, we also observe similar phenomena in Fig. 7b and Fig. 7c, respectively.

7.5.5 The robustness analysis

In this section, we add some noise in the input speech-based NLQs and test the robustness of the methods. Specifically, we add the different ratios of noise into the original speech waveform and then check the performance of the speech-to-SQL model. The signal-to-noise ratio (SNR) is used to evaluate the degree of noise added, and the smaller the value, the more noise it is added. The result is shown in Fig. 8.

As we can see, with the input of more noise, the accuracy of the models decreases slightly. However, there is around a 5% accuracy decrease when the noisy level increases to 40db. This validates that our model (as well as the cascaded methods) has some robustness. We believe that exploring robust speech-driven SQL systems (e.g., more comprehensive study of model robustness against lexical and phrasal variability [24]) will be a very promising research direction, and we left it as a next step study.

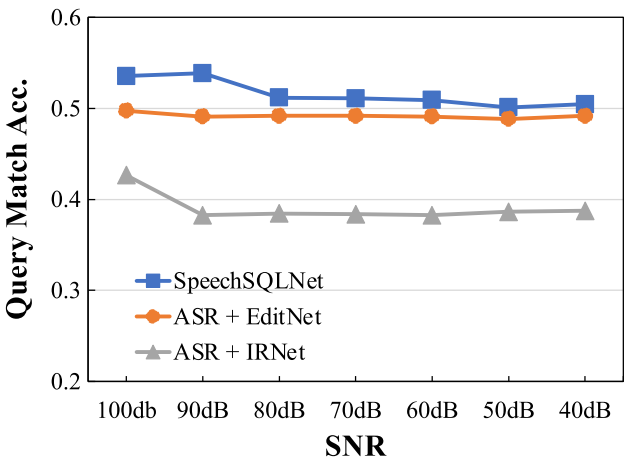


Fig. 8 The robustness analysis

7.5.6 Case study

We list two cases to vividly show the generated SQLs by the various baselines as well as our model in Table 8. The

Table 9 A failure example due to the challenges in speech-to-SQL

Case	Case 1
Speech Query	Give me the name of the customer who ordered the most items in total.
Schema	Addresses [address_id, address_content, city, zip_postcode, state_province_county, country, other_address_details], Products [product_id, product_details], Customers [customer_id, payment_method, customer_name, date_became_customer, other_customer_details], Customer_Addresses [customer_id, address_id, date_address_from, address_type, date_address_to], Customer_Contact_Channels [customer_id, channel_code, active_from_date, active_to_date, contact_number], Customer_Orders [order_id, customer_id, order_status, order_date, order_details], Order_Items [order_id, product_id, order_quantity]
Ground Truth SQL	SELECT t1.customer_name FROM customers AS t1 JOIN customer_orders AS t2 ON t1.customer_id = t2.customer_id JOIN order_items AS t3 ON t2.order_id = t3.order_id GROUP BY t1.customer_name ORDER BY sum(t3.order_quantity) DESC LIMIT 1
Our Predicted	SELECT T1.customer_name FROM Customers AS T1 JOIN Customer_Orders AS T2 ORDER BY T2.order_date ASC LIMIT 1

first case illustrates the error compounding problem between the ASR and the text-to-SQL components. The cascaded methods fail since the ASR model misrecognized "*draws*" to "*drawers*" and "*byes*" to "*bites*", and thus, the downstream text-to-SQL models all failed. This reflects the fact that the robustness of the cascaded solution is weak and the error compounding problem greatly affects the final SQL conversion accuracy. However, the results generated by the end-to-end baselines such as SpeechSeq2SQL and Transformer are partly correct. Compared with all these baselines, our designed SpeechSQLNet could accurately capture the semantics conveyed by the speech NLQs and then generates the SQL query accurately.

The second case shows how the schema could affect the predicted SQL. Since the required columns are from two tables, the desired SQL has a complicated 'Join' operations between the 'election' and the 'representative' tables. In this case, even the ASR module correctly recognizes the transcript, most of the cascaded approaches such as ASR-EditSQL and ASR-Transformer still fail to give the correct predictions. In contrast, SpeechSQLNet could accurately identify the corresponding tables and columns, which validates the necessity of the end-to-end approach for this speech-to-SQL task.

7.5.7 Failure analysis and challenges

We also analyze the cases that still fail by the current speech-to-SQL model. We identified the following two challenging points from the observations. (i) The large dataset required to construct speech-driven applications and models; As we mentioned before, speech-driven models usually require a larger dataset to train the model compared with text-based models. For these failed cases, increasing the size of the training datasets will alleviate this problem and improve the accuracy. Thus, one potential approach is to use other datasets (e.g., weakly labeled data) to pre-train the speech-to-SQL model. In the text-to-SQL area, there are these pre-training mechanisms like GraPPA [93]. (ii) The huge modality gap between speech-based NLQ and schema; Even though we designed a novel mechanism to alleviate the huge modality gap problem between speech and schema, there are still rooms to improve it further, and we believe pre-training language models (PLMs) would be a potential research direction to this problem. For the text-to-SQL problem, we already have various PLMs (e.g., TaBERT [90] and TaPas [30]) that could generate high-quality hidden representations for the text-based input. These PLMs are usually trained with very large-scale datasets, and thus, they could accurately capture the hidden semantic information behind the text and the schema. However, currently, no previous studies have been conducted on PLMs dedicated to speech-to-SQL (i.e., a PLM that could understand both speech-based NLQ and schema). We believe this would also be a good future research direction for this problem. A case that failed due to this reason is shown in Table 9.

7.5.8 Summary and limitations

The experimental evaluations on the SpeechQL dataset demonstrate the superiority of SpeechSQLNet model over several strong baselines, including various cascaded ones as well as the end-to-end ones. For example, SpeechSQLNet achieves up to 10.22% exact match accuracy improvements compared with the advanced IRNet model. The effectiveness of each designed component is further validated by the ablation study. Specifically, the novel pre-training mechanisms bring around 17.98% relative query match accuracy improvements in the test set. The two cases in Sect. 7.5.6 also vividly reflect the value of this problem and the superiority of our proposed method.

We also identify several limitations of this study and they have shown potential directions for next step research. (i) Human-generated Datasets. The speech NLQs are currently generated using the TTS techniques. Even though this approach is common in studying speech-driven applications [47, 51, 83, 84, 87], it would be better if we could hire diverse native speakers to generate the speech waves of the NLQs. This will require much more effort, and we expect the community will propose this kind of study after we release this paper, using techniques like crowd-sourcing. (ii) Robust speech-to-SQL models. In the ASR area, an important research direction is focusing on constructing robust systems. In text-to-SQL area, many studies [24] have also been conducted to explore the robustness against lexical and phrasal variability. For the speech-to-SQL problem, an interesting following direction is to construct robust speech-to-SQL systems and models. (iii) Other network structures and alternatives. We only design and explore the current network structure for this problem. This model validates the possibility of direct SQL generation from speech NLQs without the intermedia of text. We believe there are more other novel neural networks to further improve the performance, just like the rich community in the parallel text-to-SQL area. (iv) More Difficult Datasets with Diverse SQL Queries. The experimental analysis shows that the proposed model performs better on simple queries, which make up a substantial portion of the dataset. It would be very valuable to construct large-scale speech-to-SQL datasets containing more difficult and diverse SQL queries to further challenge the task.

8 Related work

Our topic is closely related to the research fields of text-to-SQL (Sect. 8.1), speech-driven querying systems (Sect. 8.2) and speech-to-X (Sect. 8.3). We briefly survey the most related work from these three aspects.

8.1 Text-to-SQL

Text-to-SQL aims to provide databases with a text-based interface. It has a long history dating back to 1970 s. Past literature in this field can be classified into two categories: rule and template-based approach and end-to-end neural network-based approach. Typical studies in rule and template-based approaches include studies such as SODA [10], QUICK [98], SINA [67] and NLQ/A [102]. These approaches normally work in a similar style and suffer the drawback of poor flexibility since users may express their questions using different linguistic styles. For example, SODA [10] adopts a five-step pipeline to translate a keyword-based NL input question into a SQL query. NLQ/A [102] extends the input to pattern-based NL and supports more complex questions like concepts. A comprehensive survey on classic approaches can be found in [3].

Recent trends in deep neural networks (DNNs) also promote the development on end-to-end neural network-based approaches. Representative studies in this category include Seq2SQL [103], SQLNet [89], TypeSQL [92], Syntax SQL [73], IRNET [28] and its extensions such as NL2pSQL [15]. Furthermore, many studies focus on pre-train the text-to-SQL models with augmented data, with representative studies like [93] and [95]. Compared with the traditional approaches, these neural-based models have the advantages of reducing the workload of engineers of maintaining multiple components and, at the same time, achieve a much better performance since the whole model enjoys a single optimization objective (i.e., SQL generation accuracy). The neural-based approach has already dominated the text-to-SQL area. A survey that summarizes recent progress in these neural-based models can be found in [31].

The success of the neural-based approach relies heavily on the releasing of evaluation benchmarks. There are some public datasets widely used in the community, such as Spider [96], TableQA [72] and WikiSQL [103]. The Spider dataset is a small-scale cross-domain dataset that contains complex nested and join queries. WikiSQL is much larger in size, but, it mainly focuses on simple queries on limited domains. TableQA is a large-scale cross-domain NL2SQL dataset in the Chinese language. However, for speech-to-SQL, despite several studies related to speech-driven querying systems [49, 54, 65], there are no public benchmarks in the community. Our constructed SpeechQL dataset aims to fill this void.

The rapid development of the neural-based text-to-SQL approach also influences the development of the speech-to-SQL area. On one hand, even though text-to-SQL can bridge the gap between SQL and Natural Language Processing (NLP) and can provide a text-based interface for DMBS, we believe that the voice-based interface is a much easier and faster way for interacting with DBMS. According to the user study in [65], the voice-based interface can enable users to compose SQL queries considerably faster by up to 6.7x compared to typing on a text-based interface in a tablet device. Hence, in this work, we take one step further to explore the possibility of a speech-driven interface for databases. On the other hand, the development history and trend in text-to-SQL (i.e., from traditional *multi-stage* or *multi-component* approach to recent *end-to-end* neural-based approach) also inspires us (i) to formally define this speech-to-SQL task and construct the benchmarks; (ii) to explore this end-to-end approach for speech-to-SQL task, which is a pioneering study in this area. Our work is a good start that promotes the development of speech-to-SQL, and we expect speech-to-SQL will soon become as popular as text-to-SQL.

8.2 Speech-driven querying systems

Speech-driven querying systems have a long history with a substantial amount of industrial applications. To name a few, Nuance's Dragon NaturallySpeaking[4] supports various basic voice commands. Mainstream search engines such as Google, Baidu and Bing all provide voice-based inputs to search the web. With the popularity of smart devices and mobile phones, AI-powered assistants such as Xiaoice, Siri, Alexa and Google Home also provide user interaction with the system based on voice, e.g., querying daily weather and traffic, keeping track of airline flight and so on.

Speech-driven SQL-based systems have also been studied in the research community. For example, EchoQuery [49] aims to translate user's voice input into SQL queries. SpeakQL [65, 66] supports a subset of SQL statements and enables users to interact with the system with a speech-based interface. TalkSQL [54] implements a similar function to SpeakQL, which also works in a three-step manner, first allowing a user to input some voice command, then translating the inputs into SQL queries and finally delivering execution results to the user. CiceroDB-Zero [77] enables participants to explore large data sets via voice interfaces.

However, none of these studies focuses directly on SQL generation from *arbitrary* common NLQs expressed in speech, and they usually work in a cascaded manner. Furthermore, some of them restrict the spoken queries to be an NL-based version of SQL or its variants with a limited subset of SQL grammar, and thus, they still require users to have a professional background in SQL. Furthermore, our developed VoiceQuerySystem [70] also demonstrates the feasibility of a speech-driven database querying system with the cascaded and the end-to-end techniques provided in this study.

8.3 Speech-to-X

Speech-to-X refers to a wide range of speech-driven tasks including speech-to-text, speech-to-image, speech-to-code, speech-to-model and so on. Among all these tasks, speech-to-text is the most common one with dozens of existing studies. When the text refers to the transcript corresponding to the speech, this problem is also well-known as the ASR problem. Currently, the most commonly used ASR model is the hybrid model, which typically consists of two components: an acoustic model (AM) and a language model (LM). The AM translates the speech signals features (e.g., MFCC [23]) into the corresponding phonemic representation, and the LM calculates the probability of the decoded word sequences from the natural language perspective. With the dominating performance of DNN-based models in NLP tasks, the end-to-end ASR system also becomes quite popular in the research community with studies such as LAS [13], RNN Transducer [61], attention-based models [7, 85] and RNN transducer with Attention [75]. However, in terms of ASR applications in the industry, the hybrid one still dominates the market due to its relatively good performance. In our work, we also employ the hybrid ASR [68, 69] in the cascaded baselines.

The speech-to-text problem also covers diverse applications besides ASR. Another task worth mentioning is the speech-to-text translation [8], where the "text" here refers to the translated text in another language different from the one in the source speech. The speech-to-text translation is very helpful especially for low-resource scenarios where neither machine translation nor ASR is available. Speech-to-speech translation [81] takes one step further to directly generate speech in the target language, and it is applied in NLP applications such as cross-lingual dialogue systems. Besides text, dozens of studies have also explored the possibility of converting other modalities such as image (i.e., speech-to-image) [47, 83, 84] and software model (i.e., voice-driven modeling) [9]. Our proposed speech-to-SQL problem can also be considered as a special case of the speech-to-code problem (i.e., programming-by-voice) [19, 41]. To the best of our knowledge, in the literature of the speech-to-code, there are studies for generating code in general programming languages such as Python and Java [1, 2]. All these systems (e.g., Serenade[5] and Talon[6]) could not be adapted or used to generate SQL. This is because i) SQL is more complicated due to the existence of database schema; ii) these systems usually work by mapping limited pre-defined voice-based programming commands into code (e.g., delete line three to line four), and they have a limited ability both in understanding common NLQ and translating NLQ to code because they usually involve limited (DNN) learning-based models. They usually require users to iteratively input a series of predefined commands rather than an NLQ to generate a program (e.g., "add class Test", $\cdots$, "go to line two", "add function string helloworld", "add return string hello world" and creating a simple "Test" class with a function that returns a "helloworld" string). In terms of high-level design logic, these studies are similar to existing speech-to-SQL systems like SpeakQL and EchoQuery surveyed in Sect. 8.2, which either requires the user's input to be an exact SQL query or follow some pre-defined command templates. Since none of them achieve translates common NLQ into programming code (including SQL), these systems still require the users have a strong background in programming languages.

To sum up, speech-to-SQL is an urgent task that aims to explore the information conveyed by human speech and convert it into a SQL statement. These aforemen-

[4] https://www.nuance.com/dragon

[5] https://serenade.ai/

[6] https://talonvoice.com/

tioned speech-driven applications, including speech-to-SQL, directly generate the target due to following reasons: (i) avoiding the possible error compounding between components in the cascaded systems; (ii) further reducing the error by retaining the rich linguistic information in the speech; and (iii) improving the query processing efficiency; (iv) unlocking the power of advanced NLP technologies to the languages that lack a commonly used written form [84]. We believe that speech-to-SQL would not only benefit the database area for providing user-friendly interfaces of DBMS, but also promote research of programming-by-voice in software engineering and enlarge the speech-to-X family.

9 Conclusion and discussion

In this paper, we propose a novel paradigm speech-driven interface for the relational database, together with its corresponding task speech-to-SQL, aiming to directly convert human speech into SQL queries. Cascaded methods, as well as end-to-end models named SpeechSQLNet, are proposed to solve this problem. Extensive experimental evaluations on the constructed corpus show that SpeechSQLNet can generate high-quality SQL queries, outperforming several competitive baselines. The experimental results also validate the rationale of the speech-to-SQL problem.

In the next step, we would like to explore other novel end-to-end network structures on speech-to-SQL. In particular, as a good start, SpeechSQLNet shows that speech signals contain rich linguistic information, and it would be interesting to explore the inner structure of speech signals when designing speech-to-SQL models. Furthermore, VoiceQuerySystem validates the feasibility of a speech-driven database querying system with the techniques provided in this study, and we would also like to follow this line of research of developing more user-friendly speech-driven systems, especially in vertical domains like the medical or ticket booking scenarios.

The recent trend of large language models (LLMs, e.g., ChatGPT [55], GPT-4 [56], LLaMA [76] and ChatGLM [97][21]) has evolved almost all areas, including the NLP and database communities. While LLMs have demonstrated impressive reasoning abilities, such as few-shot and zero-shot learning [12] as well as chain-of-thought reasoning [86], their scale results in substantial computational costs. In contrast, traditional neural network models like our designed SpeechSQLNet benefit from cheaper training due to their relatively compact size. For most of the existing tasks, including text-to-SQL, users could design suitable prompts for querying the LLMs about the corresponding SQL query. Rather than spending efforts in designing the architecture of the neural networks, the focus of solving existing tasks by prompting LLMs lies in how to design suitable prompts (i.e., prompting engineering) and how to decouple the complicated task into multiple sub-tasks and then tackle them step-by-step. The popularity of LLMs has provided us with new opportunities and alternative approaches to solving text-to-SQL and speech-to-SQL problems. In a nutshell, we believe that speech-to-SQL will become as popular a task as text-to-SQL or other speech-to-X tasks and will also inspire further work on designing novel speech-driven applications in the database area.

Acknowledgements We thank the editor and the reviewers for their valuable comments in improving this draft. Xuefang Zhao is the corresponding author.

References

1. Serenade ai. (Last accessed 16 Oct. 2022). https://serenade.ai/
2. Talon voice. (Last accessed 16 Oct. 2022). https://talonvoice.com/
3. Affolter, K., Stockinger, K., Bernstein, A.: A comparative survey of recent natural language interfaces for databases. VLDB J. **28**(5), 793–819 (2019)
4. Alateeq, A., Roantree, M., Gurrin, C.: Voxento: A prototype voice-controlled interactive search engine for lifelogs. In: Proceedings of the Third Annual Workshop on Lifelog Search Challenge, pp. 77–81 (2020)
5. Audhkhasi, K., Rosenberg, A., Sethy, A., Ramabhadran, B., Kingsbury, B.: End-to-end asr-free keyword search from speech. IEEE J. Selected Top. Signal Process. **11**(8), 1351–1359 (2017)
6. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. In: NIPS 2016 Deep Learning Symposium (2016)
7. Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., Bengio, Y.: End-to-end attention-based large vocabulary speech recognition. In: 2016 ICASSP, pp. 4945–4949. IEEE (2016)
8. Bansal, S., Kamper, H., Lopez, A., Goldwater, S.: Towards speech-to-text translation without speech recognition. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 474–479 (2017)
9. Black, D., Rapos, E.J., Stephan, M.: Voice-driven modeling: Software modeling using automated speech recognition. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 252–258. IEEE (2019)
10. Blunschi, L., Jossen, C., Kossmann, D., Mori, M., Stockinger, K.: Data-thirsty business analysts need soda: search over data warehouse. In: Proceedings of the 20th ACM international conference on Information and knowledge management, pp. 2525–2528 (2011)
11. Bogin, B., Berant, J., Gardner, M.: Representing schema structure with graph neural networks for text-to-sql parsing. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4560–4565 (2019)
12. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Adv. Neural Inf. Process. Syst. **33**, 1877–1901 (2020)
13. Chan, W., Jaitly, N., Le, Q., Vinyals, O.: Listen, attend and spell: a neural network for large vocabulary conversational speech recognition. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4960–4964. IEEE (2016)

14. Chazan, D., Hoory, R., Cohen, G., Zibulski, M.: Speech reconstruction from mel frequency cepstral coefficients and pitch frequency. In: 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100), vol. 3, pp. 1299–1302. IEEE (2000)
15. Chen, F., Hwang, S.w., Choo, J., Ha, J.W., Kim, S.: Nl2psql: Generating pseudo-sql queries from under-specified natural language questions. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 2603–2613 (2019)
16. Chen, T., Wong, R.C.W.: Handling information loss of graph neural networks for session-based recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1172–1180 (2020)
17. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing (EMNLP 2014) (2014)
18. Currey, A., Heafield, K.: Incorporating source syntax into transformer-based neural machine translation. In: Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers), pp. 24–33 (2019)
19. Désilets, A., Fox, D.C., Norton, S.: Voicecode: An innovative speech interface for programming-by-voice. In: CHI'06 Extended Abstracts on Human Factors in Computing Systems, pp. 239–242 (2006)
20. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (2019)
21. Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z., Tang, J.: Glm: General language model pretraining with autoregressive blank infilling. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 320–335 (2022)
22. Fan, Y., Qian, Y., Xie, F.L., Soong, F.K.: Tts synthesis with bidirectional lstm based recurrent neural networks. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
23. Foote, J.T.: Content-based retrieval of music and audio. In: Multimedia Storage and Archiving Systems II, vol. 3229, pp. 138–147. International Society for Optics and Photonics (1997)
24. Gan, Y., Chen, X., Huang, Q., Purver, M., Woodward, J.R., Xie, J., Huang, P.: Towards robustness of text-to-sql models against synonym substitution. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 2505–2515 (2021)
25. Gkini, O., Belmpas, T., Koutrika, G., Ioannidis, Y.: An in-depth benchmarking of text-to-sql systems. In: Proceedings of the 2021 International Conference on Management of Data, pp. 632–644 (2021)
26. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323. JMLR Workshop and Conference Proceedings (2011)
27. Graves, A.: Long short-term memory. In: Supervised Sequence Labelling with Recurrent Neural Networks, pp. 37–45. Springer (2012)
28. Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.G., Liu, T., Zhang, D.: Towards complex text-to-sql in cross-domain database with intermediate representation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4524–4535 (2019)
29. Hernandez, F., Nguyen, V., Ghannay, S., Tomashenko, N., Estève, Y.: Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation. In: International Conference on Speech and Computer, pp. 198–208. Springer (2018)
30. Herzig, J., Nowak, P.K., Mueller, T., Piccinno, F., Eisenschlos, J.: Tapas: Weakly supervised table parsing via pre-training. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 4320–4333 (2020)
31. Iacob, R.C.A., Brad, F., Apostol, E.S., Truică, C.O., Hosu, I.A., Rebedea, T.: Neural approaches for natural language interfaces to databases: a survey. In: Proceedings of the 28th International Conference on Computational Linguistics, pp. 381–395 (2020)
32. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)
33. Kedar, S.: Database Management System. Technical Publications (2009)
34. Kim, H., So, B.H., Han, W.S., Lee, H.: Natural language to sql: Where are we today? Proceedings of the VLDB Endowment **13**(10), 1737–1750 (2020)
35. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. Tech. rep. (2014)
36. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings (2017)
37. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1097–1105 (2012)
38. Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W.Z., Sotelo, J., de Brébisson, A., Bengio, Y., Courville, A.C.: Melgan: Generative adversarial networks for conditional waveform synthesis. Adv. Neural Inf. Process. Syst. **32** (2019)
39. Lakew, S.M., Cettolo, M., Federico, M.: A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 641–652 (2018)
40. Le, H., Sahoo, D., Chen, N., Hoi, S.: Multimodal transformer networks for end-to-end video-grounded dialogue systems. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 5612–5623 (2019)
41. Lee, H., Fenwick Jr, J.B., Klima, R.E., McRae, A.A., Vahlbusch, J.: Disability assistive programming: using voice input to write code. Ph.D. thesis, Appalachian State University (2019)
42. Lei, W., Wang, W., Ma, Z., Gan, T., Lu, W., Kan, M.Y., Chua, T.S.: Re-examining the role of schema linking in text-to-sql. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 6943–6954 (2020)
43. Li, F., Jagadish, H.: Constructing an interactive natural language interface for relational databases. Proc. VLDB Endow. **8**(1), 73–84 (2014)
44. Li, F., Jagadish, H.V.: Nalir: an interactive natural language interface for querying relational databases. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 709–712 (2014)
45. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9267–9276 (2019)
46. Li, G., Zhou, X., Cao, L.: Ai meets database: Ai4db and db4ai. In: Proceedings of the 2021 International Conference on Management of Data, pp. 2859–2866 (2021)

47. Li, J., Zhang, X., Jia, C., Xu, J., Zhang, L., Wang, Y., Ma, S., Gao, W.: Direct speech-to-image translation. IEEE J. Selected Top. Signal Process. **14**(3), 517–529 (2020)
48. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421 (2015)
49. Lyons, G., Tran, V., Binnig, C., Cetintemel, U., Kraska, T.: Making the case for query-by-voice with echoquery. In: Proceedings of the 2016 International Conference on Management of Data, pp. 2129–2132 (2016)
50. Medsker, L.R., Jain, L.: Recurrent neural networks. Des. Appl. **5** (2001)
51. Nguyen, D.Q., et al.: Investigating the impact of asr errors on spoken implicit discourse relation recognition. In: Proceedings of the First Workshop On Transcript Understanding, pp. 34–39 (2022)
52. Nguyen, T.Q.: Near-perfect-reconstruction pseudo-qmf banks. IEEE Trans. Signal Process. **42**(1), 65–76 (1994)
53. Nihalani, N., Silakari, S., Motwani, M.: Natural language interface for database: a brief review. Int. J. Comput. Sci. Issues (IJCSI) **8**(2), 600 (2011)
54. Obaido, G., Ade-Ibijola, A., Vadapalli, H.: Talksql: A tool for the synthesis of sql queries from verbal specifications. In: 2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC), pp. 1–10. IEEE (2020)
55. OpenAI: Chatgpt (2023). https://openai.com/blog/chatgpt
56. OpenAI: Gpt-4 technical report (2023)
57. Panayotov, V., Chen, G., Povey, D., Khudanpur, S.: Librispeech: an asr corpus based on public domain audio books. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5206–5210. IEEE (2015)
58. Peng, Z., Mo, K., Zhu, X., Chen, J., Chen, Z., Xu, Q., Ma, X.: Understanding user perceptions of robot's delay, voice quality-speed trade-off and gui during conversation. In: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–8 (2020)
59. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
60. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al.: The kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, CONF. IEEE Signal Processing Society (2011)
61. Rao, K., Sak, H., Prabhavalkar, R.: Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In: 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 193–199. IEEE (2017)
62. Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., Liu, T.Y.: Fastspeech 2: Fast and high-quality end-to-end text to speech. In: International Conference on Learning Representations (2020)
63. Rousseau, A., Deléglise, P., Esteve, Y.: Ted-lium: an automatic speech recognition dedicated corpus. In: LREC, pp. 125–129 (2012)
64. Sen, J., Lei, C., Quamar, A., Özcan, F., Efthymiou, V., Dalmia, A., Stager, G., Mittal, A., Saha, D., Sankaranarayanan, K.: Athena++ natural language querying for complex nested sql queries. Proc. VLDB Endow. **13**(12), 2747–2759 (2020)
65. Shah, V., Li, S., Kumar, A., Saul, L.: Speakql: Towards speech-driven multimodal querying of structured data. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 2363–2374 (2020)
66. Shah, V., Li, S., Yang, K., Kumar, A., Saul, L.: Demonstration of speakql: speech-driven multimodal querying of structured data. In: Proceedings of the 2019 International Conference on Management of Data, pp. 2001–2004 (2019)
67. Shekarpour, S., Marx, E., Ngomo, A.C.N., Auer, S.: Sina: Semantic interpretation of user queries for question answering on interlinked data. J. Web Semant. **30**, 39–51 (2015)
68. Song, Y., Jiang, D., Huang, X., Li, Y., Xu, Q., Wong, R.C.W., Yang, Q.: Goldenretriever: A speech recognition system powered by modern information retrieval. In: Proceedings of the 28th ACM International Conference on Multimedia, pp. 4500–4502 (2020)
69. Song, Y., Jiang, D., Zhao, X., Xu, Q., Wong, R.C.W., Fan, L., Yang, Q.: L2rs: A learning-to-rescore mechanism for hybrid speech recognition. In: Proceedings of the 29th ACM International Conference on Multimedia, pp. 1157–1166 (2021)
70. Song, Y., Wong, R.C.W., Xuefang, Z., Jiang, D.: Voicequerysystem: a voice-driven database querying system using natural language questions. In: Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data (2022)
71. Stolcke, A.: Srilm-an extensible language modeling toolkit. In: Seventh International Conference on Spoken Language Processing (2002)
72. Sun, N., Yang, X., Liu, Y.: Tableqa: a large-scale chinese text-to-sql dataset for table-aware sql generation. arXiv pp. arXiv–2006 (2020)
73. Sun, Y., Tang, D., Duan, N., Ji, J., Cao, G., Feng, X., Qin, B., Liu, T., Zhou, M.: Semantic parsing with syntax-and table-aware sql generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 361–372 (2018)
74. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. Adv. Neural Inf. Process. Syst. **27** (2014)
75. Tian, Z., Yi, J., Tao, J., Bai, Y., Wen, Z.: Self-attention transducers for end-to-end speech recognition. Proc. Interspeech **2019**, 4395–4399 (2019)
76. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
77. Trummer, I.: Demonstrating the voice-based exploration of large data sets with cicerodb-zero. Proc. VLDB Endow. **13**(12), 2869–2872 (2020)
78. Utama, P., Weir, N., Binnig, C., Cetintemel, U.: Voice-based data exploration: Chatting with your database. In: Proceedings of the Workshop on Search-Oriented Conversational AI (SCAI) (2017)
79. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000–6010 (2017)
80. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. Adv. Neural Inf. Process. Syst. **28**, 2692–2700 (2015)
81. Wahlster, W.: Verbmobil: Foundations of Speech-to-Speech Translation. Springer Science & Business Media, Berlin (2013)
82. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J.: Phoneme recognition using time-delay neural networks. IEEE Trans. Acoust. Speech Signal Process. **37**(3), 328–339 (1989)
83. Wang, X., Qiao, T., Zhu, J., Hanjalic, A., Scharenborg, O.: S2igan: Speech-to-image generation via adversarial learning. In: INTERSPEECH 2020, pp. 2292–2296. ISCA (2020)
84. Wang, X., Qiao, T., Zhu, J., Hanjalic, A., Scharenborg, O.: Generating images from spoken descriptions. IEEE/ACM Trans. Audio Speech Language Process. **29**, 850–865 (2021)
85. Watanabe, S., Hori, T., Kim, S., Hershey, J.R., Hayashi, T.: Hybrid ctc/attention architecture for end-to-end speech recognition. IEEE J. Selected Top. Signal Process. **11**(8), 1240–1253 (2017)
86. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits rea-

soning in large language models. Adv. Neural Inf. Process. Syst. **35**, 24824–24837 (2022)
87. Weller, O., Sperber, M., Pires, T., Setiawan, H., Gollan, C., Telaar, D., Paulik, M.: End-to-end speech translation for code switched speech. In: Findings of the Association for Computational Linguistics: ACL 2022, pp. 1435–1448 (2022)
88. Xu, J., Tan, X., Ren, Y., Qin, T., Li, J., Zhao, S., Liu, T.Y.: Lrspeech: Extremely low-resource speech synthesis and recognition. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2802–2812 (2020)
89. Xu, X., Liu, C., Song, D.: Sqlnet: Generating structured queries from natural language without reinforcement learning. arXiv preprint arXiv:1711.04436 (2017)
90. Yin, P., Neubig, G., Yih, W.t., Riedel, S.: Tabert: Pretraining for joint understanding of textual and tabular data. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8413–8426 (2020)
91. Yu, D., Deng, L.: AUTOMATIC SPEECH RECOGNITION. Springer (2016)
92. Yu, T., Li, Z., Zhang, Z., Zhang, R., Radev, D.: Typesql: Knowledge-based type-aware neural text-to-sql generation. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pp. 588–594 (2018)
93. Yu, T., Wu, C.S., Lin, X.V., Tan, Y.C., Yang, X., Radev, D., Xiong, C., et al.: Grappa: Grammar-augmented pre-training for table semantic parsing. In: International Conference on Learning Representations (2020)
94. Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., Radev, D.: Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 1653–1663 (2018)
95. Yu, T., Zhang, R., Polozov, A., Meek, C., Awadallah, A.H.: Score: Pre-training for context representation in conversational semantic parsing. In: International Conference on Learning Representations (2021)
96. Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., et al.: Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3911–3921 (2018)
97. Zeng, A., Liu, X., Du, Z., Wang, Z., Lai, H., Ding, M., Yang, Z., Xu, Y., Zheng, W., Xia, X., et al.: Glm-130b: An open bilingual pre-trained model. arXiv preprint arXiv:2210.02414 (2022)
98. Zenz, G., Zhou, X., Minack, E., Siberski, W., Nejdl, W.: From keywords to semantic queries-incremental query construction on the semantic web. J. Web Semant. **7**(3), 166–176 (2009)
99. Zeyer, A., Bahar, P., Irie, K., Schlüter, R., Ney, H.: A comparison of transformer and lstm encoder decoder models for asr. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 8–15. IEEE (2019)
100. Zhang, R., Yu, T., Er, H., Shim, S., Xue, E., Lin, X.V., Shi, T., Xiong, C., Socher, R., Radev, D.: Editing-based sql query generation for cross-domain context-dependent questions. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 5338–5349 (2019)
101. Zhao, X., Wang, L., He, R., Yang, T., Chang, J., Wang, R.: Multiple knowledge syncretic transformer for natural dialogue generation. In: Proceedings of The Web Conference 2020, pp. 752–762 (2020)
102. Zheng, W., Cheng, H., Zou, L., Yu, J.X., Zhao, K.: Natural language question/answering: Let users talk with the knowledge graph. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 217–226 (2017)
103. Zhong, V., Xiong, C., Socher, R.: Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103 (2017)
104. Zhou, S., Dong, L., Xu, S., Xu, B.: A comparison of modeling units in sequence-to-sequence speech recognition with the transformer on mandarin chinese. In: International Conference on Neural Information Processing, pp. 210–220. Springer (2018)
105. Zhou, X., Chai, C., Li, G., Sun, J.: Database meets artificial intelligence: A survey. IEEE Trans. Knowl. Data Eng. (2020)