Disambiguating Natural Language Queries with Tuples

Christopher Baik

Zhongjun (Mark) Jin

Michael Cafarella

University of Michigan – Ann Arbor

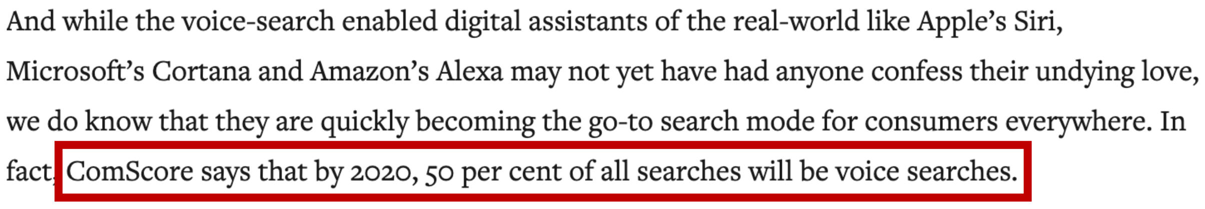
Overview

- Motivation
- Approach
- Experiments
- Conclusion

Motivation

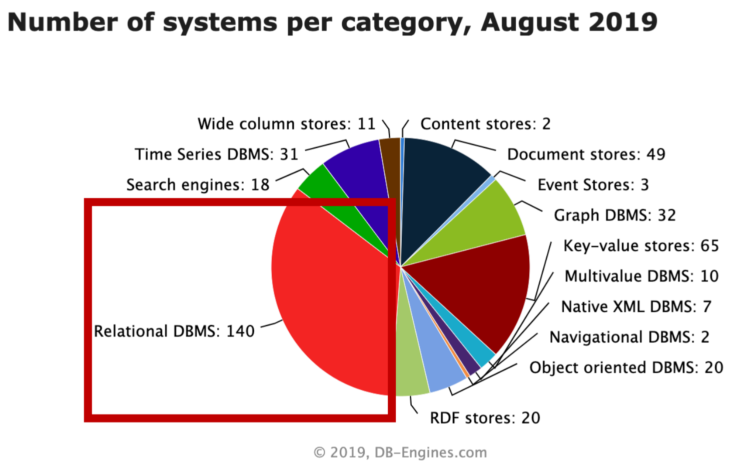
Background

People want to search by talking



And while the voice-search enabled digital assistants of the real-world like Apple's Siri, Microsoft's Cortana and Amazon's Alexa may not yet have had anyone confess their undying love, we do know that they are quickly becoming the go-to search mode for consumers everywhere. In fact, ComScore says that by 2020, 50 per cent of all searches will be voice searches.

Relational databases are common



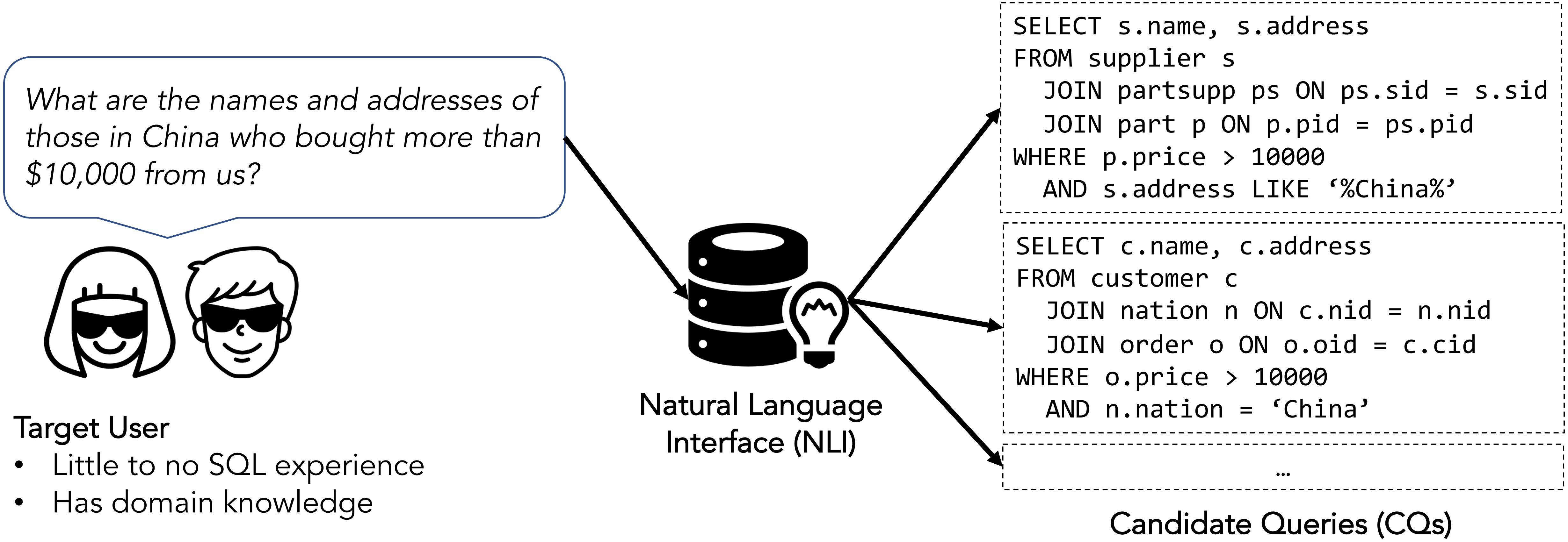
Number of systems per category, August 2019

© 2019, DB-Engines.com

Let's build natural language interfaces for databases!

Sources
https://www.campaignlive.co.uk/article/just-say-it-future-search-voice-personal-digital-assistants/1392459
https://db-engines.com/en/ranking_categories

Problem: Natural language is ambiguous



What are the names and addresses of those in China who bought more than $10,000 from us?

Target User

- Little to no SQL experience
- Has domain knowledge

Natural Language Interface (NLI)

```
SELECT s.name, s.address
FROM supplier s
  JOIN partsupp ps ON ps.sid = s.sid
  JOIN part p ON p.pid = ps.pid
WHERE p.price > 10000
  AND s.address LIKE ‘%China%’
```

```
SELECT c.name, c.address
FROM customer c
  JOIN nation n ON c.nid = n.nid
  JOIN order o ON o.oid = c.cid
WHERE o.price > 10000
  AND n.nation = ‘China’
```

…

Candidate Queries (CQs)

Database queries require more precision than typical speech

Solving the Precision Challenge

- One-shot: improve NL-to-SQL models
  - Lots of recent work in this area
- Iterative: provide clarification and refining mechanisms

Previous Approaches to NL Clarification

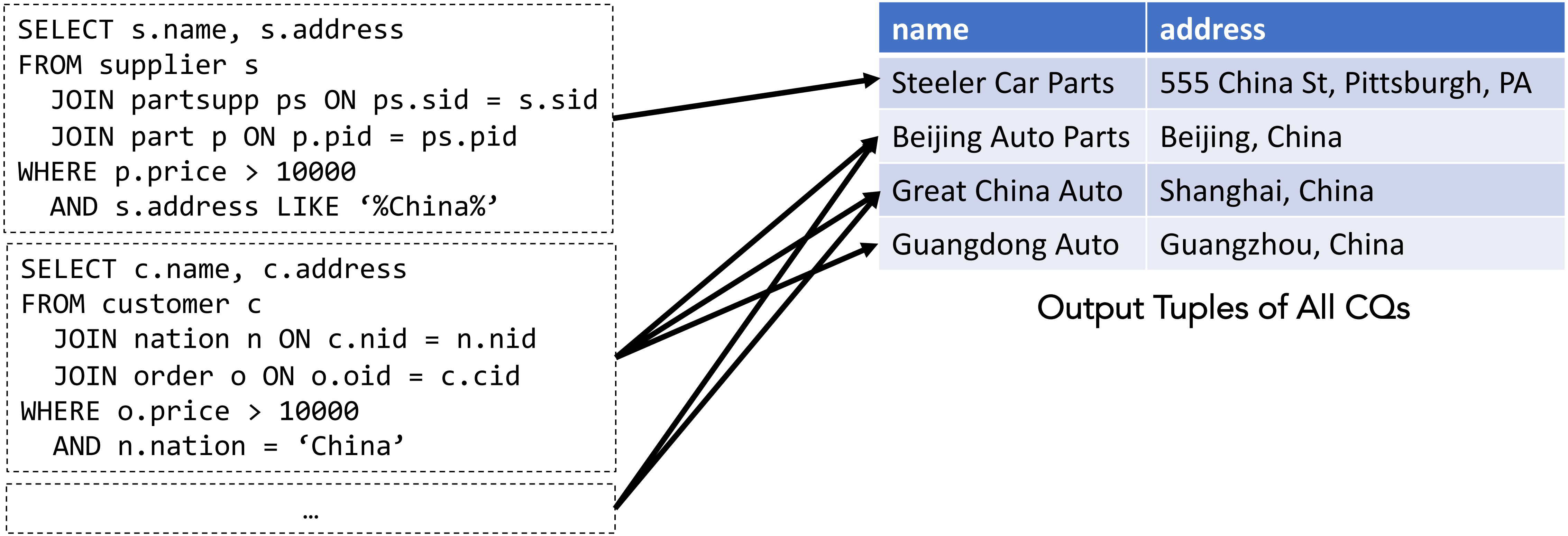
- User manually examines SQL and corresponding result sets
- Translate SQL to NL with user-provided rules [Luk 1986]
- User examines and modifies parse tree [Li 2014]
- Rephrase NL with tuples of resulting SQL [Deutch 2017]
  - Relies on initial user phrasing and may be brittle

Approach

Intuition

- Users w/o SQL knowledge can have ″tuple knowledge″ in many cases
    - Domain experts
    - Personal databases
- Utter-and-refine may be more convenient even for SQL users

Overview



```
SELECT s.name, s.address
FROM supplier s
  JOIN partsupp ps ON ps.sid = s.sid
  JOIN part p ON p.pid = ps.pid
WHERE p.price > 10000
  AND s.address LIKE ‘%China%’
```

```
SELECT c.name, c.address
FROM customer c
  JOIN nation n ON c.nid = n.nid
  JOIN order o ON o.oid = c.cid
WHERE o.price > 10000
  AND n.nation = ‘China’
```

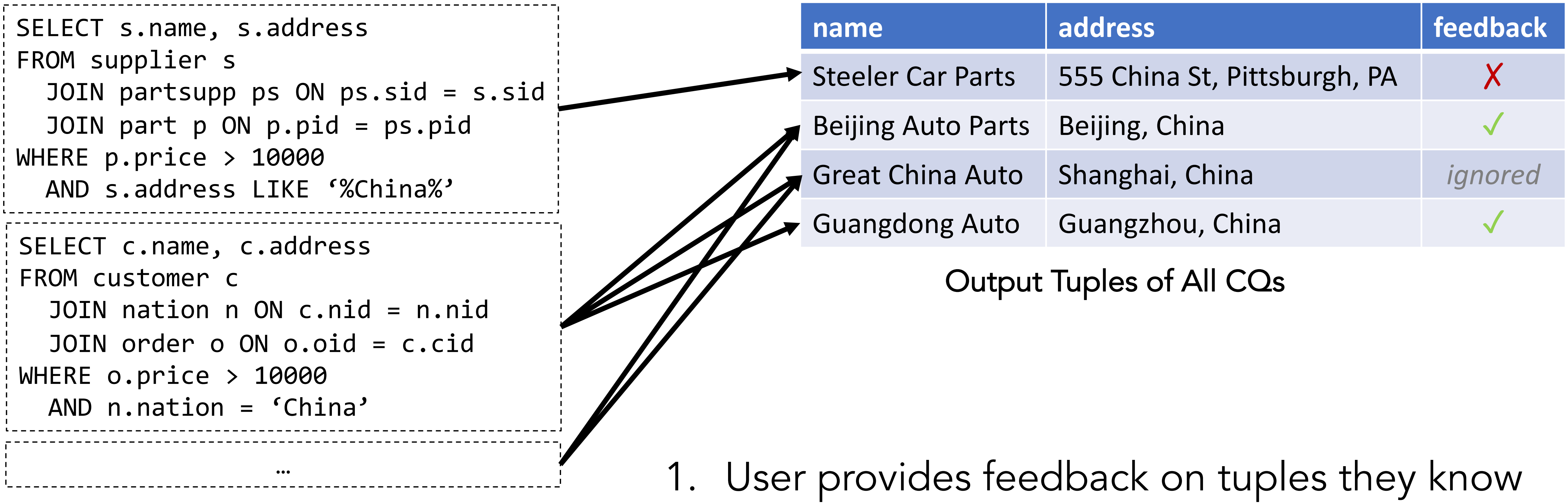
…

Candidate Queries (CQs)

name	address
Steeler Car Parts	555 China St, Pittsburgh, PA
Beijing Auto Parts	Beijing, China
Great China Auto	Shanghai, China
Guangdong Auto	Guangzhou, China

Output Tuples of All CQs

Overview



```
SELECT s.name, s.address
FROM supplier s
  JOIN partsupp ps ON ps.sid = s.sid
  JOIN part p ON p.pid = ps.pid
WHERE p.price > 10000
  AND s.address LIKE ‘%China%’
```

```
SELECT c.name, c.address
FROM customer c
  JOIN nation n ON c.nid = n.nid
  JOIN order o ON o.oid = c.cid
WHERE o.price > 10000
  AND n.nation = ‘China’
```

…

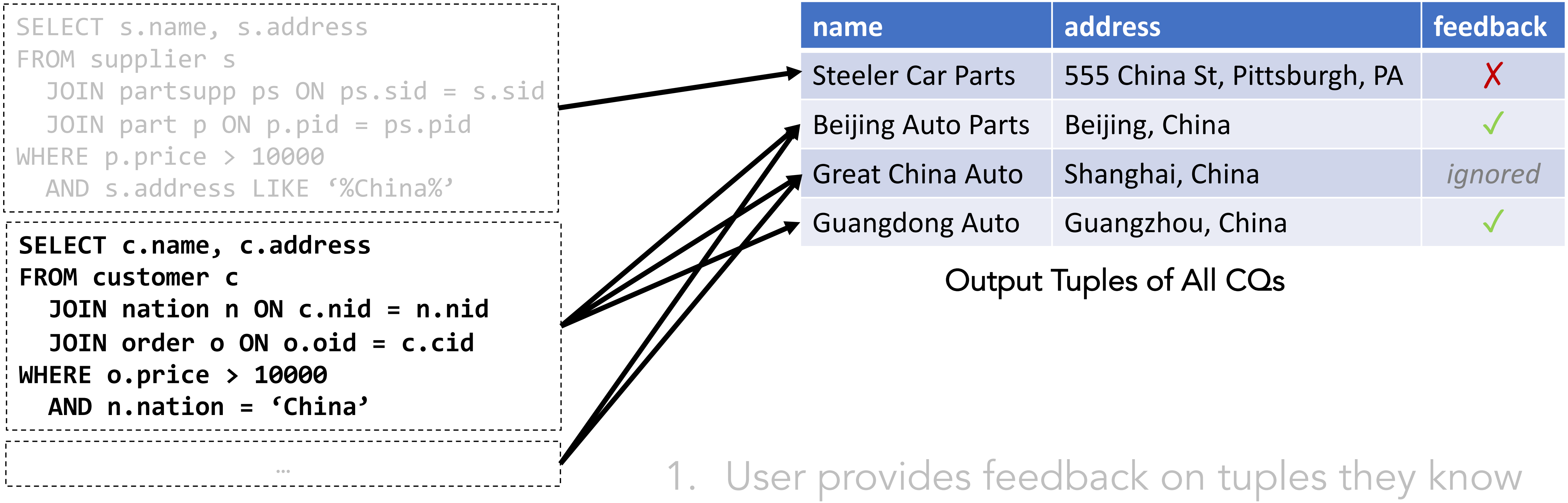
Candidate Queries (CQs)

name	address	feedback
Steeler Car Parts	555 China St, Pittsburgh, PA	✗
Beijing Auto Parts	Beijing, China	✓
Great China Auto	Shanghai, China	*ignored*
Guangdong Auto	Guangzhou, China	✓

Output Tuples of All CQs

1. User provides feedback on tuples they know
2. Desired query is returned

Overview



```
SELECT s.name, s.address
FROM supplier s
  JOIN partsupp ps ON ps.sid = s.sid
  JOIN part p ON p.pid = ps.pid
WHERE p.price > 10000
  AND s.address LIKE '%China%'
```

```
SELECT c.name, c.address
FROM customer c
  JOIN nation n ON c.nid = n.nid
  JOIN order o ON o.oid = c.cid
WHERE o.price > 10000
  AND n.nation = 'China'
```

```
…
```

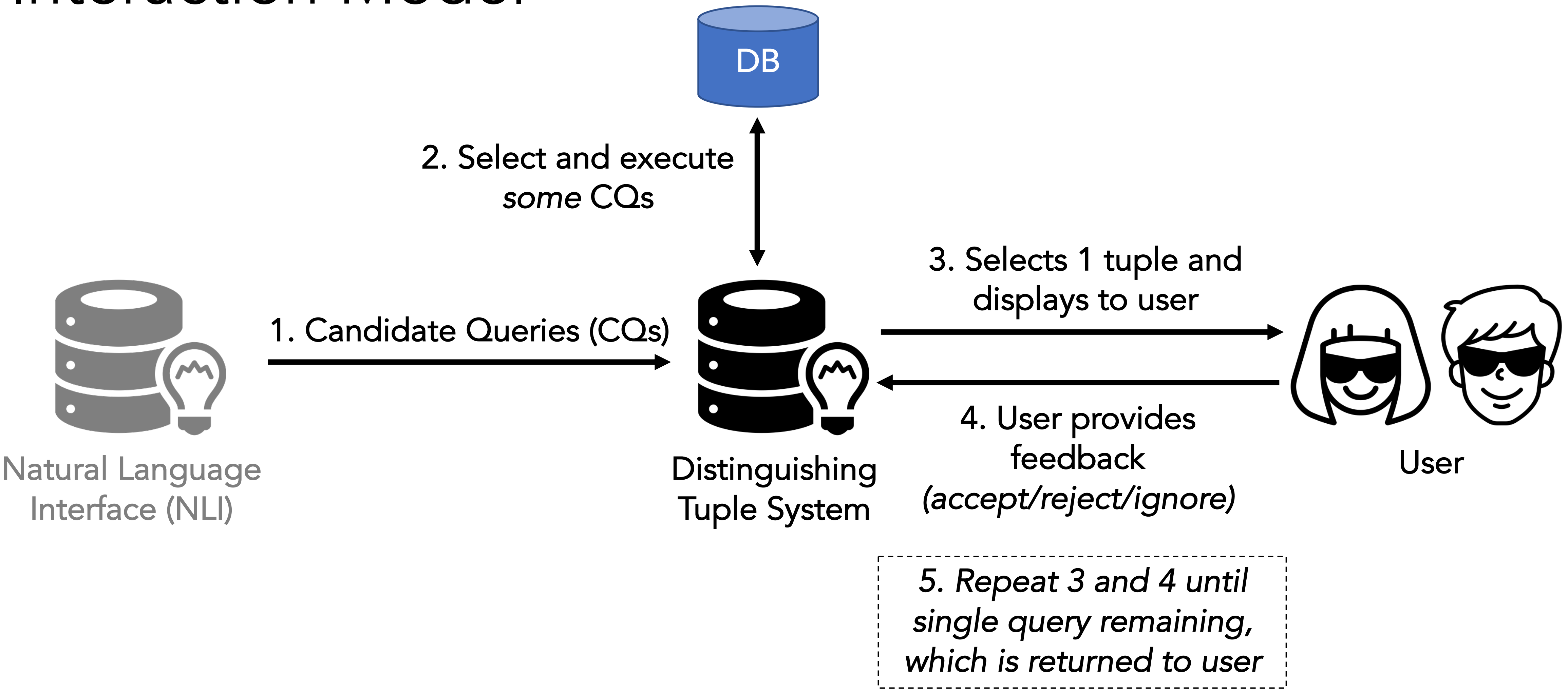
Candidate Queries (CQs)

name	address	feedback
Steeler Car Parts	555 China St, Pittsburgh, PA	✗
Beijing Auto Parts	Beijing, China	✓
Great China Auto	Shanghai, China	*ignored*
Guangdong Auto	Guangzhou, China	✓

Output Tuples of All CQs

1. User provides feedback on tuples they know
2. Desired query is returned

Interaction Model

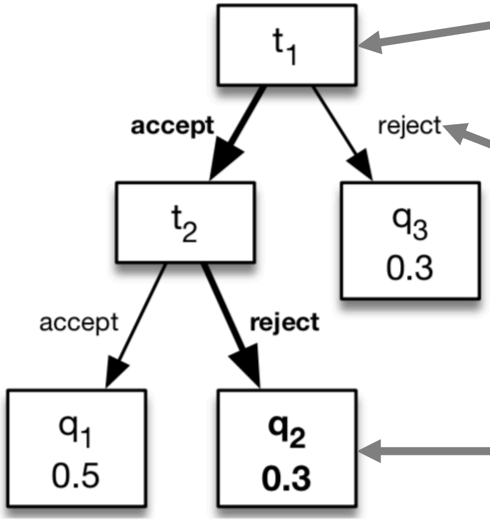

DB
2. Select and execute *some* CQs
1. Candidate Queries (CQs)
3. Selects 1 tuple and displays to user
4. User provides feedback *(accept/reject/ignore)*
Natural Language Interface (NLI)
Distinguishing Tuple System
User
5. Repeat 3 and 4 until single query remaining, which is returned to user

Goals

1. Minimize **user effort** (i.e. the **number of tuples** displayed to the user)
2. Minimize **system execution time**

Problem: Minimize User Effort

Formally: Find the minimum cost split tree.



internal nodes: tuples from CQ result sets

user feedback

ignore not shown

leaf nodes: candidate queries

each CQ has a weight 0 <= w <= 1

Split Tree

A "flowchart" of possible user responses

Many possible cost functions, we choose **total weighted cost**

$$c(\mathcal{T}) = \sum_{i=1}^{n} l_i w(q_i)$$

for each leaf node

length from root to leaf

i.e. number of tuples

weight

Solution Sketch

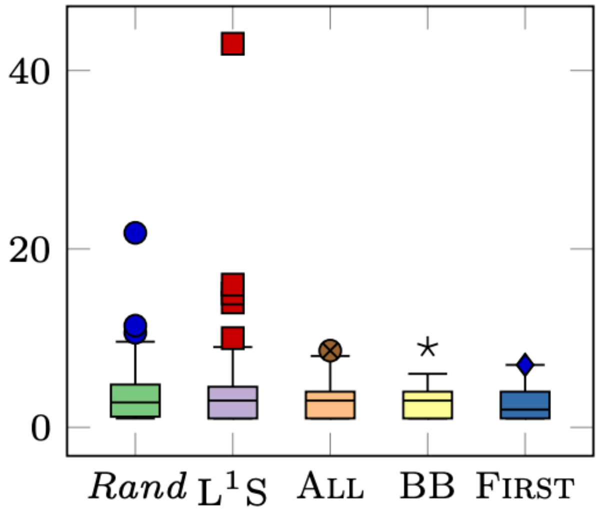
- Problem is NP-hard
- Greedy algorithm constructs split tree top-down, one tuple at a time
- Improve using static analysis to avoid executing certain CQs

Experiments

Experiment Setup

- Simulated user provides correct feedback every time (accept or reject)
- CQs are assigned equal weight
- Tuples are presented to user one at a time until target query remains
- IMDB NLQ-SQL dataset from [Yagmazadeh 2017]
  - Execute NLQ on generic NLI to get CQs
  - Original labeled SQL is target query
- Compared approaches
  - Our 3 algorithms (All, BB, First)
  - Randomly selecting a tuple
  - L^1S from [Bonifati 2016]

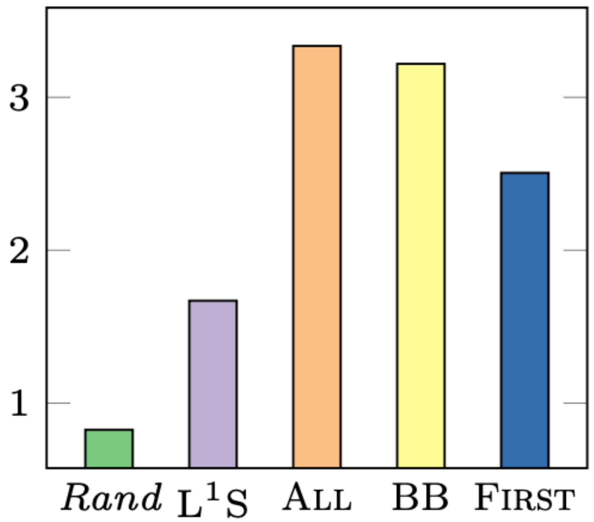
Number of tuples displayed to user



(a) Iterations for each task

- Our algorithms (All, BB, First) mitigate effect of worst-case outliers

System runtime per iteration



(b) System runtime (s)/iter

- Algorithms require overhead
- Total runtime for task is system runtime + user response time
- More study needs to be done

Conclusion

Takeaways

- Natural language interfaces (NLIs) can be useful for database querying
- Precise clarification mechanisms are needed for NLIs
- Distinguishing tuples are one potential solution

Questions and comments

cjbaik@umich.edu

Icons

- AomAm from the Noun Project
- knowledge database by sahua d from the Noun Project