INFO 290T

Human-Centered Data Management

SpeakQL

OK, so…

- We've talked about various ways to express SQL queries
  - Visual query builders
  - GestureDB
  - Query by Example

- Onto speech!

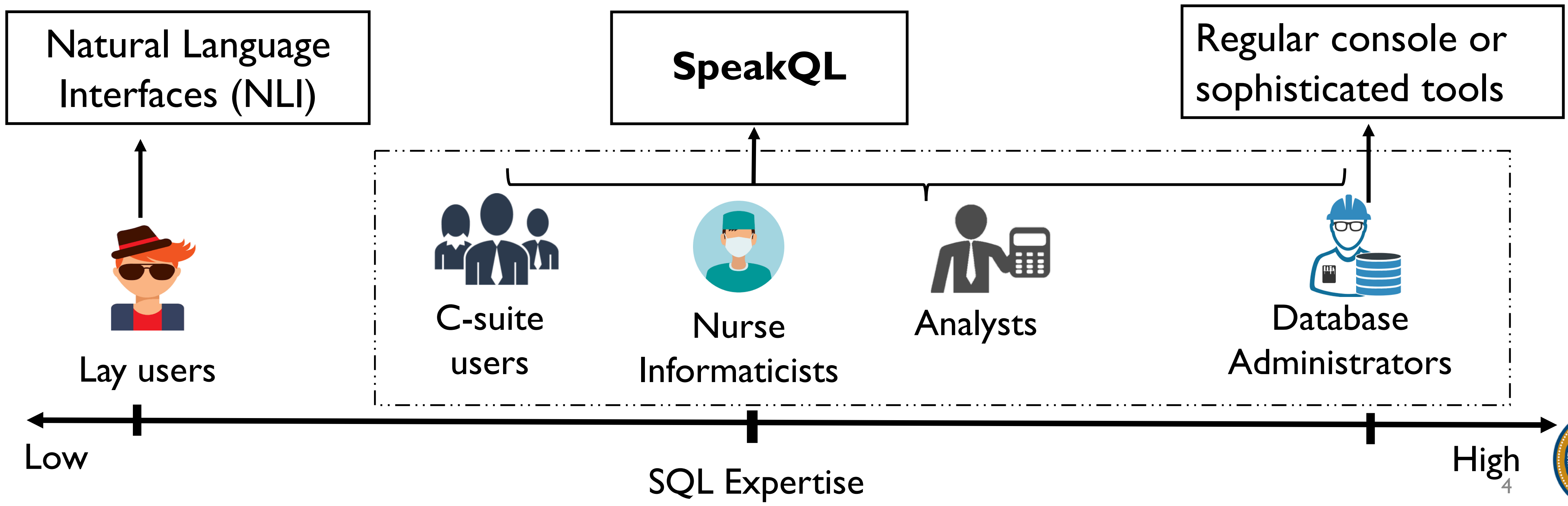
Why is Speech to SQL interesting?

- Technology trends:
  - Speech interfaces are becoming more popular
    - Think cell phones, smart speaker devices, audio in computers
    - Speech-based assistants: Alexa, Siri, Cortana, …
  - Automatic Speech Recognition (ASR) progress enough to understand speech

- Human factors:
  - (Not a focus of the paper) Accessibility
    - "A rising tide raises all boats" – accessible interfaces help disabled people as well as older people, people in rural areas, … plus also folks without disabilities!
  - Preferred by folks "on the go" or those who prefer speech to typing

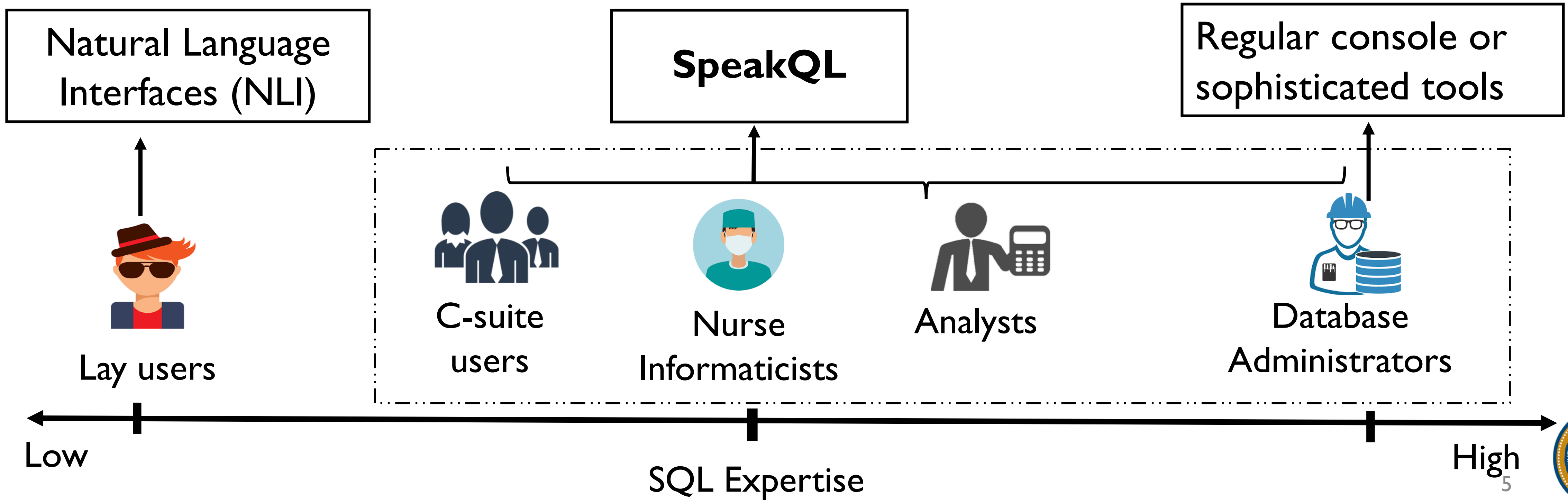
Why SQL as a target?

- SQL is still the primary means of interacting with data
- SQL expertise of users differ – some are more likely to want to use a speech based-interface



Why SQL as a target?

- We focus on the setting where the user does know SQL, but wants to express it via speech
  - Could potentially be extended to fuzzy settings but not our focus
  - This is the focus of NLIs



Formative Interview Study

- 26 SQL users from 17 sectors
- SQL users care about:
  - Support for ad-hoc queries
  - Unambiguous responses
  - Anytime and anywhere access to data
- Typing on tablets/smartphones is painful!
- *Can we dictate queries based on speech and touch capabilities of such platforms?*

- Sidebar: this study may potentially have been a bit biased towards the fact that they wanted to showcase a need for their tool rather than them trying to solve the most pressing need

Why not focus on truly fuzzy queries? Why SQL?

- SQL has several advantages that professionals find useful:
  - Sophisticated query support
  - Unambiguous grammar & succinctness
  - Guarantee of result correctness
- If we accept truly fuzzy queries, we would:
  - Need to deal with challenges understanding user intent
    - General natural language understanding problem: AI-hard!
- With training, people can get really good at SQL: and if so, why not empower them to use SQL on the go?

SpeakQL System Overview

SpeakQL is an open-domain, speech-driven multimodal system that

- Supports a subset of regular SQL DML
- Uses an existing ASR and focuses on SQL-specific issues
- Supports any database schema in any application domain
- Supports interactive multimodal query correction with screen display

SpeakQL Challenges

1. Heterogeneity of failures for ASR

2. Unbounded vocabulary problem

e.g. "*CUSTID_1729A*" split into multiple tokens or not recognized at all

Siri: "*I'm sorry, I don't understand the question*"

3. Real-time efficiency

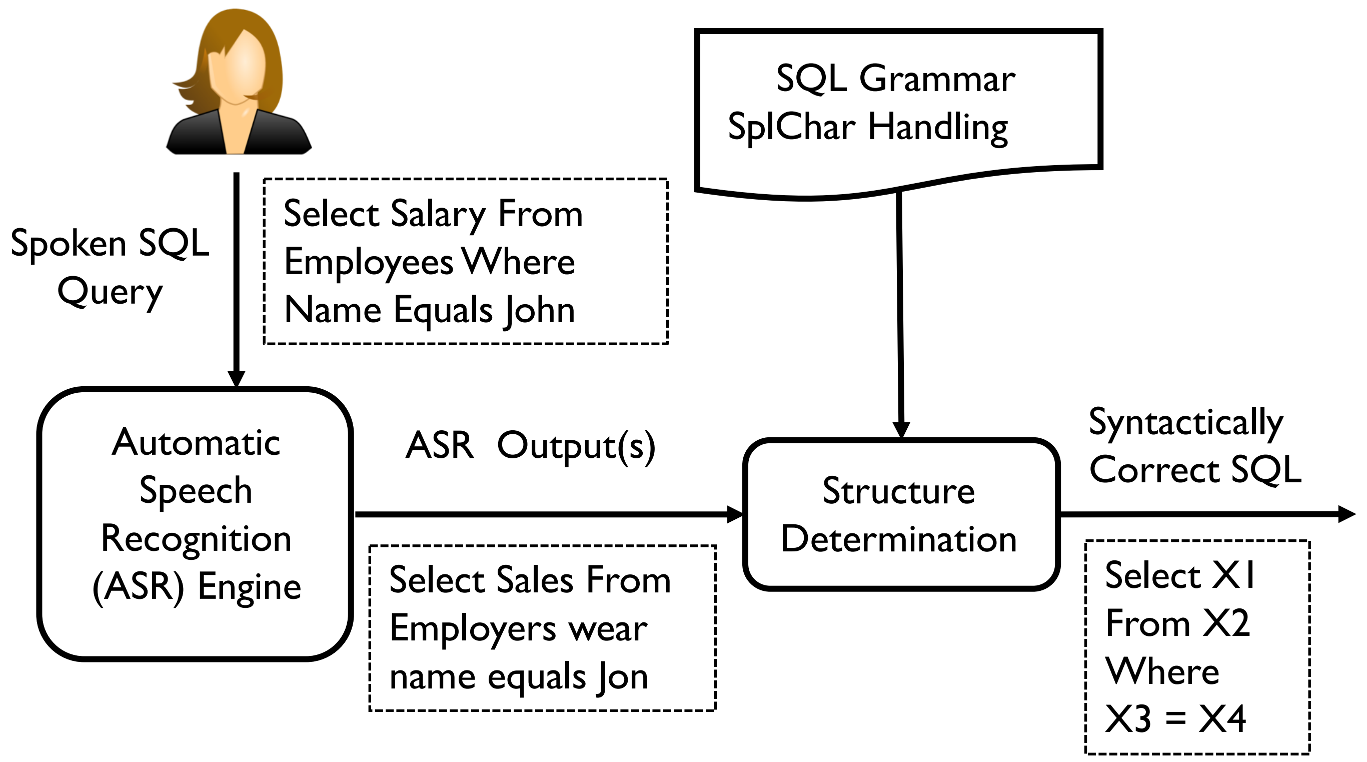
Type of Errors	**Ground truth token**	**ASR transcription**
Homophony (Keywords to Literals)	sum	some
Homophony (Literals to Keywords)	fromdate	from date
Splitting of numbers into multiple tokens	45412	45000 412
Erroneously transcribed dates	1991-05-07	may 07 90 91
Unbounded vocabulary for Literals	CUSTID_1729A	custody _ 1 7 2 9 8
	oid_73fc	or ID _ 7 3 f c

Key Insight: Decomposition of Problem

- SQL has Keywords, Special Characters (SplChar) & Literals (Attr Names, Values, …)
- Determining everything at one go is problematic
- *What if we first determine the structure using Keywords and SplChars, then "fill in" the Literals?*

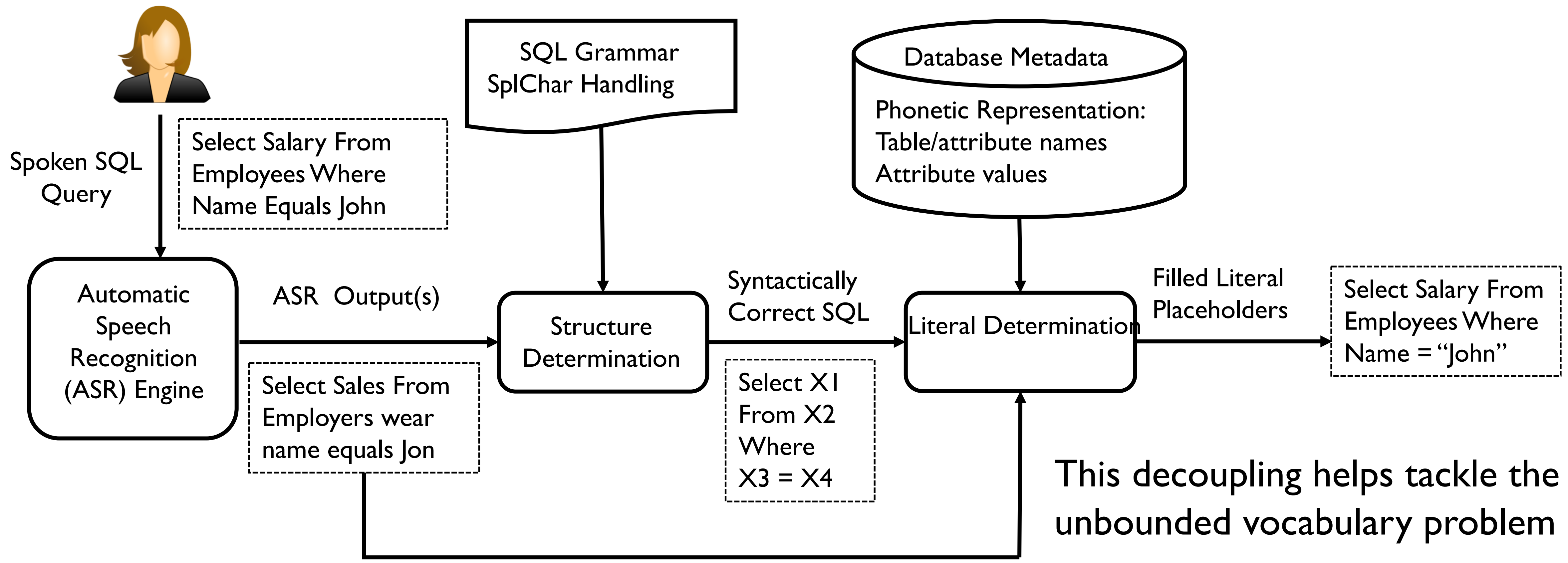
Type of Errors	Ground truth token	ASR transcription
Homophony (Keywords to Literals)	sum	some
Homophony (Literals to Keywords)	fromdate	from date
Splitting of numbers into multiple tokens	45412	45000 412
Erroneously transcribed dates	1991-05-07	may 07 90 91
Unbounded vocabulary for Literals	CUSTID_1729A	custody _ 1 7 2 9 8
	oid_73fc	or ID _ 7 3 f c

SpeakQL: System Architecture


Spoken SQL Query
Select Salary From Employees Where Name Equals John
SQL Grammar SplChar Handling
Automatic Speech Recognition (ASR) Engine
ASR Output(s)
Select Sales From Employers wear name equals Jon
Structure Determination
Syntactically Correct SQL
Select X1 From X2 Where X3 = X4

Structure Determination. Leverage rich structure of SQL (unambiguous CFG) to get correctly recognized Keywords + Special Characters

SpeakQL: System Architecture

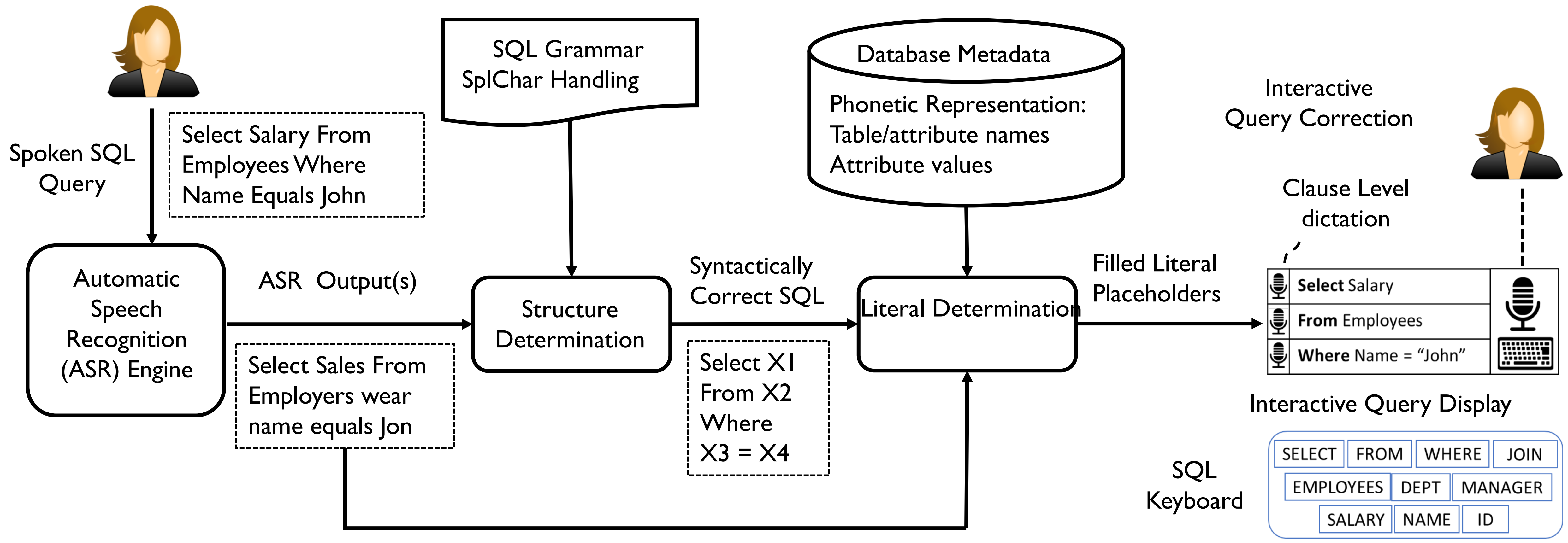


Spoken SQL Query
Select Salary From Employees Where Name Equals John
SQL Grammar SplChar Handling
Database Metadata
Phonetic Representation: Table/attribute names Attribute values
Automatic Speech Recognition (ASR) Engine
ASR Output(s)
Select Sales From Employers wear name equals Jon
Structure Determination
Syntactically Correct SQL
Select X1 From X2 Where X3 = X4
Literal Determination
Filled Literal Placeholders
Select Salary From Employees Where Name = "John"

This decoupling helps tackle the unbounded vocabulary problem

Literal Determination. Leverage phonetic representation of database instances to fill in literals using voting algorithm

SpeakQL: System Architecture


Spoken SQL Query
Select Salary From Employees Where Name Equals John
SQL Grammar SplChar Handling
Database Metadata
Phonetic Representation: Table/attribute names Attribute values
Interactive Query Correction
Clause Level dictation
Automatic Speech Recognition (ASR) Engine
ASR Output(s)
Structure Determination
Syntactically Correct SQL
Literal Determination
Filled Literal Placeholders
Select Salary
From Employees
Where Name = "John"
Select Sales From Employers wear name equals Jon
Select X1 From X2 Where X3 = X4
Interactive Query Display
SQL Keyboard
SELECT
FROM
WHERE
JOIN
EMPLOYEES
DEPT
MANAGER
SALARY
NAME
ID

Interactive Display. User-in-the-loop query correction through speech or touch/click

Demo

- https://vimeo.com/295693078

Flow of Steps

- ASR
- Structure Determination
- Literal Identification
- Interactive Correction

ASR

- Custom language model: Azure's Custom Speech Service
- Trained on a dataset of spoken SQL queries

- May end up with queries like
  - "select sales from employers wear first name equals Jon"

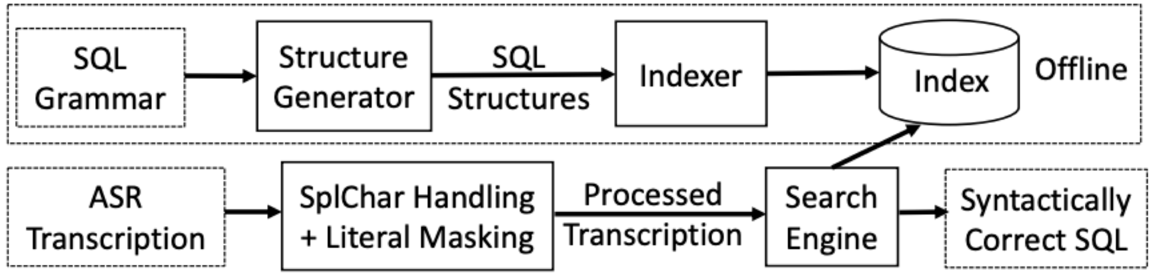
Structure Determination

- Goal: obtain syntactically correct SQL with placeholders for literals
  - E.g., select x1 from x2 where x3 = x4

- Similarity search algorithm on a weighted edit distance metric that leverages the SQL CFG

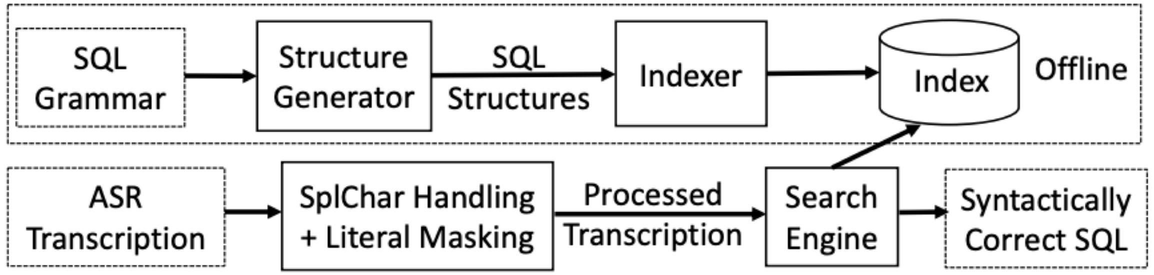
- Focus: Select-Project-Join-Aggregate, with Limits and Order.
- Keywords: SELECT, FROM, …
- SplChars: < , *) > . …

Structure Determination II

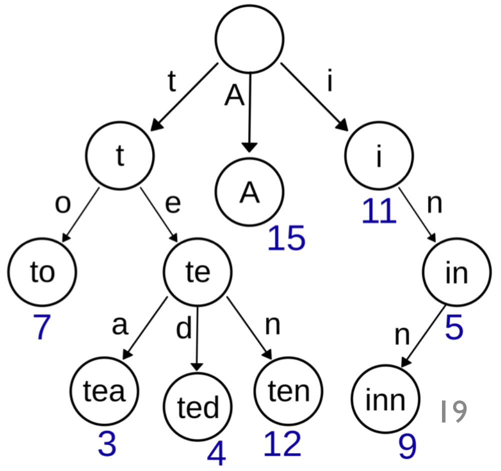


- SplChars identified are replaced with operators, e.g., "less than" to "<"
- Anything not matching a keyword or SplChar is replaced by a placeholder
  - select sales from employers wear first name equals Jon - to
  - SELECT x1 FROM x2 x3 x4 = x5
- This is then fed to the search engine that looks up a precomputed index

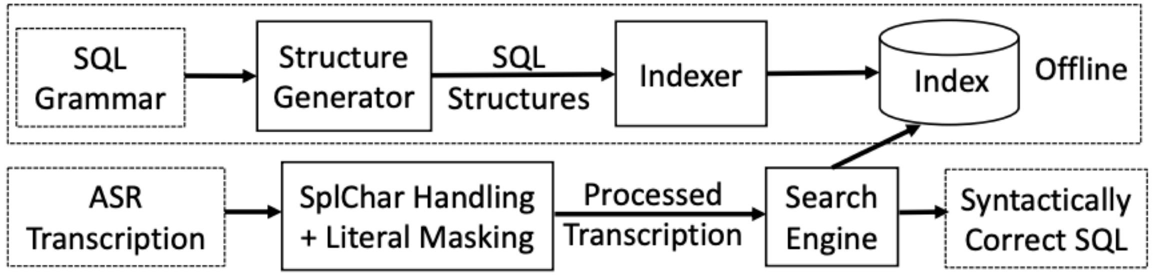
Structure Determination II



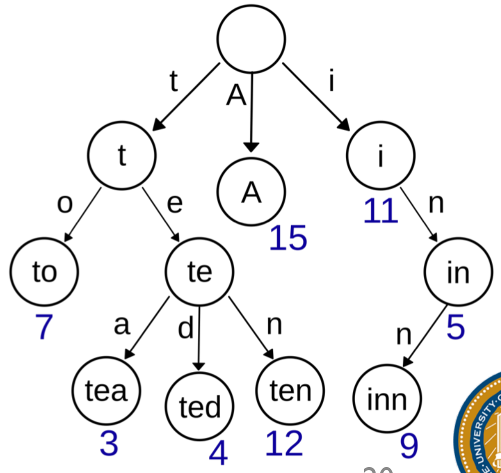
- Offline index construction to look up queries like:
  - SELECT x1 FROM x2 x3 x4 = x5
- Compute all SQL queries of up to length 50 tokens (= 1.6M queries)
- But searching each one is too expensive
- Thankfully, lots of redundancy!
  - Many strings share prefixes
- Store 1 trie per length of query
  - Search per trie
  - (Sidebar: Not entirely clear why one trie per length is needed)



Structure Determination II



- Dynamic programming approach
  - Find lowest edit distance for prefix of ASR query + prefix of each indexed SQL query
    - Store this as part of each "node" in your trie as you're doing the search
  - Natural substructure here
- Some additional nuances
  - Different weights for keywords, …
  - Some optimization/pruning techniques

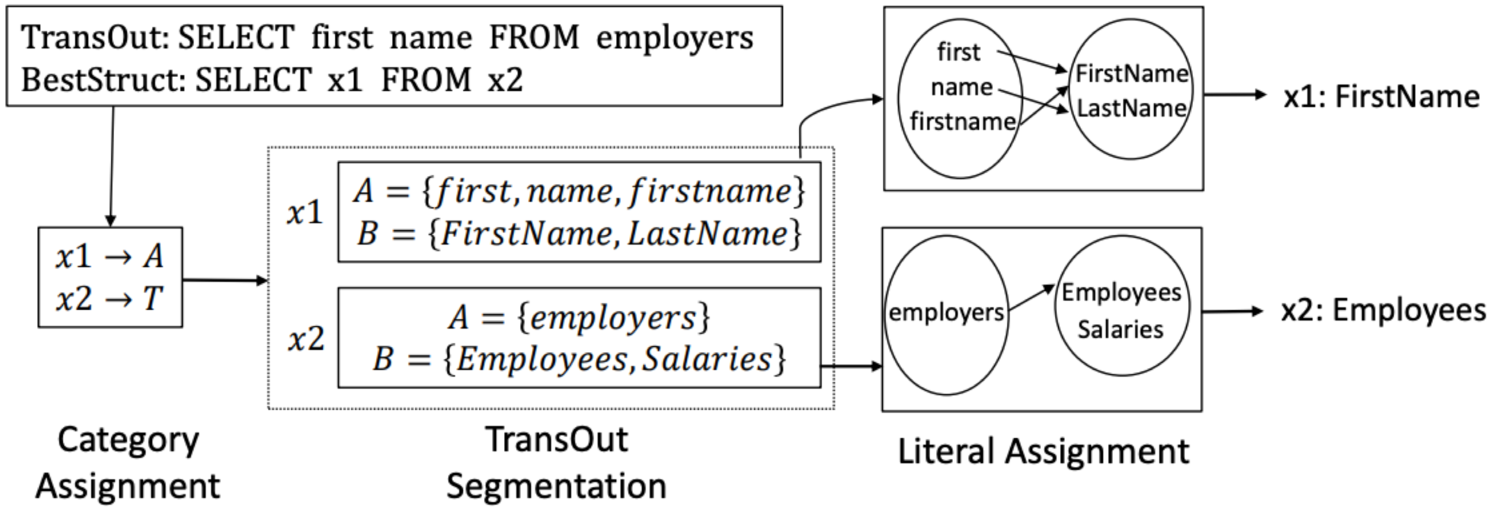


Literal Identification

- Finds a ranked list of literals for each placeholder using the raw ASR output and phonetic representations of the potential literals in the database
- To get the phonetic representation for literals, we use Metaphone, a phonetic algorithm

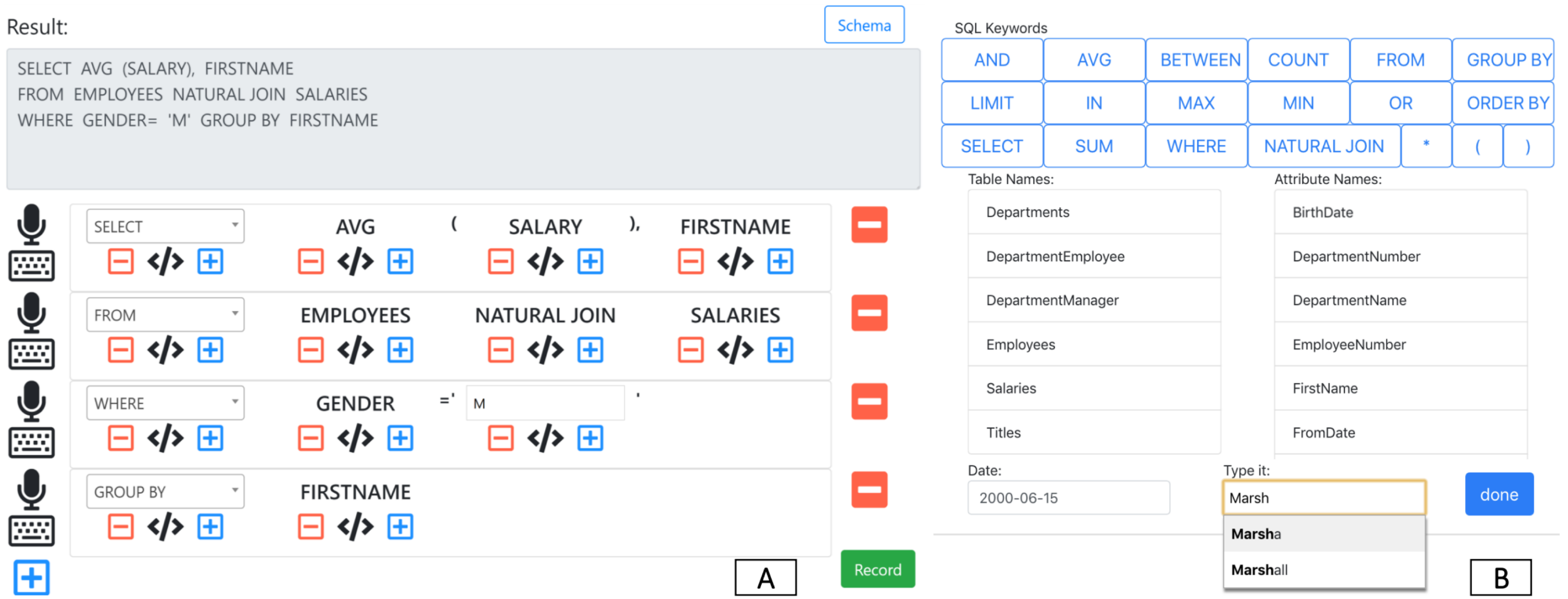
Literal Identification

- Restricts to potential candidates (attribute or table names)
- Edit distance between each spoken phrase and candidate
- Each candidate gets a "vote"
- Most voted on candidate wins!



Interactive Display

- Provides a single SQL query
- Can be fixed via speech and touch-based mechanisms
    - Ambiguity widgets!
    - Can re-dictate clauses or use a SQL-centric keyboard



Result:

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

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

Table Names: Departments, DepartmentEmployee, DepartmentManager, Employees, Salaries, Titles

Attribute Names: BirthDate, DepartmentNumber, DepartmentName, EmployeeNumber, FirstName, FromDate

Date: 2000-06-15

Type it: Marsh

Marsha

Marshall

Sidebar: Contrast to Datatone

- Q: How does this approach (structure $\rightarrow$ literals) compare to the approach in Datatone?

Sidebar: Contrast to Datatone

- Q: How does this approach (structure → literals) compare to the approach in Datatone?

- It’s actually the opposite approach!
- Datatone is bottom up, identifying attributes, filters, then determining structure
- SpeakQL is top down, determining structure, then identifying attributes, filters

Evaluation: Dataset & ASR

Existing datasets are for NLI, not for this problem…

New Dataset for Spoken SQL. Using data generation procedure that is scalable and applicable to arbitrary schema

1. Generated 1250 textual SQL queries on Employees Database using CFG

2. Used Amazon Polly to obtain spoken SQL queries

For ASR: Azure's Custom Speech API; (Training: 750, Testing: 500)

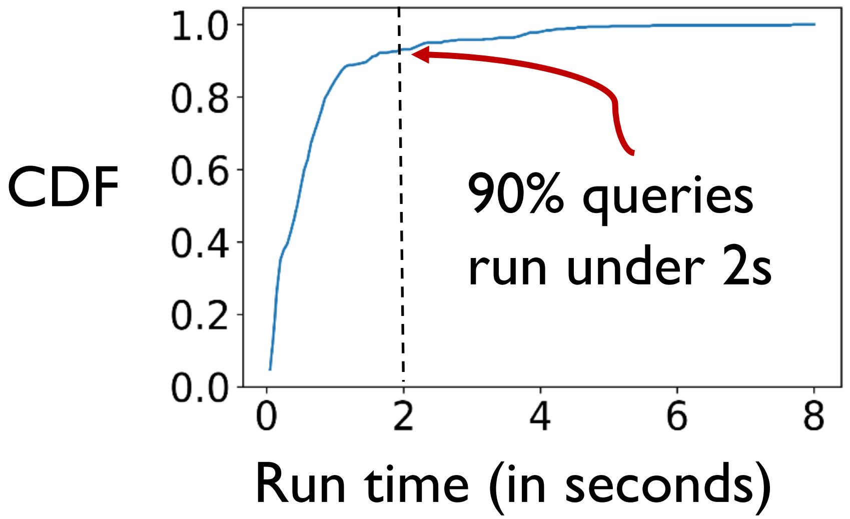
Evaluation: End-to-End Results

Accuracy by F1 Score

	Keywords	Special Characters	Literals	Word
ASR	0.88	0.91	0.51	0.7
SpeakQL	0.98	0.98	0.82	0.9

Takeaway: SpeakQL corrects large fraction of errors over ASR

Latency



Takeaway: SpeakQL achieves near real-time latency

Evaluation: User Study

Setup. 15 participants familiar with SQL; Each composed 12 queries on tablet

Query Set. Select-Project-Join-Aggregate queries

Tasks. (1) Use SpeakQL interface vs (2) Raw typing

Results.

	Speedup in Time
Mean	2.7x
Max	6.7x

Takeaway: SpeakQL help users speedup SQL specification time significantly

Summary

First end-to-end speech-driven multimodal system for querying with SQL

Released the first dataset of spoken SQL queries

SpeakQL reduces query specification time (2.7x avg speedup) and effort (10x avg reduction) compared to raw typing

Thoughts on paper?

- Writing?
- Approach?
- Evaluation?

Announcements

- Please talk to us to get feedback on projects!
- Alternate class time proposal