Demonstration of SpeakQL: Speech-driven Multimodal Querying of Structured Data

Vraj Shah Side Li Kevin Yang Arun Kumar Lawrence Saul
University of California, San Diego
{vps002,s7li,khy009,arunkk,saul}@eng.ucsd.edu

ABSTRACT

In this demonstration, we present SpeakQL, a speech-driven query system and interface for structured data. SpeakQL supports a tractable and practically useful subset of regular SQL, allowing users to query in any domain with unbounded vocabulary with the help of speech/touch based user-in-the-loop mechanisms for correction. When querying in such domains, automatic speech recognition introduces countless forms of errors in transcriptions, presenting us with a technical challenge. We characterize such errors and leverage our observations along with SQL's unambiguous context-free grammar to first correct the query structure. We then exploit phonetic representation of the queried database to identify the correct Literals, hence delivering the corrected transcribed query. In this demo, we show that SpeakQL helps users reduce time and effort in specifying SQL queries significantly. In addition, we show that SpeakQL, unlike Natural Language Interfaces and conversational assistants, allows users to query over any arbitrary database schema. We allow the audience to explore SpeakQL using an easy-to-use web-based interface to compose SQL queries.

ACM Reference Format:
Vraj Shah Side Li Kevin Yang Arun Kumar Lawrence Saul. 2019. Demonstration of SpeakQL: Speech-driven Multimodal Querying of Structured Data. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3299869.3320224

1 INTRODUCTION

Automatic Speech Recognition (ASR) is the key enabler of many speech-driven applications on emerging device environments such as smartphones and tablets and even personal conversational assistants such as Siri, Cortana, and Alexa. The recent success of speech-driven interfaces has motivated an important fundamental question: *How should a speech-driven system be designed to query structured data?*

There has been a stream of research on studying new querying modalities like touch-oriented [2], visual [7] and natural language interfaces (NLIs) [3, 4], especially for constrained querying environments such as smartphones, tablets, and conversational assistants. At the user level, almost all of these query interfaces obviates the need to specify SQL. However, what is missing from the prior work is a speech-driven interface for regular SQL or other structured querying.

One might wonder: *Why dictate SQL and not just use NLIs or visual interfaces?* Existing typed or spoken NLIs are restricted by the capability of the natural language parsing tools to identify the semantics of the natural language. The ambiguity of human language often causes query misunderstanding. The lack of well-defined semantics for correct answers further aggravate this situation. Conversational assistants allow users to query curated app-specific datasets, and not any arbitrary database schemas. Hence, NLIs compromise query sophistication in order achieve high usability as shown in Figure1(A) [1]. As a result, current NLIs are imposing more structure to mitigate the AI-hard natural language understanding (NLU) problem [3], as shown by Figure 1(B). They are moving to the right towards the "cliff" of ambiguity, where on the right hand side of the cliff, we have languages with unambiguous context-free grammars (CFG).

On the other hand, SQL is already a structured English query language, which provides high query sophistication and less ambiguity due to its CFG and succinct nature. Structured data querying is ubiquitous in many domains such as enterprise, C-suite, and healthcare. Typing queries in SQL is the gold standard for such querying. However, typing SQL is painful in the constrained environments. A spoken SQL interface could speed up the query specification process. In this work, we aim to achieve SQL-level query sophistication, while also improving ease of use, as shown in Figure1(A). This would allow users to interact with the structured data using spoken queries over any arbitrary database schema. Thus, instead of forcing all users to only use NLIs, we leverage ASR for SQL to make spoken querying more "natural" without losing the sophistication of unambiguous CFGs.

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-5643-5/19/06...$15.00
https://doi.org/10.1145/3299869.3320224

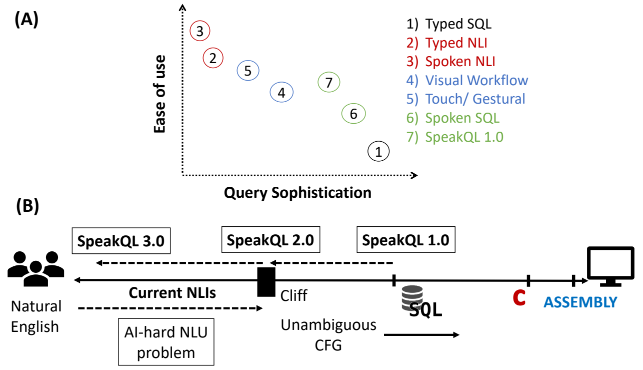


Figure 1: (A) Qualitative comparison of trade-off between ease of use and currently feasible query sophistication. (B) Contrasting SpeakQL's goals and current NLIs in terms of the "naturalness" of query language.

In this demo, as the first major step in our vision, we deliver the first spoken querying system for a subset of SQL. We call our system SpeakQL 1.0. Complementary to existing NLIs, visual and touch-oriented interfaces, SpeakQL offers the following functionalities. First, SpeakQL allow users to query structured data using regular SQL with a tractable subset of the CFG grammar. Second, and most importantly, SpeakQL allow users to query in any application domain over any database schema. Finally, SpeakQL allows users to perform multimodal interactive query correction using speech and touch with a screen display. In summary, SpeakQL is an open-domain, speech-driven, and multimodal querying system for regular SQL that allows users to dictate the query and perform interactive correction using speech/touch.

From an ASR's perspective, unlike regular English speech, SQL speech, presents several interesting and novel challenges. There exist infinite kinds of literal instances, such as `oid_123A` in SQL. ASR might split a single token from SQL's standpoint into multiple tokens. Hence, ASR fails to transcribe such tokens correctly, since they are simply not present in their vocabulary. Such non-vocabulary tokens occur more in SQL than natural English. We call this the *unbounded vocabulary problem*, and it is a key technical challenge for SpeakQL. Note that this problem has not been solved even for spoken NLIs. For example, every time a non-vocabulary token is spoken, Cortana returns "Sorry, I didn't get that". Hence, addressing this problem benefits spoken NLIs too. Moreover, ASR might fail to generate correct transcription even for in-vocabulary homophonic tokens, e.g., "`men`" vs. "`min.`" These myriad kind of transcription errors present SpeakQL with yet another technical challenge.
Relationship to prior work. Speech recognition for data querying has been explored in some prior systems. Nuance's Dragon Naturally Speaking allows users to query using spoken commands to retrieve the text content of a document [6]. However, to the best of our knowledge, there does not exist a query interface for spoken SQL. The recent system Echoquery [4] is designed as a conversational NLI in form of an Alexa skill. Although, this system certainly enables non-experts to query data easily, ASR can cause a series of errors and would restrict users from specifying "hard" queries. In addition, such a system might impose a higher cognitive load on users when a large query result is returned; a screen mitigates such issues, e.g., as in the Echo Show. Moreover, a recent user study [5] on a text messaging app conducted by Baidu, showed that the input rate is significantly faster when users uses speech to perform only the first dictation, and then errors are corrected and refined through touch. SpeakQL's novel multimodal query interface allows users to easily mix speech-driven query specification with speech-driven or touch-driven interactive query correction.

2 SYSTEM ARCHITECTURE

In recent years, deep neural networks have led to significant advances in ASR engines, making them powerful enough for many real-world applications. To reduce the need of additional engineering effort in developing a SQL-specific ASR, we instead use an existing ASR technology and focus solely on issues related to SQL as desribed below.

First, in SQL, there exist only three types of tokens: *Keywords*, *Special Characters* ("SplChar"), and *Literals*. SQL Keywords such as `SELECT` and `FROM`, and SplChars such as `*` and `=` have a finite vocabulary of tokens, that occurs only from the SQL grammar[1]. A literal can be a table name, an attribute name or an attribute value. Table names and attribute names have a finite vocabulary. However, the attribute value can be any value from the database or any generic value. Hence, the domain size of the Literals is likely infinite.

Second, the ASR engine has some interesting points of failure when transcribing SQL. The ASR may convert Literals into SplChars or Keywords and vice versa when tasked with homophones. For instance, SQL Keyword `min` detected as "men." Even a single-token transcription might be completely wrong because the token is simply not present in the ASR's vocabulary. Even worse, ASR might split a non-vocabulary token like `oid_123A` into a series of tokens in the transcription output, possibly intermixed with Keywords and SplChars.

Based on these observations, we propose to segregate the task of structure determination from the task of literal determination. Figure 2 shows the complete four-component end-to-end system. *This decoupling of structure and literal determination is a critical design decision that helps us handle the problem of unbounded vocabulary.* SpeakQL has four

[1] http://forcedotcom.github.io/phoenix

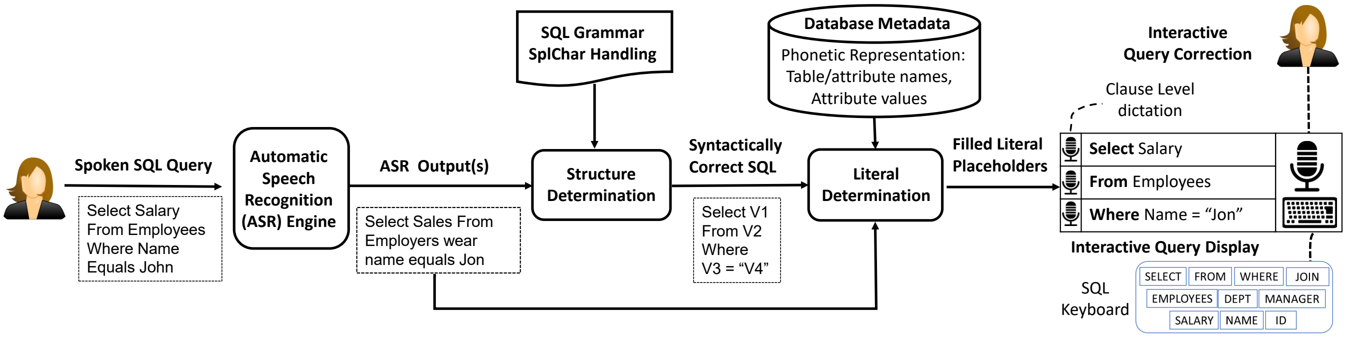

Spoken SQL Query
Select Salary From Employees Where Name Equals John
Automatic Speech Recognition (ASR) Engine
ASR Output(s)
Select Sales From Employers wear name equals Jon
SQL Grammar SplChar Handling
Structure Determination
Syntactically Correct SQL
Select V1 From V2 Where V3 = "V4"
Database Metadata
Phonetic Representation: Table/attribute names, Attribute values
Literal Determination
Filled Literal Placeholders
Interactive Query Correction
Clause Level dictation
Select Salary
From Employees
Where Name = "Jon"
Interactive Query Display
SQL Keyboard
SELECT FROM WHERE JOIN
EMPLOYEES DEPT MANAGER
SALARY NAME ID

Figure 2: End-to-end Architecture of SpeakQL. We show an example of a simple spoken SQL query, and how it gets converted to a query displayed on a screen, which the user can correct interactively.

major components: ASR Engine, Structure Determination module, Literal Determination module and Interactive Display module. We describe each component below.

ASR Engine. This component transcribes the recorded spoken SQL query to a textual output. A modern speech recognition system consists of two major components: acoustic model and the language model. The acoustic model represents how sounds combine to produce words and the language model represents how words combine to produce utterances. The language model essentially captures the vocabulary that the application is likely to use. We utilize Azure's Custom Speech Service[2] to create a custom language model by training on a dataset of spoken SQL queries. For the acoustic model, we use Microsoft's state-of-the-art search and dictation model. Figure 2 shows a dictated SQL query. The transcription obtained by ASR engine could be `select sales from employers wear first name equals Jon`.

Structure Determination. This component converts the ASR transcription to a syntactically correct SQL structure. SQL structure fixes the Keywords and SplChars, while Literals are masked out with numbered placeholder variables. We leverage CFG of our currently supported subset of SQL to obtain all possible ground truth structures. A ground truth structure is a syntactically correct SQL string obtained from our SQL grammar by applying the production rules recursively. We obtain the closest matching structure by doing a similarity search based on edit distances with ground truth structures. The detected structure in our running example is `Select x1 From x2 Where x3 = x4`. Here, the Literals are represented as placeholder items `x1`, `x2`, `x3` and `x4`.

Literal Determination. The Literal Determination component determines a ranked list of Literals for each placeholder variable using both the raw ASR output and a pre-computed phonetic representation of the database being queried. The phonetic representation of the existing Literals in the database helps us disambiguate the words from transcription output that sounds similar. For instance, variable `x1` is replaced as a top k list of attribute names. Phonetically, among all the attribute names, `Salary` is the closest to `Sales`, and thus, `x1` would be bound to `Salary`.

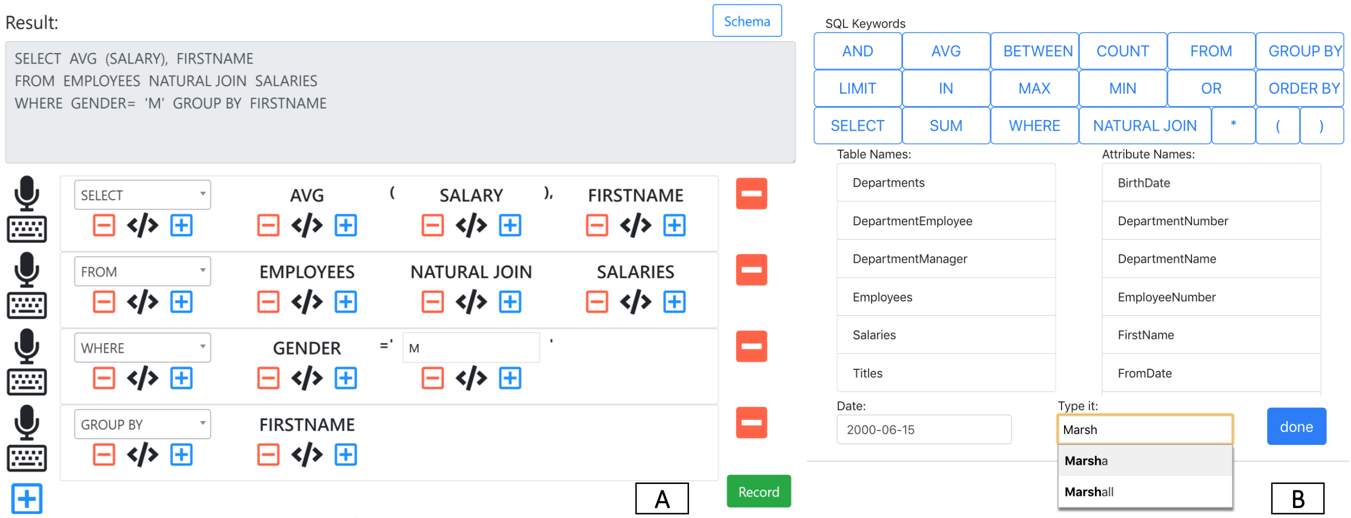
Interactive Display. This component displays the best possible transcription generated by SpeakQL. Even with our query determination algorithms, there is still a possibility that some tokens in the transcription are incorrect, particularly for Literals not present in ASR vocabulary. Hence, this component allows user to perform interactive query correction through speech and/or touch. To correct the displayed query, the user can dictate/re-dictate queries at the clause level or use the novel SQL keyboard tailored to reduce their effort in correcting the displayed query.

3 DEMONSTRATION

Data. In the demonstration, we plan to show the use of SpeakQL on two real world database schemas, the Employees Sample Database from MySQL (dev.mysql.com/doc/employee/en/) and the Yelp Database (kaggle.com/yelp-dataset). Note that ASR is trained on spoken SQL queries from the Employees database. Hence, querying on the Yelp database will test the generalizabilty of SpeakQL to new database schemas. This will showcase that SpeakQL can be utilized to query any arbitrary database schema.

Query Set. The audience will dictate several SQL queries using an interactive web interface on a Samsung Tablet. Two tablets would be available at the demo. The queries will be equally divided into two segments: *simple* and *complex*. *Simple* queries have less than 20 tokens; the rest are considered *complex*. Thus, composing a *complex* query imposes a higher cognitive load relative to a simple query. Natural language description of the query along with the database schema will be provided to the participant. For the demonstration purpose, the interface would include 6 such queries. The queries includes most Select-Project-Join-Aggregation (SPJA) queries along with `ORDER BY` and `LIMIT`, without any restrictions on the number of joins, aggregates or predicates.

[2] westus.cris.ai/Home/CustomSpeech



```
SELECT  AVG  (SALARY),  FIRSTNAME
FROM  EMPLOYEES  NATURAL JOIN  SALARIES
WHERE  GENDER=  'M'  GROUP BY  FIRSTNAME
```

Figure 3: SpeakQL Interface. (A) The Interactive Display showing the dictated query after being processed by the SpeakQL engine, as well as the touch-based editing functionalities and clause-level redictation capabilty. (B) Our simple SQL keyboard designed for touch-based editing of the rendered query string.

This will demonstrate that SpeakQL can effectively handle meaningful and practically useful subset of regular SQL Data Manipulation Language (DML).

Walkthrough. Each participant will be first made familiar with the database schema and SpeakQL interface through an introductory video[3]. The introductory video covers a complete walkthrough of the interface and allow users to learn about several functionalities of the system. The participant would interact with the SpeakQL interface shown in Figure 3 (A). The user can dictate the entire query in one go using the "Record" button at the bottom right. At the same time, the user can dictate or correct (through re-dictation) the queries at the clause level using record button to the left of each clause. For instance, the user can choose to dictate only the `FROM` clause or `WHERE` clause. +/− buttons allows users to insert/delete Keywords and Literals from the query. The notable </> button allows for a quick insertion or removal of SplChars. If the displayed literal is incorrect, the user can touch its box and a drop-down menu will display the ranked lists of alternatives for that literal.

Figure 3 (B) shows the novel "SQL Keyboard" that consists of the entire lists of SQL Keywords, table names, and attribute names. Since the attribute values (including dates) can be potentially infinite, they cannot be seen in a list view. But the user can type with the help of an auto complete feature. Dates can be specified easily with the help of a scrollable date picker. Such keyboard design allows for a quick in-place editing of stray incorrect tokens, present anywhere in the SQL query string. In the worst case, if SpeakQL fails to identify the correct query structure and/or Literals, the user can type one token, multiple tokens, or the whole query from scratch in the query display box, or redictate the clauses or the whole query again. Finally, the user can run the correct query over the underlying RDBMS with the help of a "Run Query" button and see the query results on the screen.

Leaderboard Competition. In order to make this demonstration more fun, we plan to run a competition that tests audience's proficiency in speaking SQL. The users that are interested in participating will dictate a single complex SQL query using the SpeakQL interface. We will maintain a leaderboard of top participants who finished the query in the shortest amount of time. The winner and runner-up will be rewarded with gift cards.

Acknowledgments. This work was supported in part by the National Science Foundation under grant IIS-1816701. All findings and opinions expressed in this work are those of the authors and do not necessarily reflect the views of the NSF. We thank the members of UC San Diego's Database Lab for their feedback on this work.

REFERENCES

[1] D. Chandarana et al. 2017. SpeakQL: Towards Speech-driven Multimodal Querying. *HILDA* (2017).
[2] A. Crotty et al. 2015. Vizdom: Interactive Analytics Through Pen and Touch. *PVLDB* (2015).
[3] F. Li et al. 2014. Constructing an interactive natural language interface for relational databases. *PVLDB* (2014).
[4] G. Lyons et al. 2016. Making the case for query-by-voice with echoquery. *SIGMOD* (2016).
[5] S. Ruan et al. 2016. Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices. *CoRR* (2016).
[6] https://www.nuance.com/dragon.html. 2018. Nuance's Dragon Speech Recognition.
[7] M. Zloof et al. 1975. Query by Example. *National Computer Conference and Exposition* (1975).

[3] https://vimeo.com/295693078