Bridging the Semantic Gap with SQL Query Logs in Natural Language Interfaces to Databases

Christopher Baik[1], H. V. Jagadish[1], Yunyao Li[2]

[1]University of Michigan – Ann Arbor

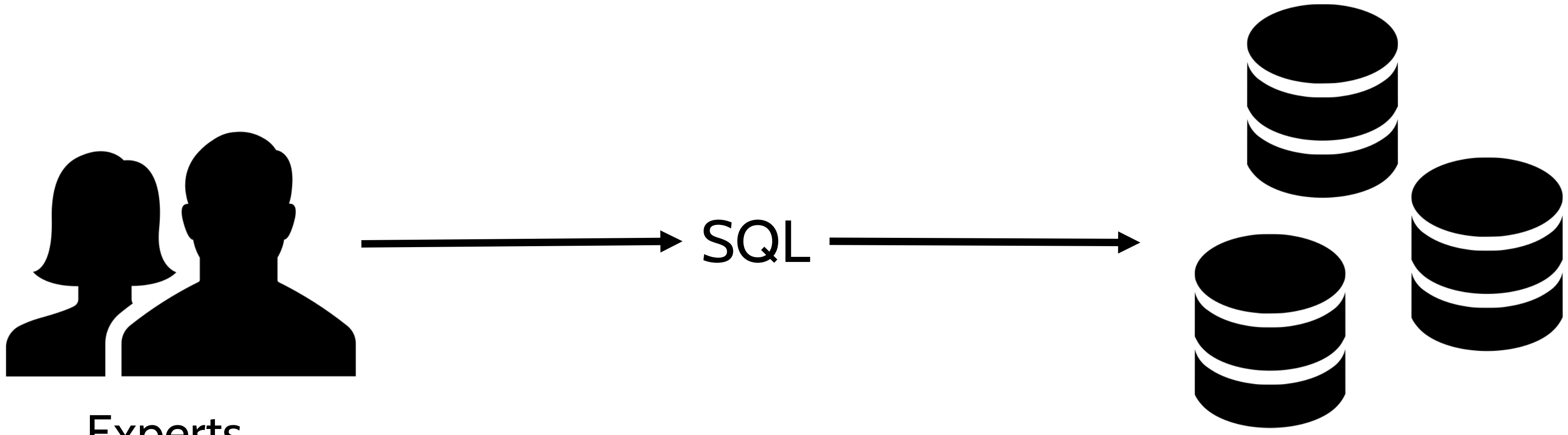
[2]IBM Research – Almaden

Agenda

- Motivation
- Solution Approach
- Solution Details
- Experiments
- Conclusion

Motivation

Motivation > NLIDBs



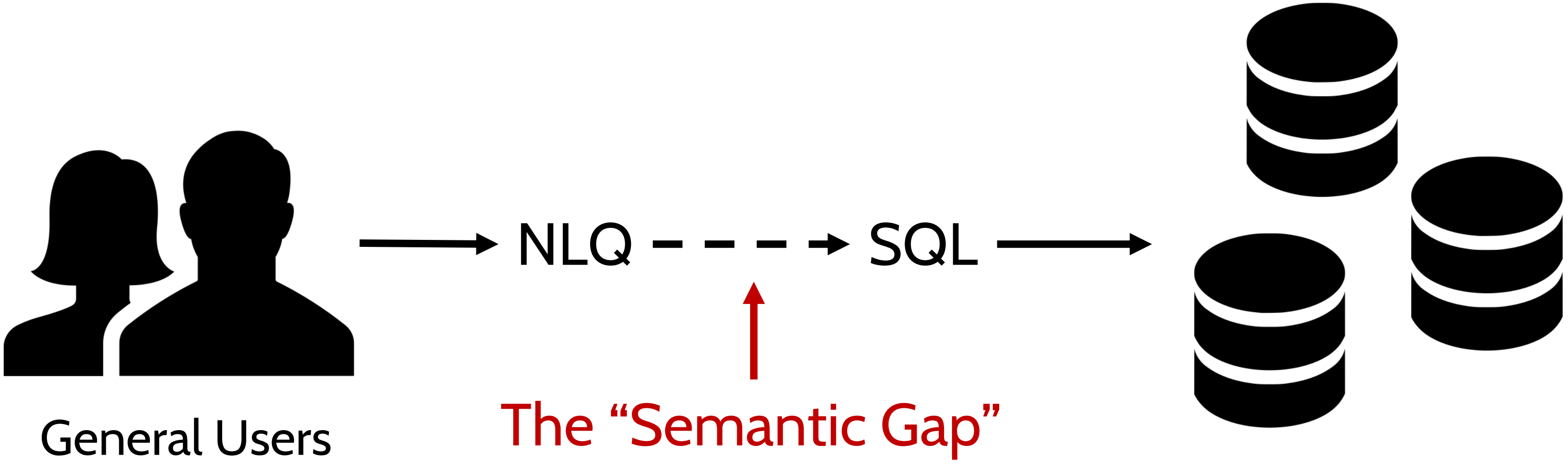
SQL

Experts

- Understand relational model
- Know about specific schema and contents

Relational Databases

Motivation > NLIDBs



General Users

The “Semantic Gap”

Relational Databases

- Relational model is challenging
- No knowledge of schema and contents

Motivation > The Semantic Gap

NLQ
Find papers after 2005 in the Databases domain.

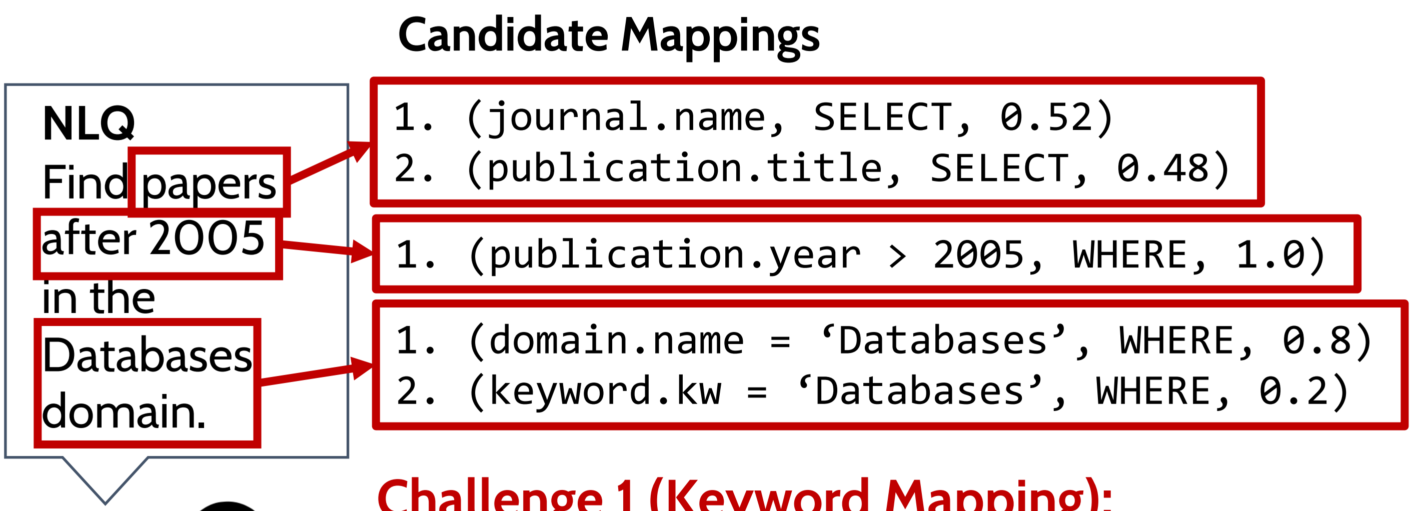
Subproblems

1. Keyword mapping
2. Join path inference

Intended SQL

```
SELECT p.title
FROM publication AS p
JOIN publication_keyword pk
  ON p.pid = pk.pid
JOIN keyword k ON pk.kid = k.kid
JOIN domain_keyword dk
  ON k.kid = dk.kid
JOIN domain d ON d.did = dk.did
WHERE d.name = ‘Databases’
  AND p.year > 2005
```

Motivation > The Semantic Gap > 1. Keyword Mapping



Candidate Mappings

NLQ
Find papers after 2005 in the Databases domain.

```
1. (journal.name, SELECT, 0.52)
2. (publication.title, SELECT, 0.48)
```

```
1. (publication.year > 2005, WHERE, 1.0)
```

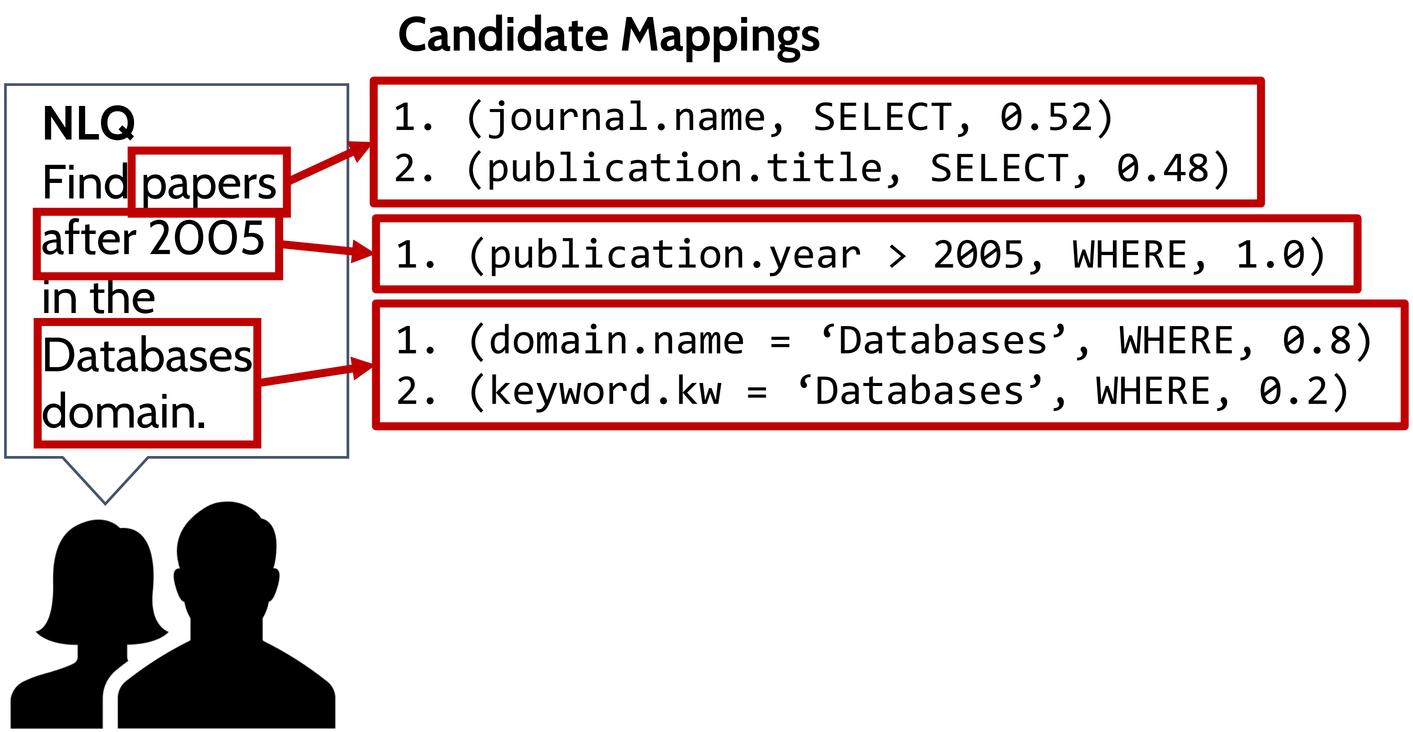
```
1. (domain.name = ‘Databases’, WHERE, 0.8)
2. (keyword.kw = ‘Databases’, WHERE, 0.2)
```

Challenge 1 (Keyword Mapping):
How do we decide which mapping to select?

Existing Approaches

- *Standard*: Highest word embedding similarity score
- Ask the user to pick [Popescu 2003, Li 2014]
- Pre-define mappings with ontology [Saha 2016]

Motivation > The Semantic Gap > 2. Join Path Inference



Candidate Mappings

NLQ

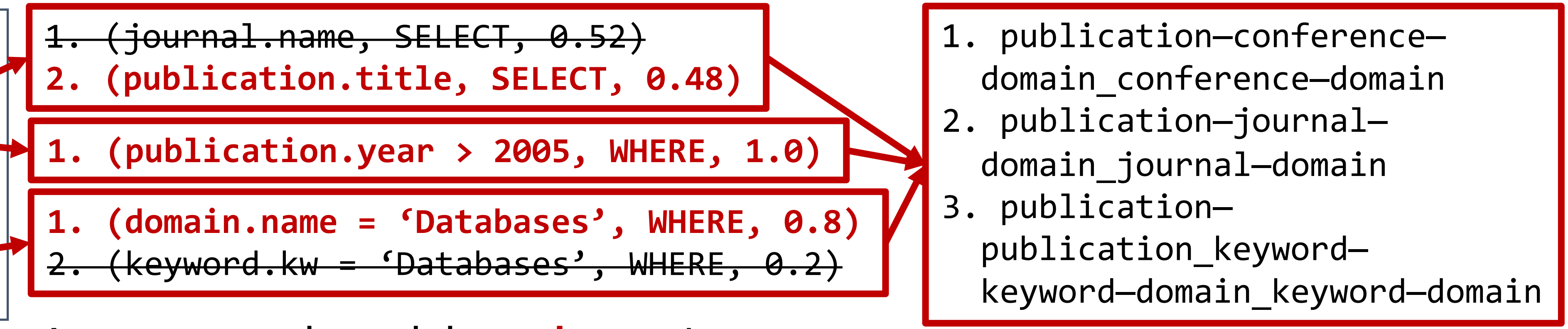
Find papers after 2005 in the Databases domain.

```
1. (journal.name, SELECT, 0.52)
2. (publication.title, SELECT, 0.48)
```

```
1. (publication.year > 2005, WHERE, 1.0)
```

```
1. (domain.name = ‘Databases’, WHERE, 0.8)
2. (keyword.kw = ‘Databases’, WHERE, 0.2)
```

Motivation > The Semantic Gap > 2. Join Path Inference



Candidate Mappings

~~1. (journal.name, SELECT, 0.52)~~
2. (publication.title, SELECT, 0.48)

1. (publication.year > 2005, WHERE, 1.0)

1. (domain.name = ‘Databases’, WHERE, 0.8)
~~2. (keyword.kw = ‘Databases’, WHERE, 0.2)~~

Candidate Join Paths

1. publication–conference–domain_conference–domain
2. publication–journal–domain_journal–domain
3. publication–publication_keyword–keyword–domain_keyword–domain

Assume we selected the **red** mappings

Challenge 2 (Join Path Inference):
How do we decide which join path to select?

Existing Approaches

- *Standard*: Select shortest tree (i.e. Steiner tree)
- Ask the user to pick [Popescu 2003, Li 2014]
- Pre-define mappings with ontology [Saha 2016]

Solution Approach

Solution Approach > Typical Approach

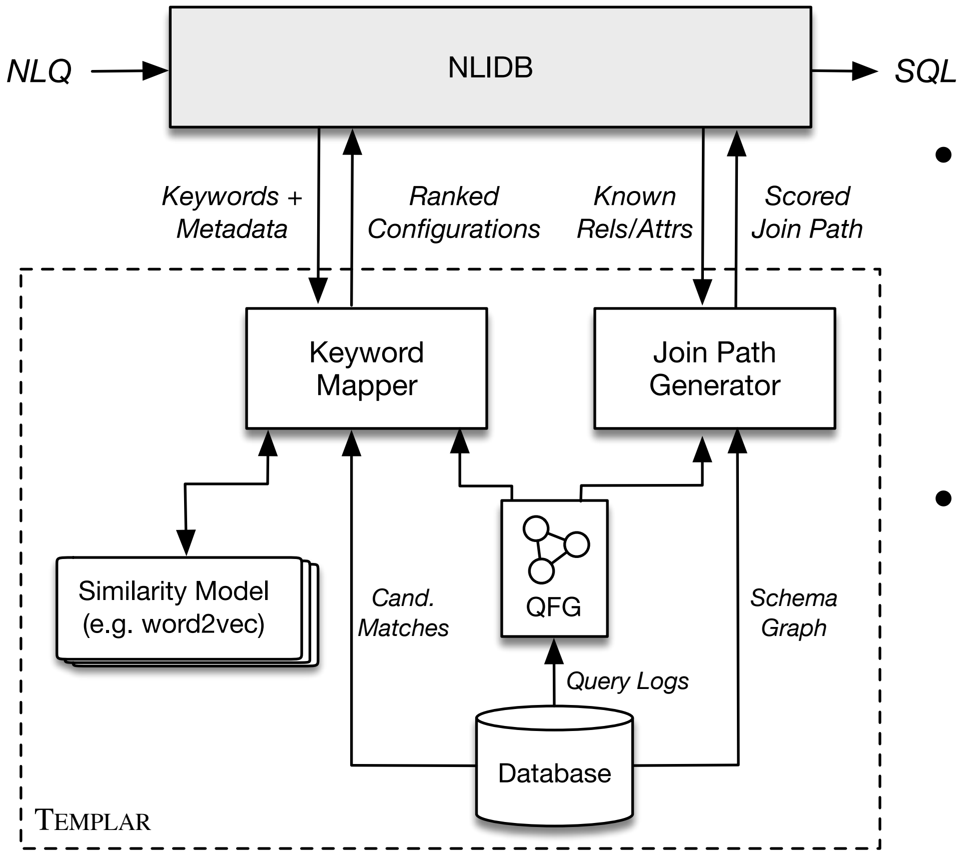
- **Goal:** Learn from existing queries/training data
- **Typical Approach**: Collect NLQ-SQL pairs
  - Lots of work [Zhong 2017, Xu 2017, Yu 2018, etc…]
- **Challenges:**
  1. Labeled pairs of NLQ-SQL are costly to obtain, requiring time and expertise
  2. High volume of data required to train system
  3. Data, esp. for join paths, must be domain-/schema-specific

Solution Approach > Our Approach

- **Our Approach:** Instead of NLQ-SQL pairs, SQL query logs
  - Readily available for production databases
  - Contain information on more common/likely user queries
- **Challenge:** Learning NLQ to SQL only using output (i.e. SQL)

Solution Details

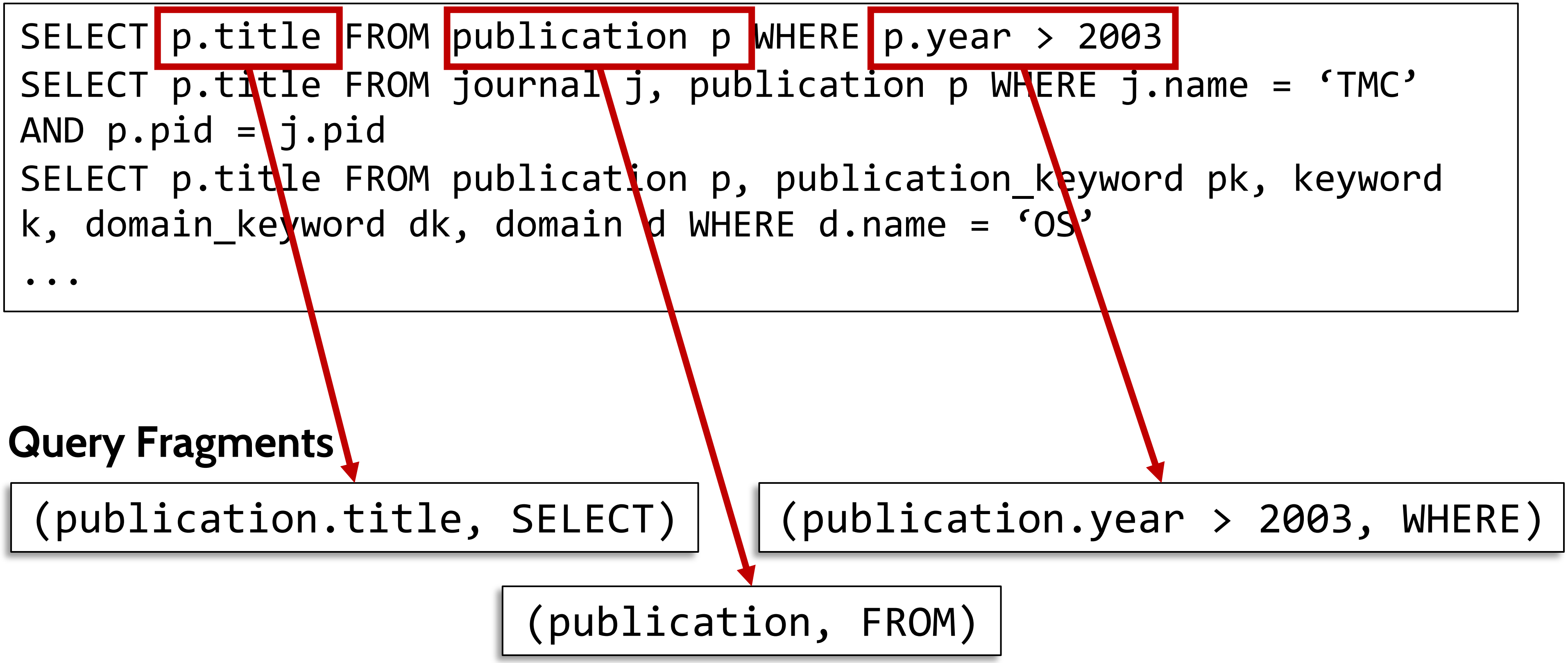
Solution Details > System Overview



- **Augments** existing NLIDBs, which are still responsible for:
  - Parsing natural language
  - Extracting keywords
  - Decoding SQL clause structure
- Note the **order of execution**
  1. Keyword mapping
  2. Join path inference

Solution Details > Query Fragments

Query Log



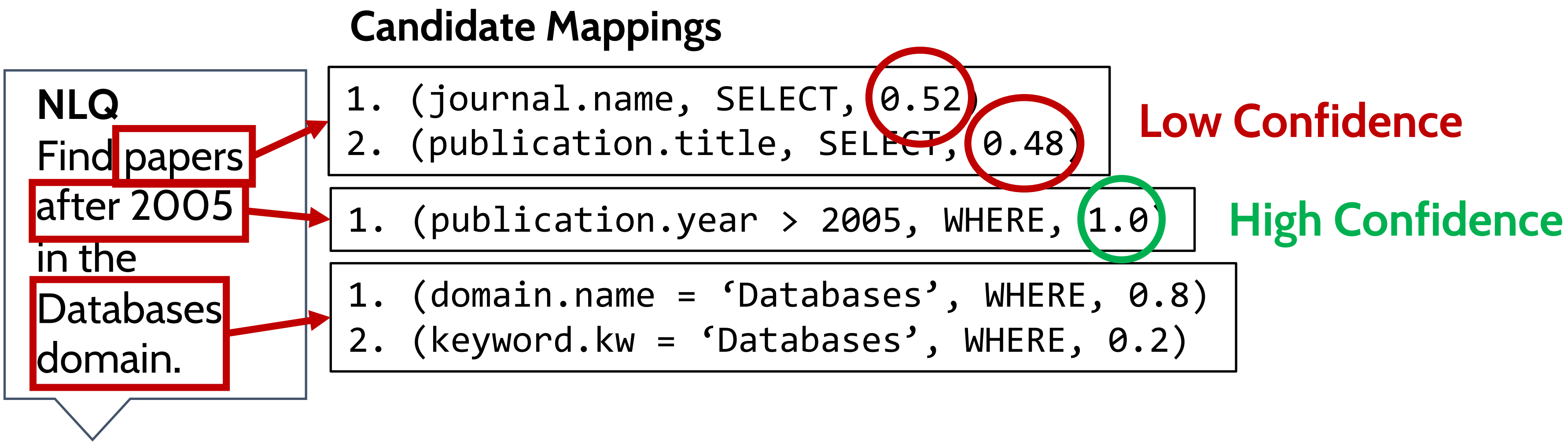
```
SELECT p.title FROM publication p WHERE p.year > 2003
SELECT p.title FROM journal j, publication p WHERE j.name = ‘TMC’
AND p.pid = j.pid
SELECT p.title FROM publication p, publication_keyword pk, keyword
k, domain_keyword dk, domain d WHERE d.name = ‘OS’
...
```

Query Fragments

```
(publication.title, SELECT)
(publication.year > 2003, WHERE)
(publication, FROM)
```

Solution Details > Modeling the SQL Query Log

- **Goal:** Model SQL query log to assist NLQ to SQL translation
- **Intuition:**
  - Keyword mappings vary in confidence
  - Anchor low-conf mappings given *co-occurrence* with high-conf ones



Candidate Mappings

NLQ
Find papers after 2005 in the Databases domain.

```
1. (journal.name, SELECT, 0.52)
2. (publication.title, SELECT, 0.48)
```

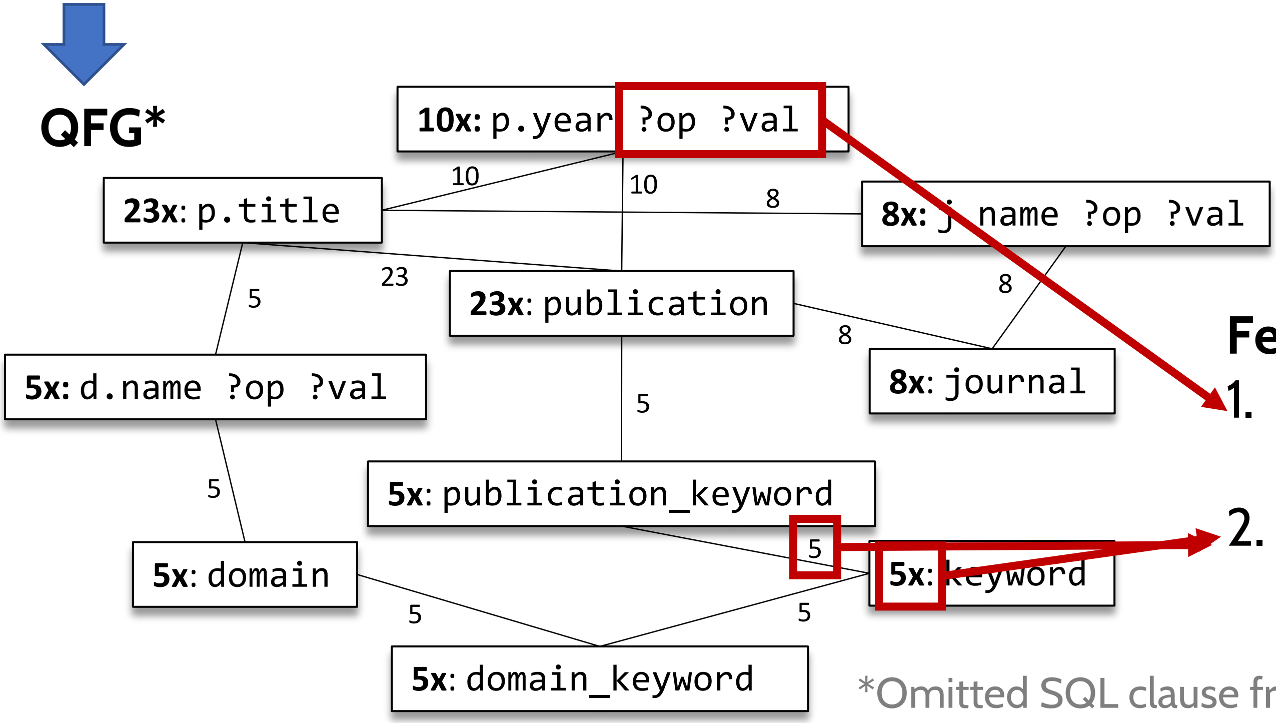
```
1. (publication.year > 2005, WHERE, 1.0)
```

```
1. (domain.name = ‘Databases’, WHERE, 0.8)
2. (keyword.kw = ‘Databases’, WHERE, 0.2)
```

Solution Details > Query Fragment Graph (QFG)

Query Log

```
SELECT p.title FROM publication p WHERE p.year > 2003
SELECT p.title FROM journal j, publication p WHERE j.name = ‘TMC’
AND p.pid = j.pid
SELECT p.title FROM publication p, publication_keyword pk, keyword
k, domain_keyword dk, domain d WHERE d.name = ‘OS’
...
```

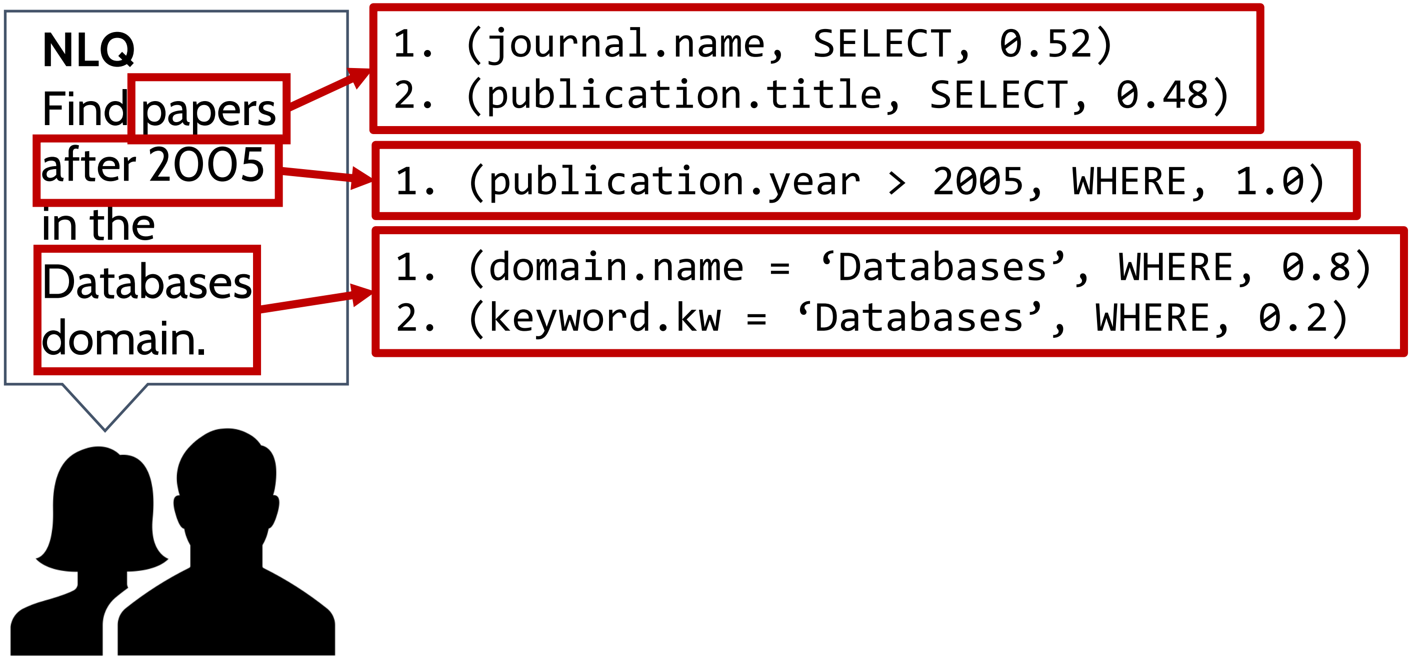


Features

1. Abstracted versions of query fragments
2. Store occurrence and co-occurrence

*Omitted SQL clause from fragments for space

Solution Details > Applying the Model > 1. Keyword Mapping



Candidate Mappings

NLQ
Find papers after 2005 in the Databases domain.

```
1. (journal.name, SELECT, 0.52)
2. (publication.title, SELECT, 0.48)
```

```
1. (publication.year > 2005, WHERE, 1.0)
```

```
1. (domain.name = ‘Databases’, WHERE, 0.8)
2. (keyword.kw = ‘Databases’, WHERE, 0.2)
```

Solution Details > Applying the Model > 1. Keyword Mapping

Candidate Mappings

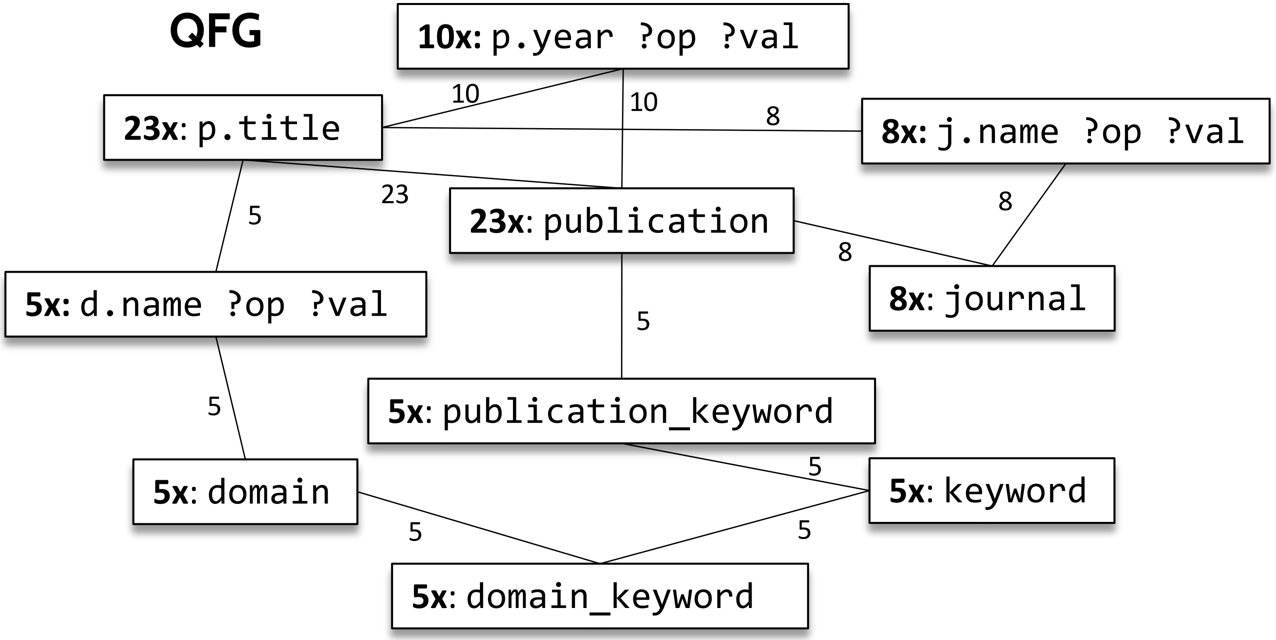
```
1. (journal.name, SELECT, 0.52)
2. (publication.title, SELECT, 0.48)
```

```
1. (publication.year > 2005, WHERE, 1.0)
```

```
1. (domain.name = ‘Databases’, WHERE, 0.8)
2. (keyword.kw = ‘Databases’, WHERE, 0.2)
```

Score each **combination** of mappings using weighted sum of similarity and QFG co-occurrence

QFG



Solution Details > Applying the Model > 1. Keyword Mapping

Candidate Mappings

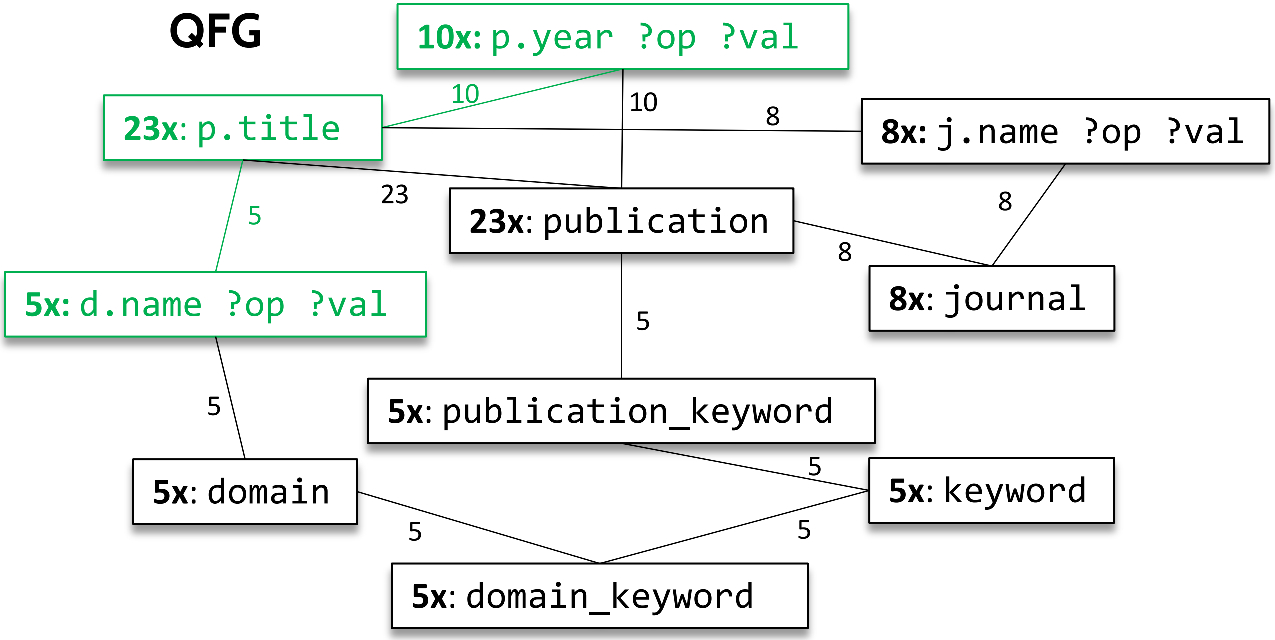
~~1. (journal.name, SELECT, 0.52)~~
2. (publication.title, SELECT, 0.48)

1. (publication.year > 2005, WHERE, 1.0)

1. (domain.name = ‘Databases’, WHERE, 0.8)
~~2. (keyword.kw = ‘Databases’, WHERE, 0.2)~~

Score each **combination** of mappings using weighted sum of similarity and QFG co-occurrence

QFG



Solution Details > Applying the Model > 2. Join Path Inference

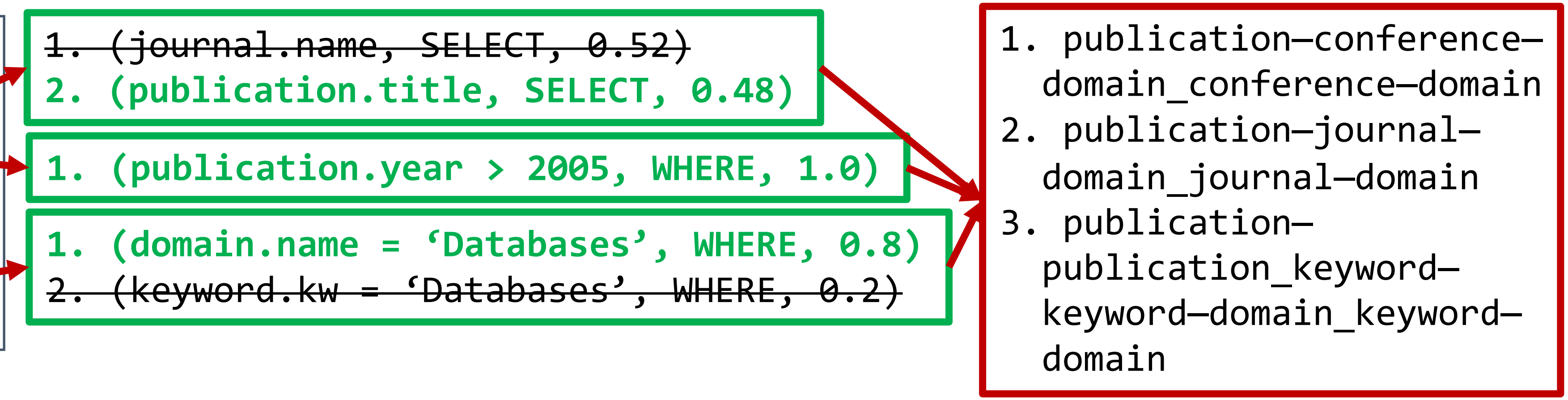
Candidate Mappings

~~1. (journal.name, SELECT, 0.52)~~
2. (publication.title, SELECT, 0.48)

1. (publication.year > 2005, WHERE, 1.0)

1. (domain.name = ‘Databases’, WHERE, 0.8)
~~2. (keyword.kw = ‘Databases’, WHERE, 0.2)~~

Solution Details > Applying the Model > 2. Join Path Inference



Candidate Mappings

~~1. (journal.name, SELECT, 0.52)~~
2. (publication.title, SELECT, 0.48)

1. (publication.year > 2005, WHERE, 1.0)

1. (domain.name = ‘Databases’, WHERE, 0.8)
~~2. (keyword.kw = ‘Databases’, WHERE, 0.2)~~

Candidate Join Paths

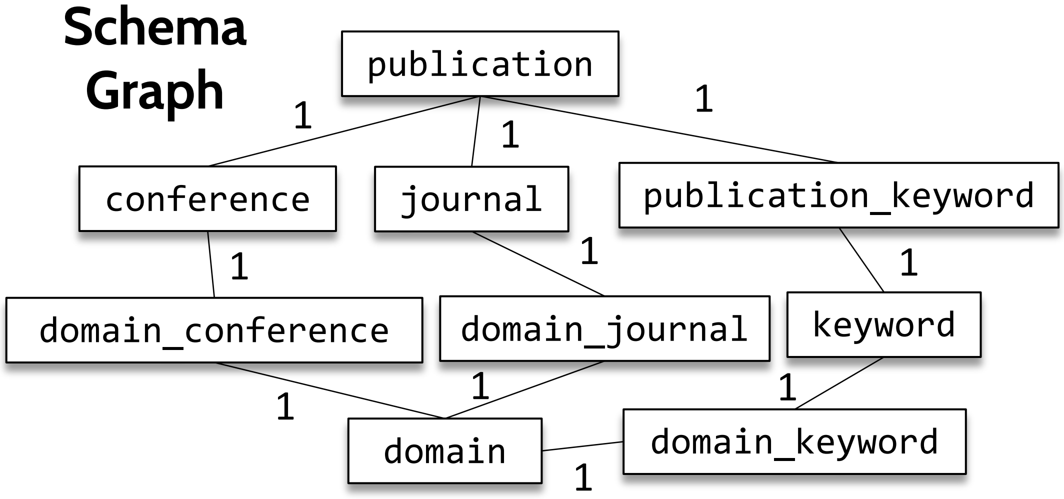
1. publication–conference–domain_conference–domain
2. publication–journal–domain_journal–domain
3. publication–publication_keyword–keyword–domain_keyword–domain

Solution Details > Applying the Model > 2. Join Path Inference

Candidate Join Paths

1. publication–conference–domain_conference–domain
2. publication–journal–domain_journal–domain
3. publication–publication_keyword–keyword–domain_keyword–domain

Schema Graph

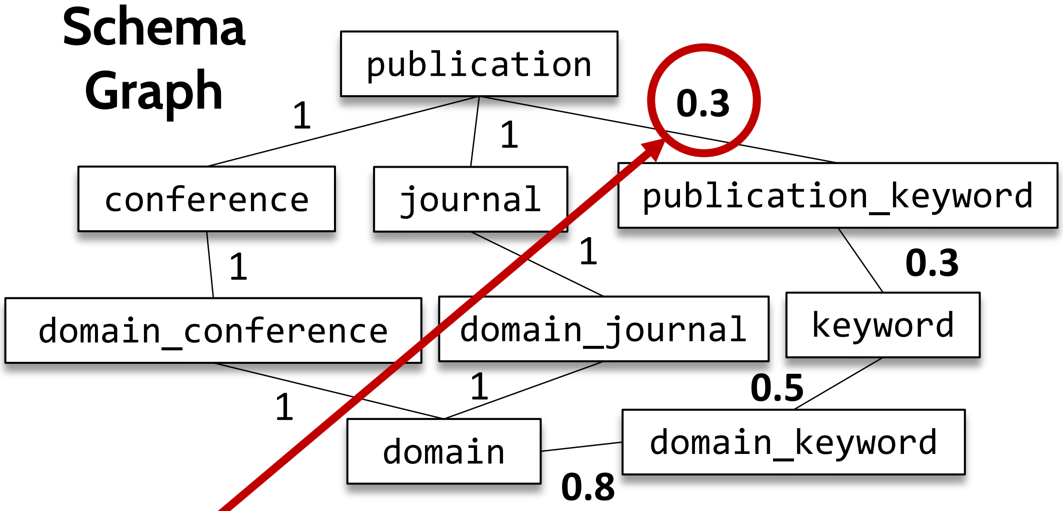


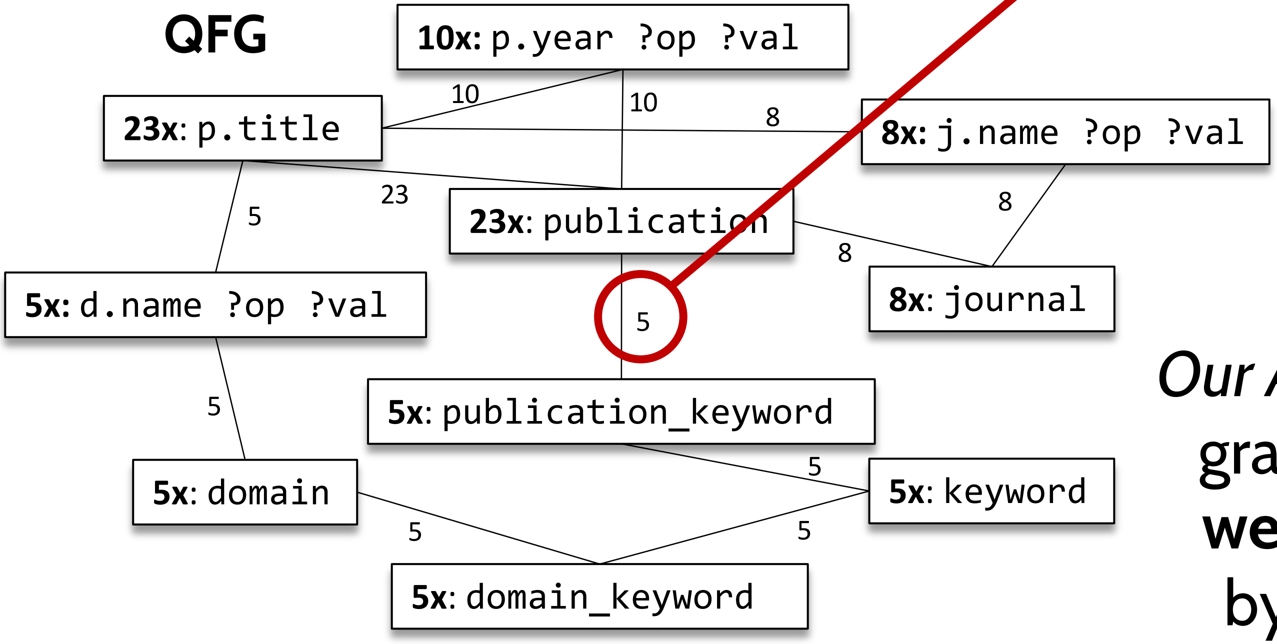
Standard: Choose shortest path on schema graph

Solution Details > Applying the Model > 2. Join Path Inference

Candidate Join Paths

1. publication–conference–domain_conference–domain
2. publication–journal–domain_journal–domain
3. publication–publication_keyword–keyword–domain_keyword–domain





Standard: Choose shortest path on schema graph

Our Approach: Schema graph edge weights **weighted inversely** by co-occurrence

Solution Details > Applying the Model > 2. Join Path Inference

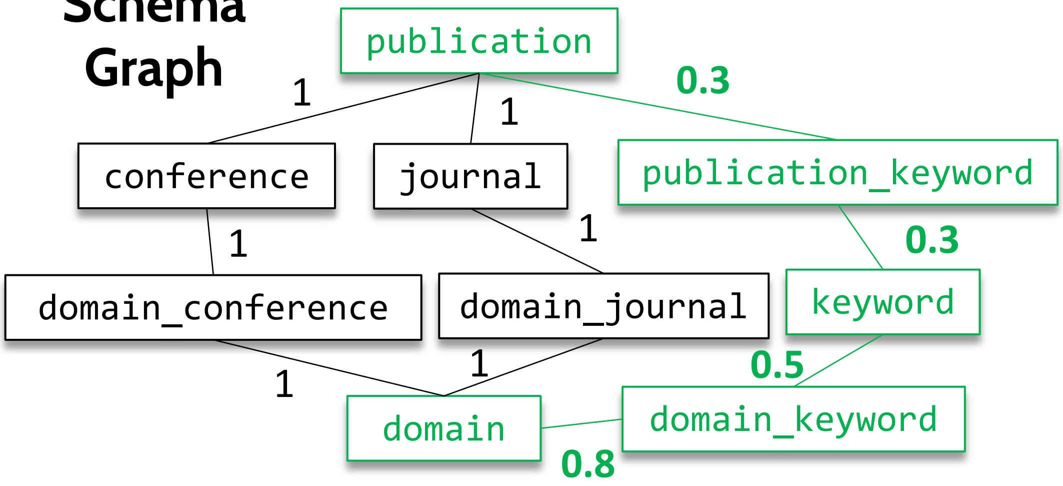
Candidate Join Paths

~~1. publication–conference–domain_conference–domain~~

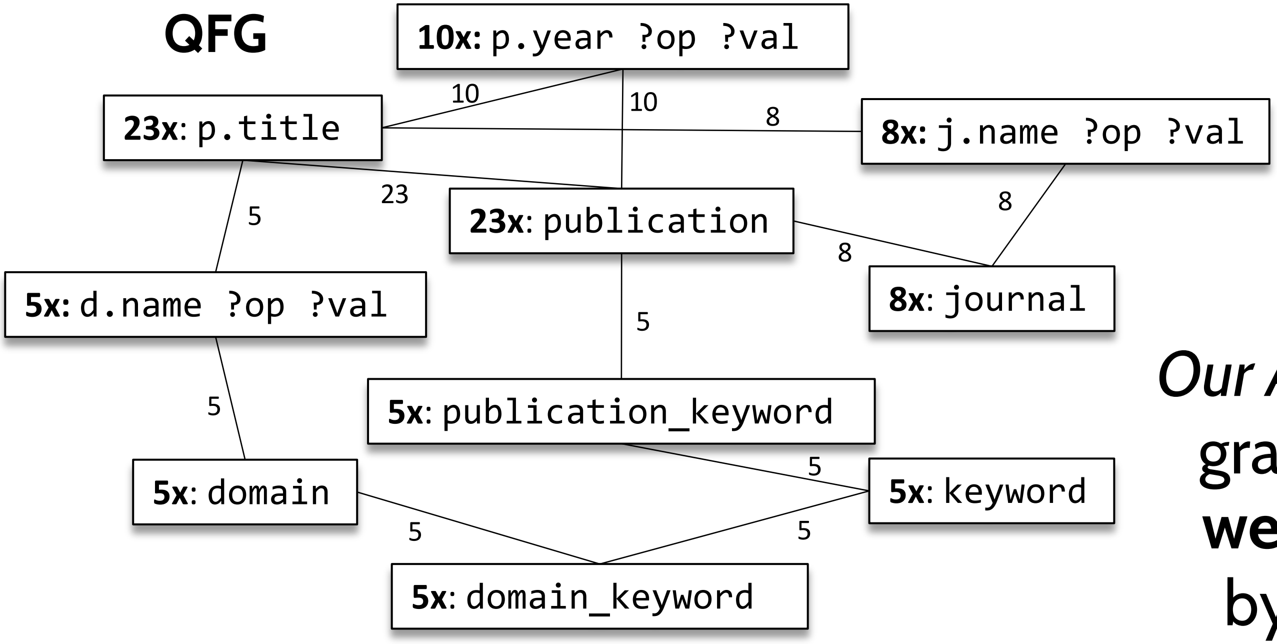
~~2. publication–journal–domain_journal–domain~~

3. publication–publication_keyword–keyword–domain_keyword–domain

Schema Graph



QFG



Standard: Choose shortest path on schema graph

Our Approach: Schema graph edge weights **weighted inversely** by co-occurrence

Experiments

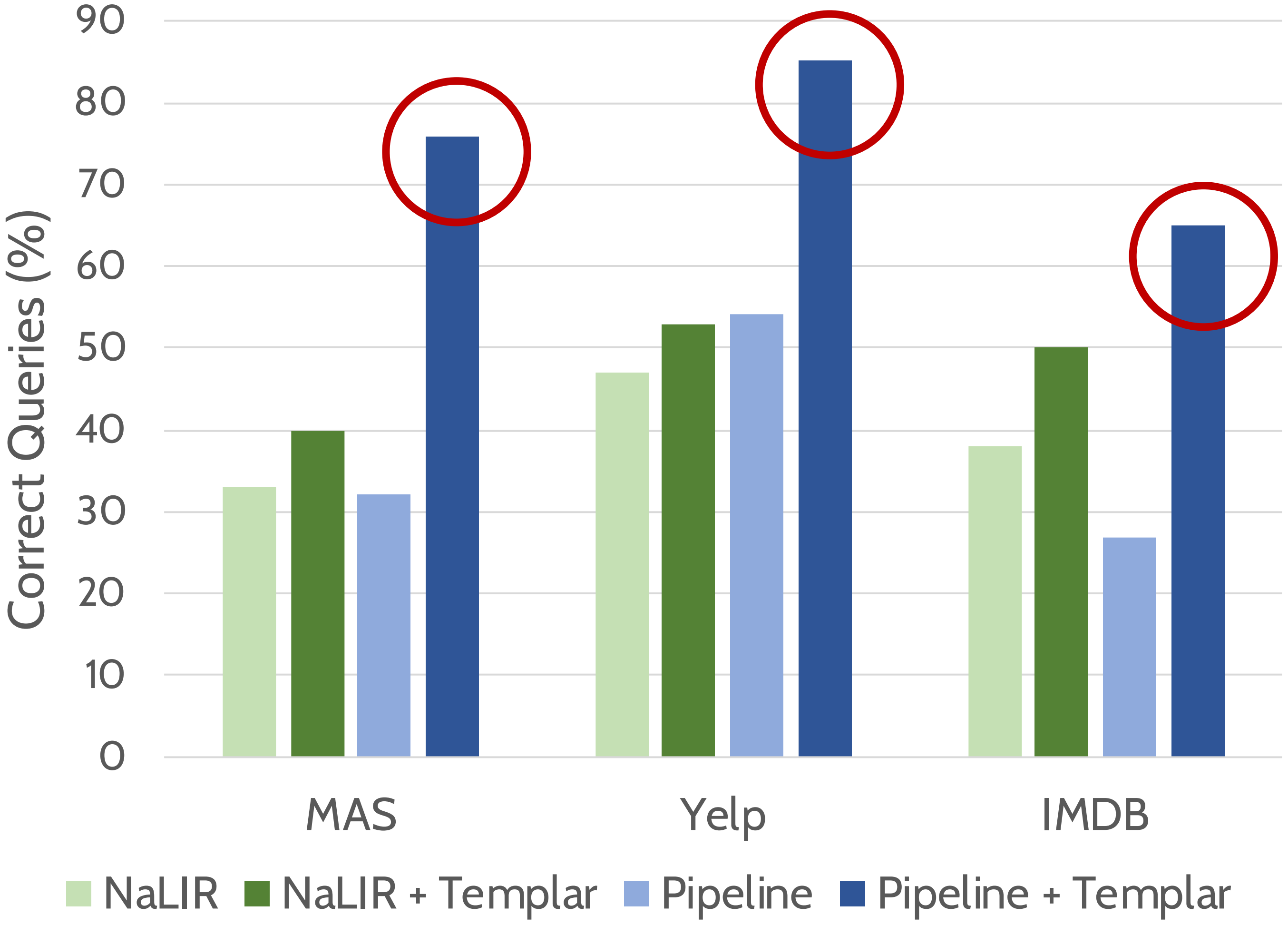
Experiments > Setup

- Benchmarks

Dataset	Queries	Tables	Cols	FK-PK Paths
MAS [Li 2014]	194 NLQ-SQL	17	53	19
Yelp [Yaghmazadeh 2017]	127 NLQ-SQL	7	38	7
IMDB [Yaghmazadeh 2017]	128 NLQ-SQL	16	65	20

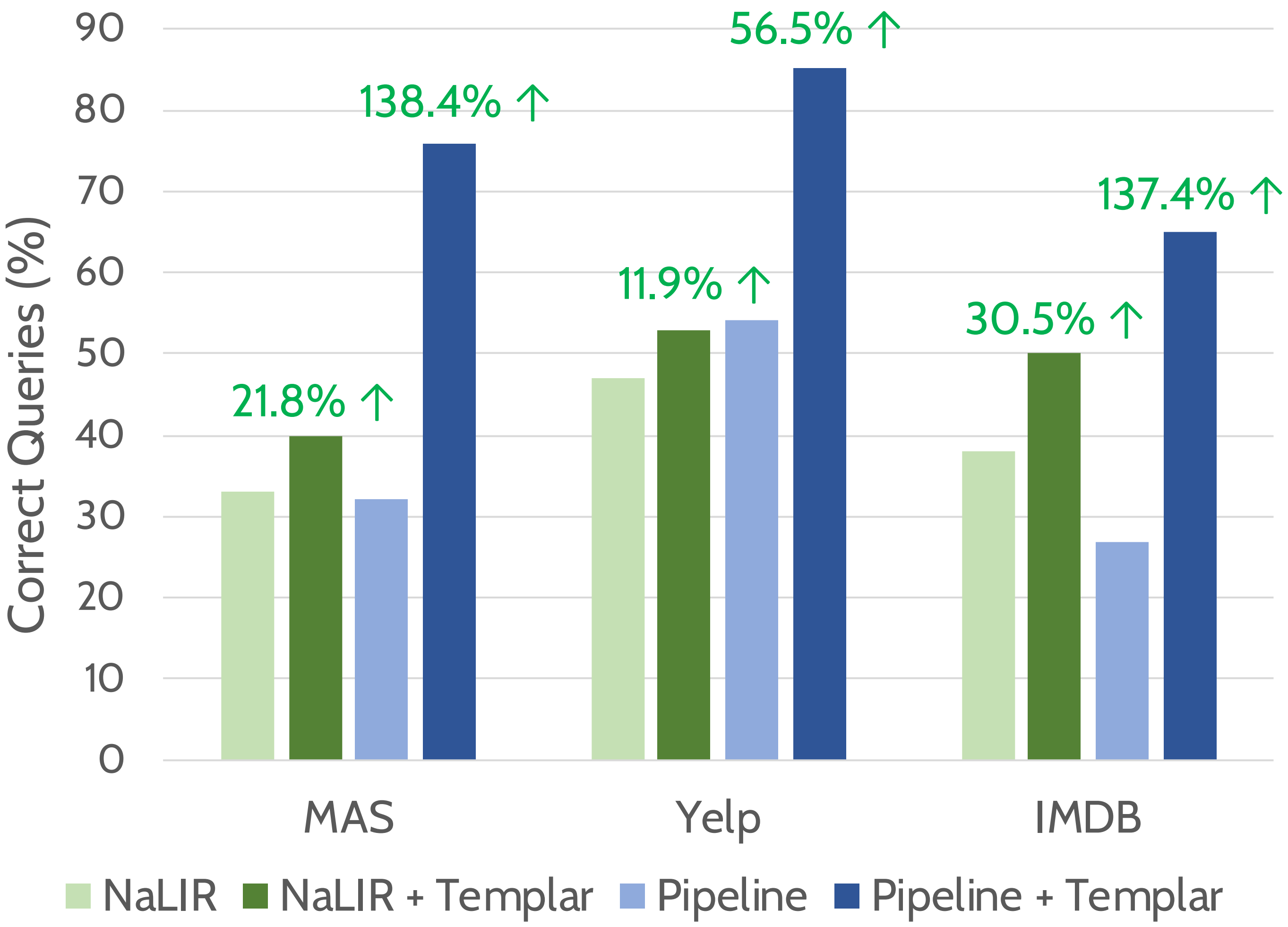
- Tested Systems
  - NaLIR [Li 2014]
  - Pipeline (emulation of SQLizer [Yaghmazadeh 2017])
- Performed 4-fold cross validation on NLQ-SQL pairs
  - Used only SQL of 3 training folds as query log
  - Tested NLQ-SQL of 1 test fold
  - **Caveat:** Assumes NLQ-SQL workload similar to SQL query log

Experiments > End-to-end



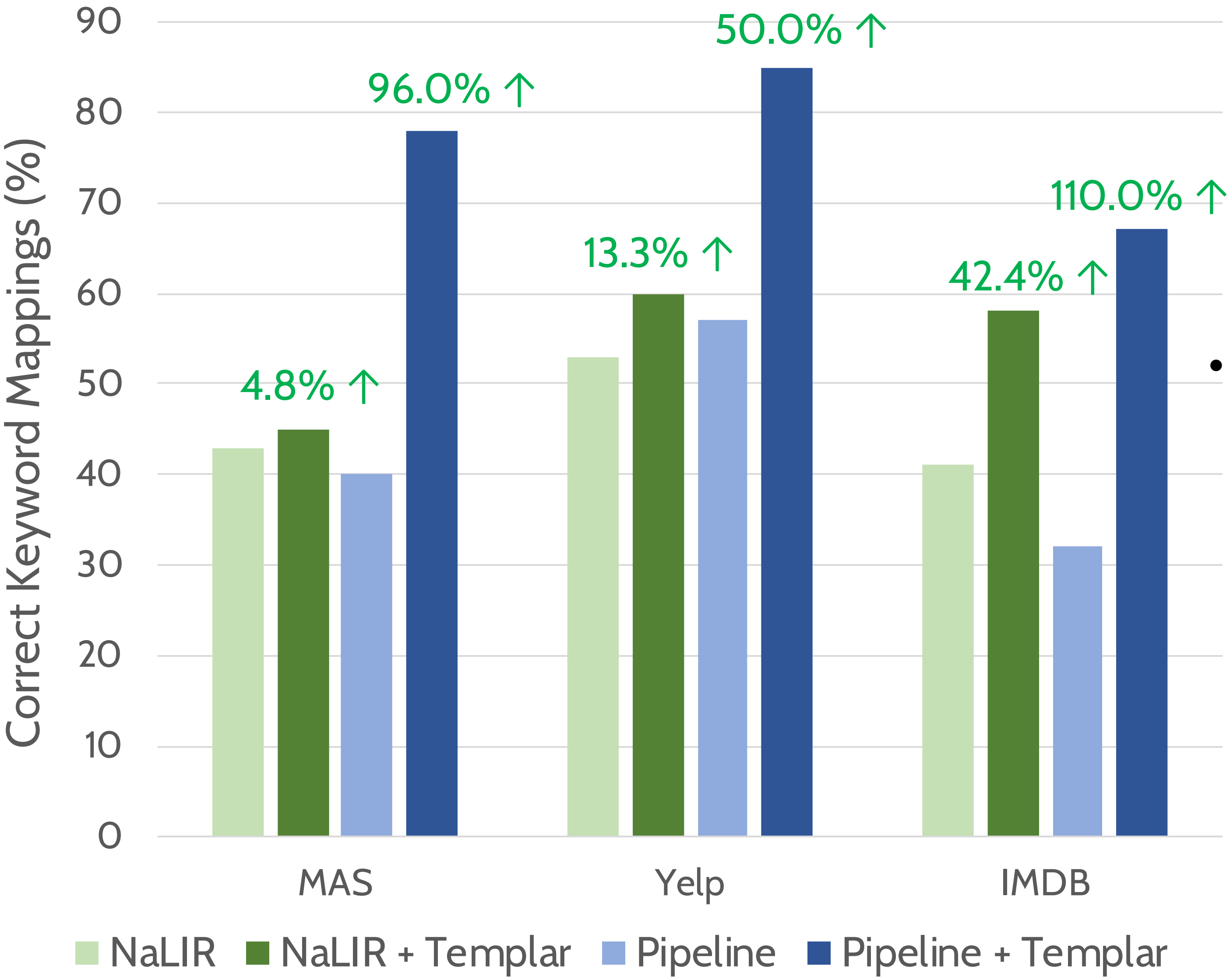
- **Pipeline + Templar** is by far the best-performing

Experiments > End-to-end



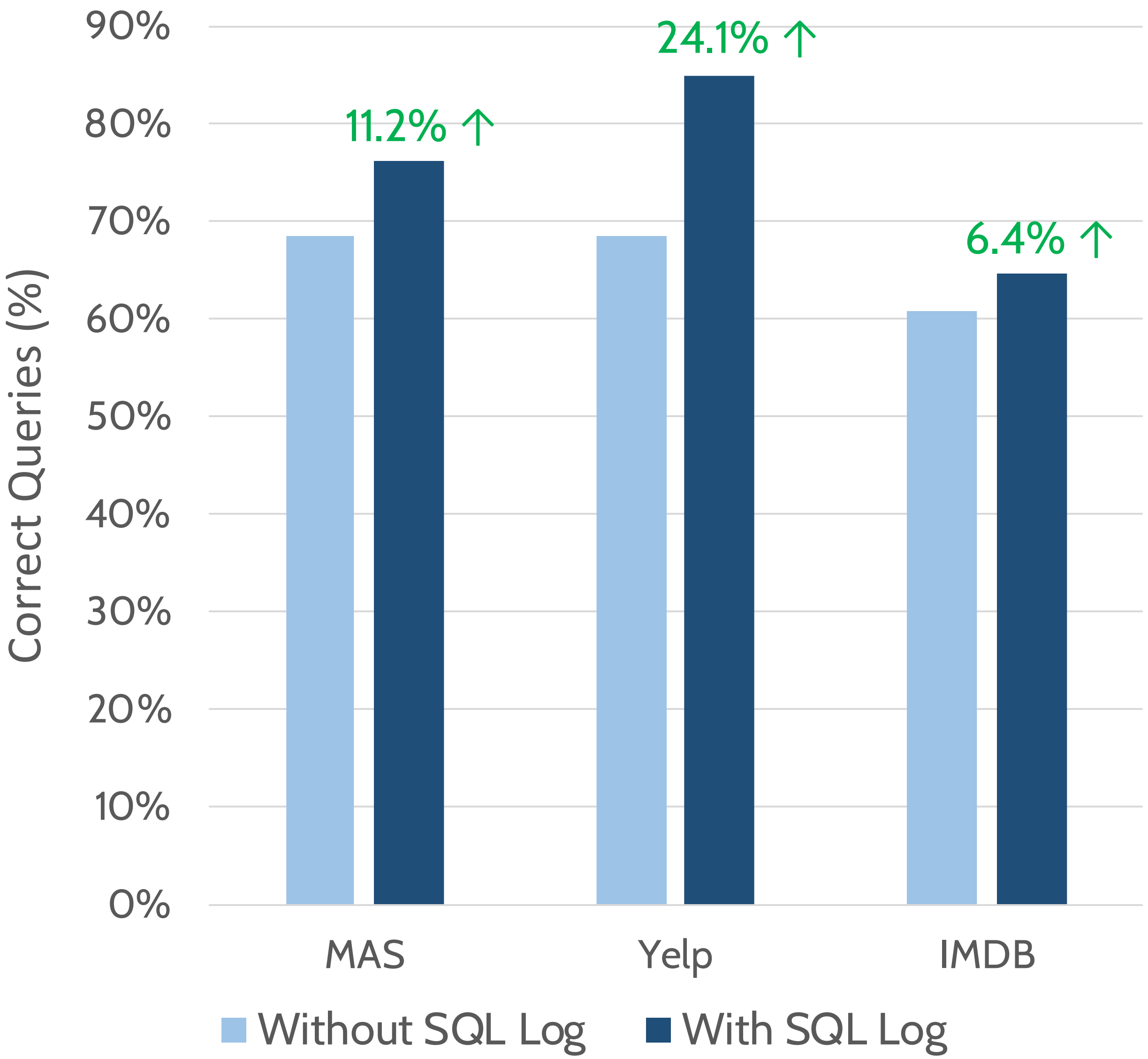
- **Pipeline + Templar** is by far the best-performing
- Augmenting with Templar significantly increases accuracy
- Effects more drastic on Pipeline than NaLIR because of upstream parser issues

Experiments > Keyword Mapping



- **Similar trend** to end-to-end results

Experiments > Join Path Inference



- Results only for **Pipeline + Templar** (effect not as drastic in NaLIR + Templar)
- Modest increases, but most gains from keyword mapping

Conclusion

Contributions

NLQ + = SQL

Query Fragment Graph (QFG)
A model for storing SQL query log info

Applying QFG
to “bridge the semantic gap” between NLQ and SQL

Templar
A system to augment existing NLIDBs with our techniques

Questions, comments, collaborations, etc.

cjbaik@umich.edu

Icon Attributions

- database by iconeu from the Noun Project
- users by Gregor Cresnar from the Noun Project
- Network by mark from the Noun Project
- knowledge database by sahua d from the Noun Project