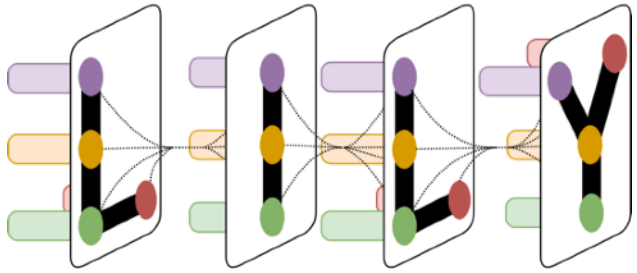
Semantic Evaluation for Text-to-SQL with Distilled Test Suites

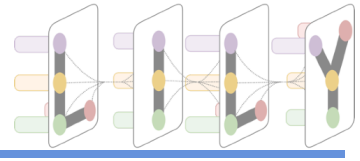
Berkeley NLP Group



Yale NLP Group

Ruiqi Zhong, Tao Yu, and Dan Klein
{ruiqi-zhong, klein}@berkeley.edu, tao.yu@yale.edu

Text-to-SQL Evaluation



Inputs

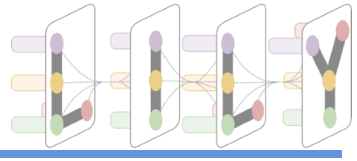
Natural Language

How old is the youngest person from department A?

"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Text-to-SQL Evaluation



Inputs

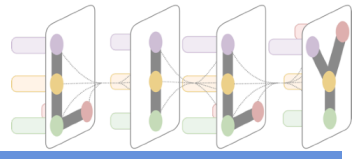
Natural Language *How old is the youngest person from department A?*

"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Output → Predicted SQL `SELECT MIN(Age) from People WHERE Department = 'A'`

Text-to-SQL Evaluation



Inputs

Natural Language *How old is the youngest person from department A?*

"People" Database

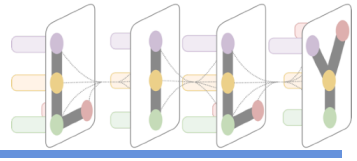
NAME	Age	Department
Alice	26	A
Bob	23	A

Execute SQL on the database

Answer: 23

Output → Predicted SQL `SELECT MIN(Age) from People WHERE Department = 'A'`

Text-to-SQL Evaluation



Inputs

Natural Language *How old is the youngest person from department A?*

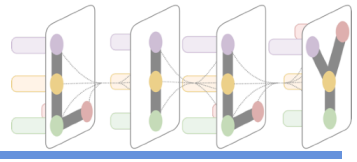
"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Output → Predicted SQL `SELECT MIN(Age) from People WHERE Department = 'A'`

How do we evaluate the predicted SQL?

Exact String Match



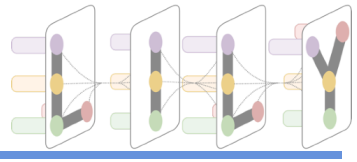
"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Exact String Match



"People" Database

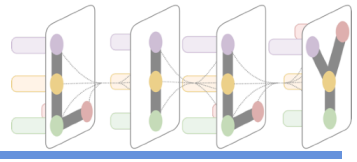
NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Attempt 1: Exact String Match.
gold SQL string == predicted SQL string ?

Exact String Match



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

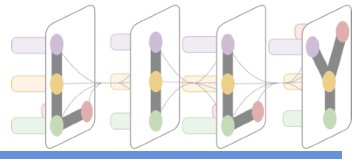
```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Attempt 1: Exact String Match.
gold SQL string == predicted SQL string ?

Predicted SQL

```
SELECT Age from People WHERE Department = 'A'
    ORDER BY Age ASC LIMIT 1
```

Exact String Match



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

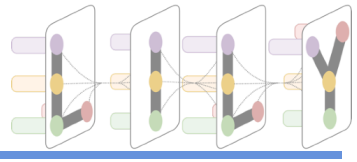
Attempt 1: Exact String Match.
gold SQL string == predicted SQL string ?

Different but Equivalent

Predicted SQL

```
SELECT Age from People WHERE Department = 'A'
    ORDER BY Age ASC LIMIT 1
```

Exact String Match



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Attempt 1: Exact String Match.
gold SQL string == predicted SQL string ?

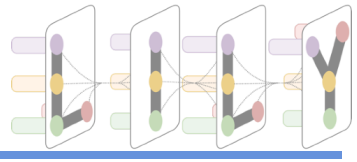
Different but Equivalent

Predicted SQL

```
SELECT Age from People WHERE Department = 'A'
    ORDER BY Age ASC LIMIT 1
```

Totally reasonable prediction.
But gold SQL != predicted SQL

Answer Match



"People" Database

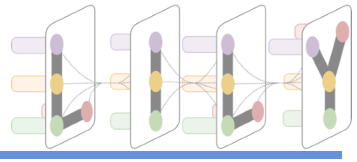
NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Attempt 2: Answer Match.
gold answer == predicted answer ?

Answer Match



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

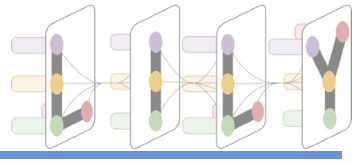
```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Attempt 2: Answer Match.
gold answer == predicted answer ?

Predicted SQL

```
SELECT MIN(Age) from People
```

Answer Match



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

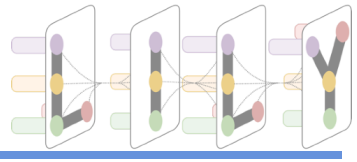
Gold SQL `SELECT MIN(Age) from People WHERE Department = 'A'`

Attempt 2: Answer Match.
gold answer == predicted answer ?

Predicted SQL `SELECT MIN(Age) from People`

Answer: 23

Answer Match



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL `SELECT MIN(Age) from People WHERE Department = 'A'`

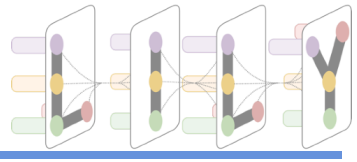
Attempt 2: Answer Match.
gold answer == predicted answer ?

Predicted SQL `SELECT MIN(Age) from People`

Answer: 23

Wrong prediction.
But gold answer == predicted answer

Semantic Correctness



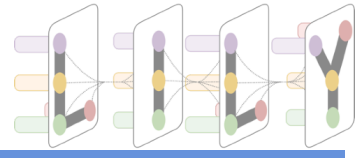
"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Semantic Correctness



"People" Database

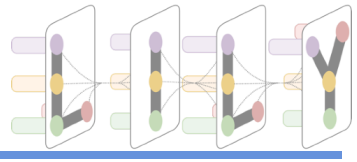
NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Semantic Correctness:
gold answer == predicted answer ?
on all possible databases

Semantic Correctness



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

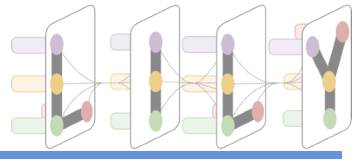
Semantic Correctness:
gold answer == predicted answer ?
on all possible databases

Predicted SQL

```
SELECT Age from People WHERE Department = 'A'
    ORDER BY Age ASC LIMIT 1
```

Correct

Semantic Correctness



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Semantic Correctness:
gold answer == predicted answer ?
on all possible databases

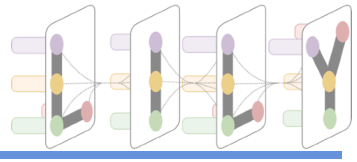
Predicted SQL

```
SELECT MIN(Age) from People
```

Wrong

NAME	Age	Department
Alice	26	A
Bob	23	B

Semantic Correctness



"People" Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Gold SQL

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Semantic Correctness:
gold answer == predicted answer ?
on all possible databases

Predicted SQL

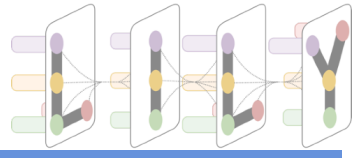
```
SELECT MIN(Age) from People
```

Wrong

NAME	Age	Department
Alice	26	A
Bob	23	B

UNDECIDABLE in general

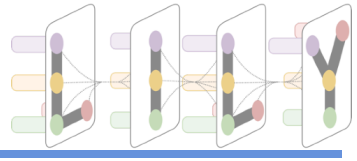
An Online Judge Analog



Problem Description

Return the sum of all even numbers in the array

An Online Judge Analog



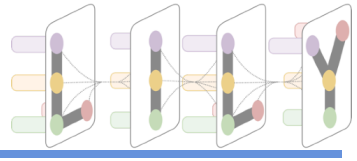
Problem Description

Return the sum of all even numbers in the array

Your Solution

```
def even_sum(arr):
    return sum(a for a in arr if a % 2 == 0)
```

An Online Judge Analog



Problem Description

Return the sum of all even numbers in the array

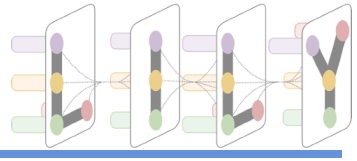
Your Solution

```
def even_sum(arr):
    return sum(a for a in arr if a % 2 == 0)
```

Test Suite

```
Input: [1, 2, 3]
Expected Output: 2

Input: [3, 5, 7]
Expected Output: 0

Input: [2, 7, 8, 2, 1]
Expected Output: 12

[More test cases omitted]
```

An Online Judge Analog



Problem Description

Return the sum of all even numbers in the array

Your Solution

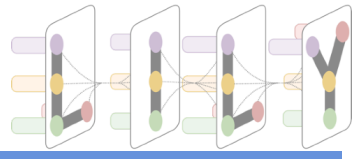
```
def even_sum(arr):
    return sum(a for a in arr if a % 2 == 0)
```

Test Suite

```
Input: [1, 2, 3]
Expected Output: 2

Input: [3, 5, 7]
Expected Output: 0

Input: [2, 7, 8, 2, 1]
Expected Output: 12

[More test cases omitted]
```

Solution is considered correct if it passes the entire test suite

An Online Judge Analog



Problem Description

Return the sum of all even numbers in the array

Your Solution

Predicted SQL

```
def even_sum(arr):
    return sum(a for a in arr if a % 2 == 0)
```

Test Suite

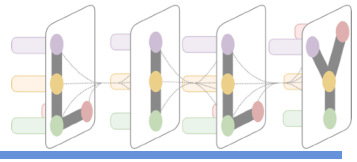
Database

```
Input: [1, 2, 3]
Expected Output: 2

Input: [3, 5, 7]
Expected Output: 0

Input: [2, 7, 8, 2, 1]
Expected Output: 12

[More test cases omitted]
```

Answer

Solution is considered correct if it passes the entire test suite

An Online Judge Analog



Natural Language

How old is the youngest person from department A?

Predicted SQL

```
SELECT MIN(Age) from People
    WHERE Department = 'A'
```

Test Suite:
a Set of Databases

NAME	Age	Department
Alice	26	A
Bob	23	A

Expected Answer: 23

NAME	Age	Department
Alice	26	A
Bob	23	B

Expected Answer: 26

[More Database-Answers Omitted]

Solution is considered correct if it passes the entire test suite

An Online Judge Analog

Natural Language

How old is the youngest person from department A?

Predicted SQL

```
SELECT MIN(Age) from People
    WHERE Department = 'A'
```

Test Suite:
a Set of Databases

NAME	Age	Department
Alice	26	A
Bob	23	A

Expected Answer: 23

NAME	Age	Department
Alice	26	A
Bob	23	B

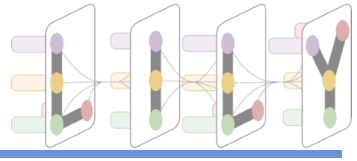
Expected Answer: 26

[More Database-Answers Omitted]

Solution is considered correct if it passes the entire test suite

How do we (automatically) construct this?

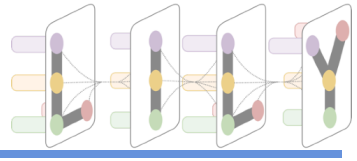
Criteria for Test Suites



Criterion 1:
Fast to run

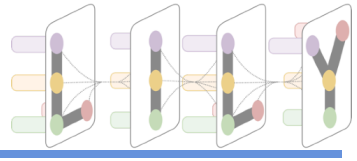
Cannot enumerate all possible databases (smaller than a certain length) …

Criteria for Test Suites



Criterion 1: Fast to run	Cannot enumerate all possible databases (smaller than a certain length) …
Criterion 2: Code Coverage	Effectively tests the use of every clause and constants

Criteria for Test Suites



Criterion 1:
Fast to run

Cannot enumerate all possible databases (smaller than a certain length) …

Criterion 2:
Code Coverage

Effectively tests the use of every clause and constants

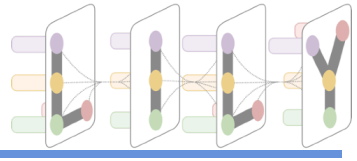
Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

not covered

Database

NAME	Age	Department
Alice	26	A
Bob	23	A

Criteria for Test Suites



Criterion 1:
Fast to run

Cannot enumerate all possible databases (smaller than a certain length) …

Criterion 2:
Code Coverage

Effectively tests the use of every clause and constants

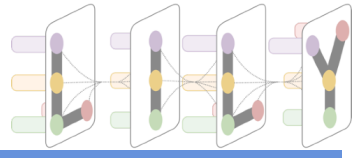
Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

not covered

Database

NAME	Age	Department
Alice	26	A
Bob	23	B

Criteria for Test Suites



Criterion 1:
Fast to run

Criterion 2:
Code Coverage

Search Objectives

Cannot enumerate all possible databases (smaller than a certain length) …

Effectively tests the use of every clause and constants

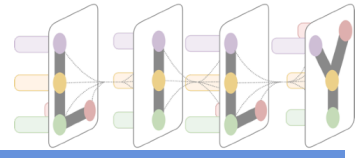
Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

not covered

Database

NAME	Age	Department
Alice	26	A
Bob	23	B

Define Code Coverage



Gold

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Test Suite

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

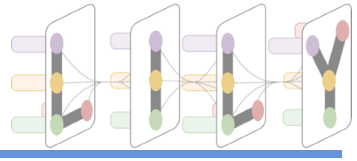
Correct Answer
26

DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer
23

Define Code Coverage



Neighbor 1

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

modify

Gold

```
SELECT MIN(Age) from People WHERE Department = 'A'
```

Test Suite

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

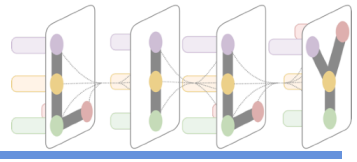
Correct Answer
26

DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer
23

Define Code Coverage



Neighbor 1 `SELECT MIN(Age) from People ~~WHERE Department = 'A'~~`

modify

Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

Test Suite

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

Correct Answer
26

Neighbor 1 Answer
23

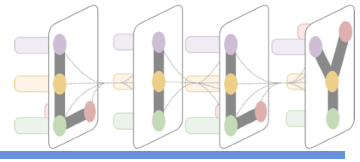
DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer
23

Neighbor 1 Answer
23

Define Code Coverage



Neighbor 1 `SELECT MIN(Age) from People ~~WHERE Department = 'A'~~` discriminated by DB 1

modify

Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

Correct Answer 26 != Neighbor 1 Answer 23

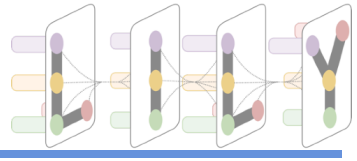
Test Suite

DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer 23 == Neighbor 1 Answer 23

Define Code Coverage



Neighbor 1 `SELECT MIN(Age) from People WHERE Department = 'A'`

Neighbor 2 `SELECT MIN(Age) from People WHERE Department = 'B'`

Neighbor 3 `SELECT MAX(Age) from People WHERE Department = 'A'`

modify

Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

Test Suite

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

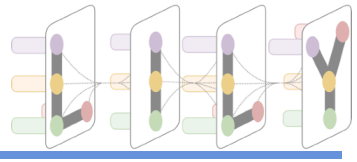
Correct Answer
26

DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer
23

Define Code Coverage



Neighbor 1 `SELECT MIN(Age) from People WHERE Department = 'A'` DB 1

Neighbor 2 `SELECT MIN(Age) from People WHERE Department = 'B'` discriminated by DB 1 and DB 2

Neighbor 3 `SELECT MAX(Age) from People WHERE Department = 'A'` DB 2

modify

Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

Test Suite

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

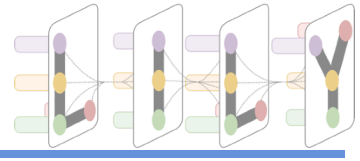
Correct Answer
26

DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer
23

Define Code Coverage



Neighbor 1 `SELECT MIN(Age) from People ~~WHERE Department = 'A'~~` DB 1

Neighbor 2 `SELECT MIN(Age) from People WHERE Department = 'B'` discriminated by DB 1 and DB 2

Neighbor 3 `SELECT MAX(Age) from People WHERE Department = 'A'` DB 2

modify

Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

For each neighbor, at least one database in the test suite discriminates the neighbor.

Test Suite

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

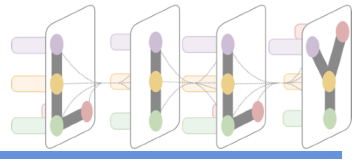
Correct Answer
26

DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer
23

Define Code Coverage



Neighbor 1 `SELECT MIN(Age) from People ~~WHERE Department = 'A'~~` DB 1

Neighbor 2 `SELECT MIN(Age) from People WHERE Department = 'B'` discriminated by DB 1 and DB 2

Neighbor 3 `SELECT MAX(Age) from People WHERE Department = 'A'` DB 2

modify

Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

For each neighbor, at least one database in the test suite discriminates the neighbor.

Test Suite

DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

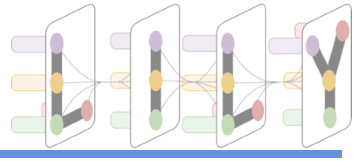
Correct Answer
26

DB 2

NAME	Age	Department
Alice	26	A
Bob	23	A

Correct Answer
23

Define Code Coverage



Neighbor 1 `SELECT MIN(Age) from People ~~WHERE Department = 'A'~~` DB 1

Neighbor 2 `SELECT MIN(Age) from People WHERE Department = 'B'` discriminated by DB 1

Neighbor 3 `SELECT MAX(Age) from People WHERE Department = 'A'` None

modify

Gold `SELECT MIN(Age) from People WHERE Department = 'A'`

Neighbor 3 cannot be discriminated by any database.

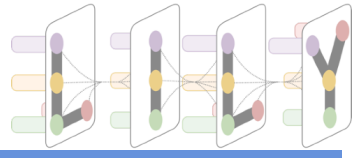
DB 1

NAME	Age	Department
Alice	26	A
Bob	23	B

Correct Answer
26

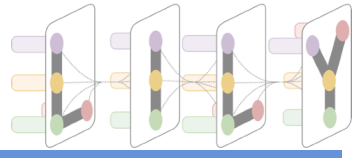
Test Suite

Constructing a Test Suite



Optimization Objective Find a small set of databases that can discriminate all neighbors.

Constructing a Test Suite

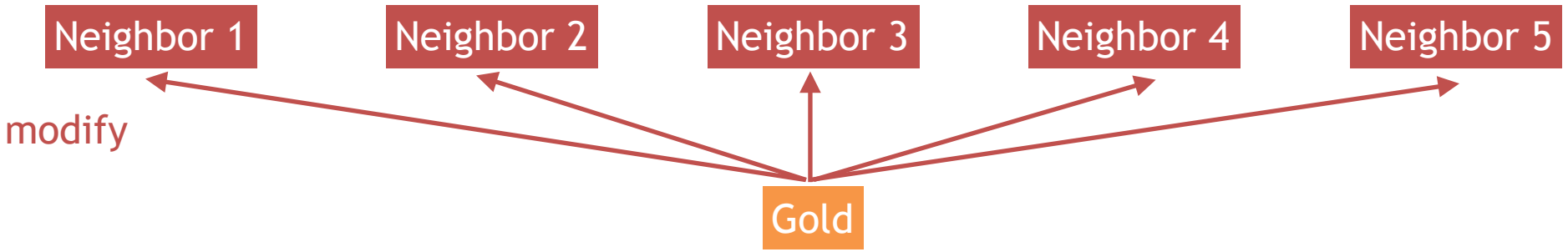


Optimization Objective Find a small set of databases that can discriminate all neighbors.

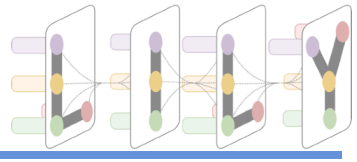
Gold

Constructing a Test Suite

Optimization Objective Find a small set of databases that can discriminate all neighbors.



Constructing a Test Suite



Optimization Objective Find a small set of databases that can discriminate all neighbors.

Neighbor 1 Neighbor 2 Neighbor 3 Neighbor 4 Neighbor 5

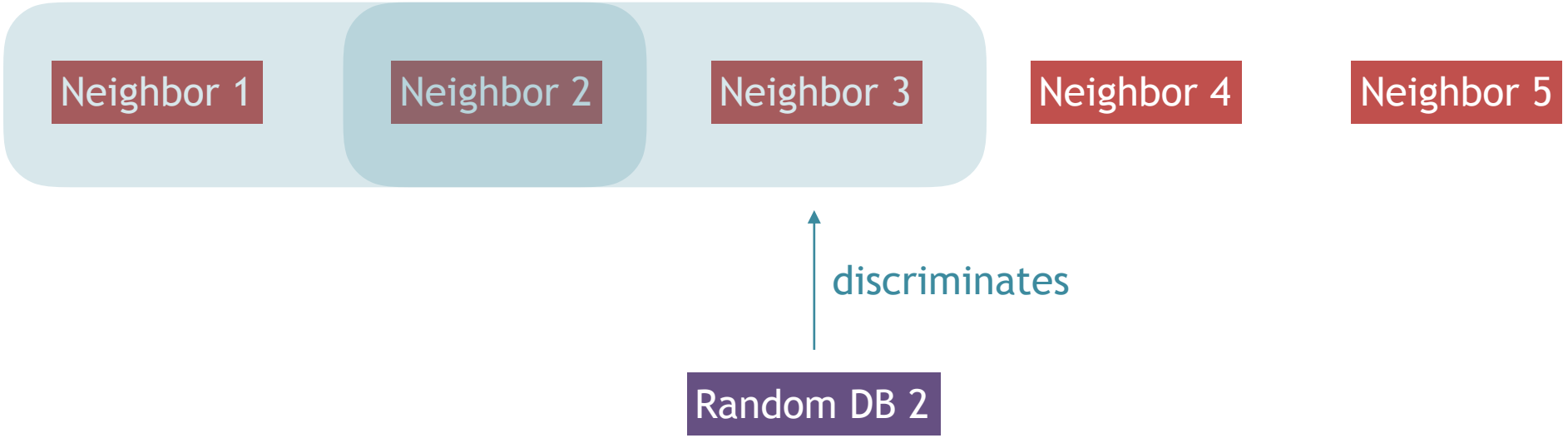
Random DB 1 Random DB 2 Random DB 3

Constructing a Test Suite

Optimization Objective Find a small set of databases that can discriminate all neighbors.

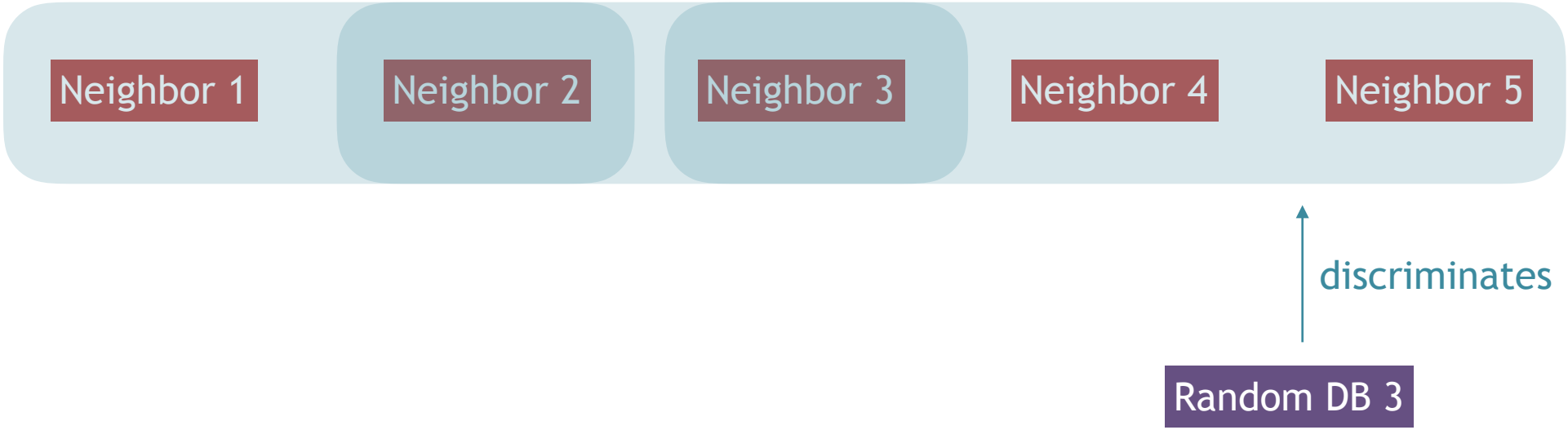
Constructing a Test Suite

Optimization Objective Find a small set of databases that can discriminate all neighbors.



Constructing a Test Suite

Optimization Objective Find a small set of databases that can discriminate all neighbors.

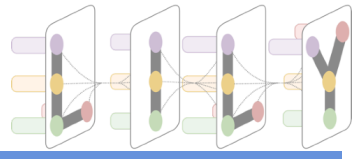


Constructing a Test Suite

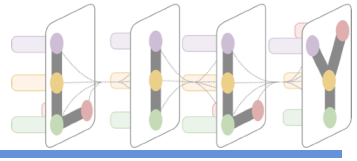
Optimization Objective Find a small set of databases that can discriminate all neighbors.

"Smallest Cover" Test Suite = { Random DB 1 , Random DB 3 }

Data

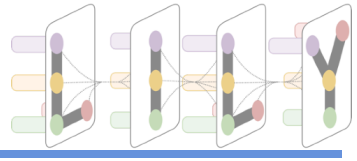


Data



SPIDER development set: 1034 pairs of English-SQL parallel data.

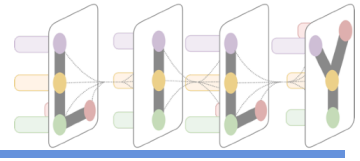
Data



SPIDER development set: 1034 pairs of English-SQL parallel data.

Official metric “Exact Set Match”: whether the predicted SQL contains the same set of clauses as the gold.

Data



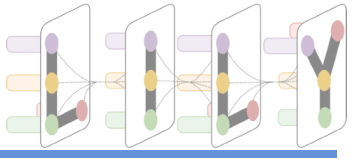
SPIDER development set: 1034 pairs of English-SQL parallel data.

Official metric “Exact Set Match”: whether the predicted SQL contains the same set of clauses as the gold.

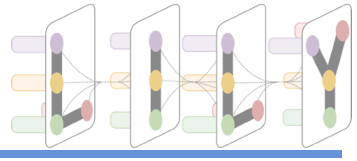
21 Leaderboard submissions with Exact Set Match ranging from 40% to 65%.

Rank	Model	Dev	Test
1 May 02, 2020	RATSQL v3 + BERT (DB content used) *Microsoft Research* (Wang and Shin et al., ACL '20) code	69.7	**65.6**
2 Sep. 8, 2020	YCSQL + BERT (DB content used) *Anonymous*	-	65.3
3 Sep. 8, 2020	ShadowGNN (DB content used) *Anonymous*	-	64.8
4 May 31, 2020	AuxNet + BART (DB content used) *Anonymous*	70.0	61.9
4 Dec 13, 2019	RATSQL v2 + BERT (DB content used) *Microsoft Research* (Wang and Shin et al., ACL '20) code	65.8	61.9

No Errors Made

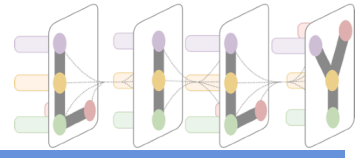


No Errors Made



False Negative	The predicted query is in fact semantically correct, but our test suite considers it to be wrong.

No Errors Made

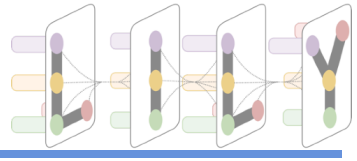


False Negative

The predicted query is in fact semantically correct, but our test suite considers it to be wrong.

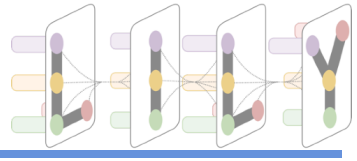
This provably never happens.

No Errors Made



False Negative	The predicted query is in fact semantically correct, but our test suite considers it to be wrong.
	This provably never happens.
False Positive	The predicted query is in fact semantically wrong, but our test suite considers it to be correct.

No Errors Made



False Negative

The predicted query is in fact semantically correct, but our test suite considers it to be wrong.

This provably never happens.

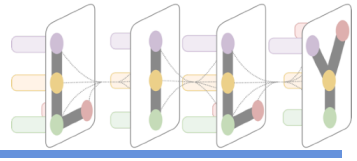
False Positive

The predicted query is in fact semantically wrong, but our test suite considers it to be correct.

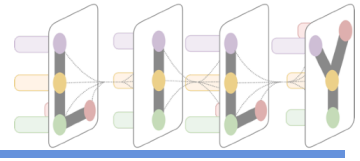
Manually examined 100 random examples where exact set match considers them wrong but test suite considers them correct.

Our metric judges correctly in all these cases.

Official Metric in Hindsight



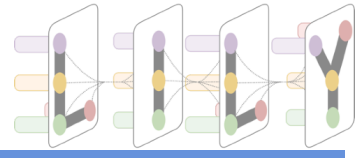
Official Metric in Hindsight



“False negative” fraction across 21 submissions.

Mean	Standard Deviation	Max
2.6%	1.7%	8.1%

Official Metric in Hindsight

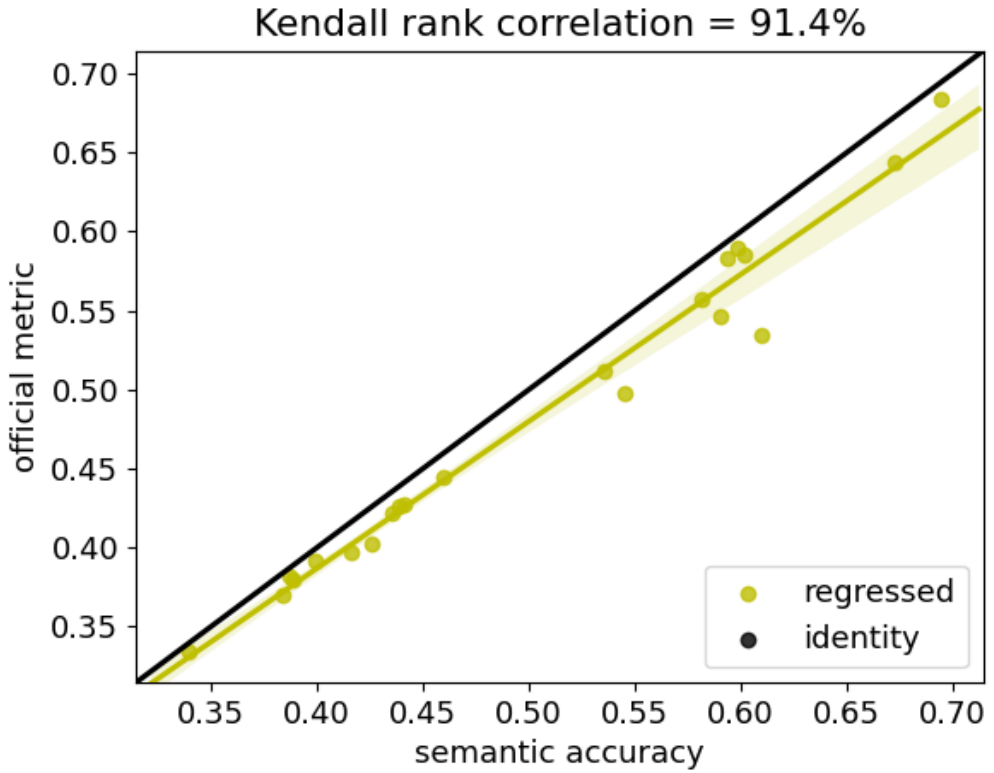


“False negative” fraction across 21 submissions.

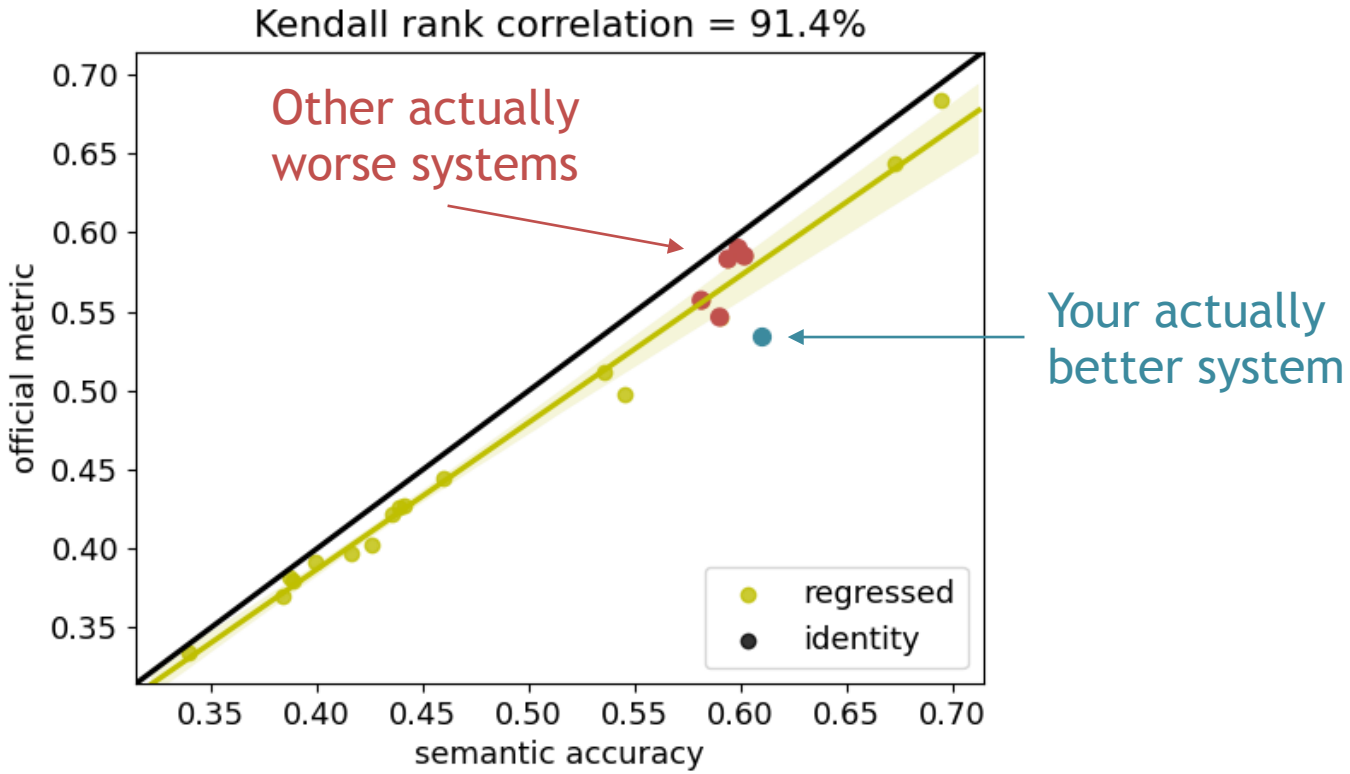
Mean	Standard Deviation	Max
2.6%	1.7%	8.1%

Makes non-negligible fraction of errors.

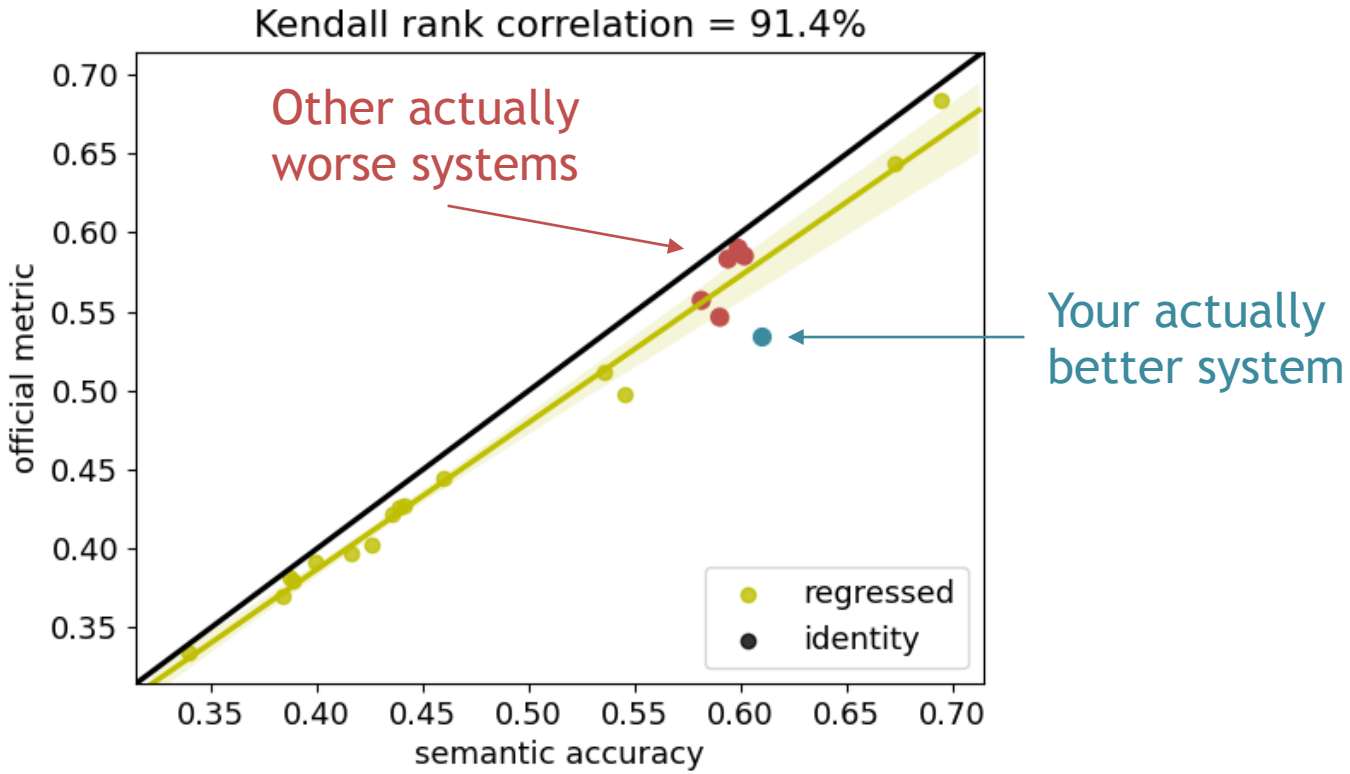
Official Metric in Hindsight



Official Metric in Hindsight

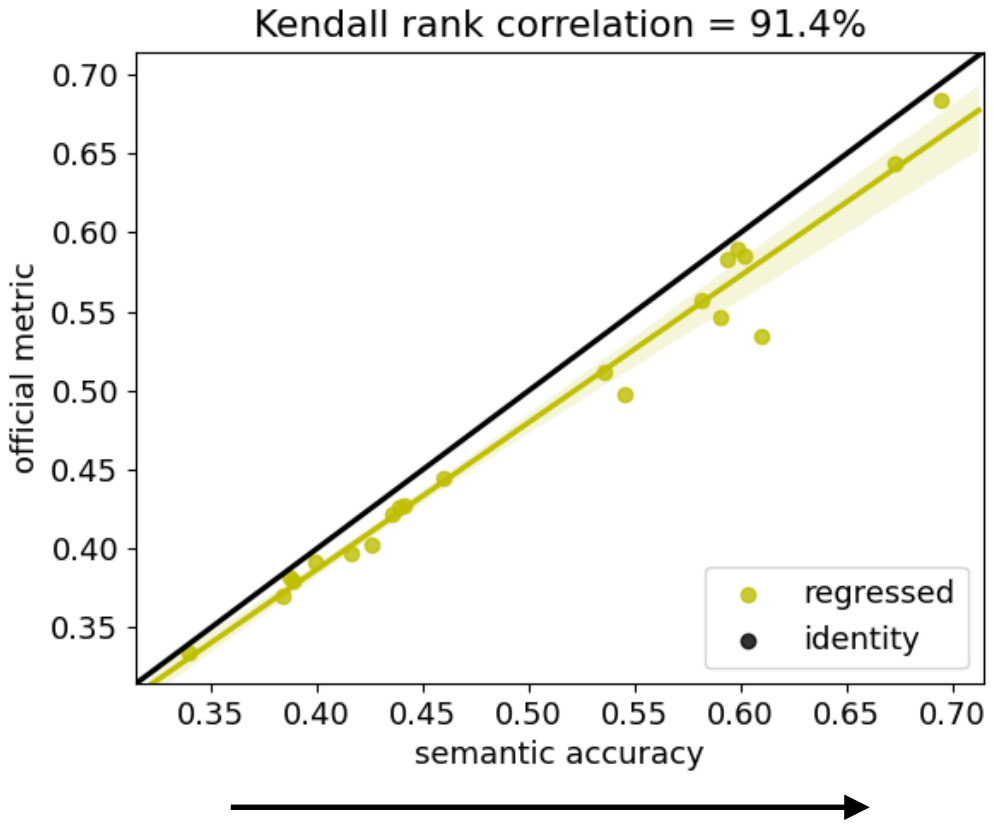


Official Metric in Hindsight



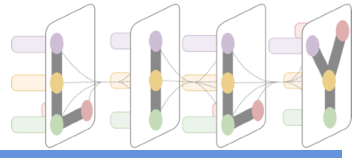
Does not reflect all semantic improvements.

Official Metric in Hindsight



Correlates less as systems become better.

New Metric Release



Test suite and our new metric implementation is now publicly available
for eleven Text-to-SQL datasets:
SPIDER, SParC, CoSQL, Academic, Advising, ATIS, GeoQuery,
IMDB, Restaurants, Scholar and Yelp

Test suite evaluation is now the official metric of
SPIDER, SParC and CoSQL leaderboards.

Takeaways

- Matching exact logical forms can be too strict and poorly reflects semantic accuracy.
- Evaluating on multiple inputs (test suite) can approximate semantic accuracy better.
- Optimizing code coverage can lead to high quality test suites.

Thank you!

Paper: https://arxiv.org/abs/2010.02840

Code: https://github.com/ruiqi-zhong/TestSuiteEval