Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing



Victoria Lin*   Richard Socher   Caiming Xiong

salesforce research

Tables and **databases** are commonly used data structures that power a variety of downstream applications.

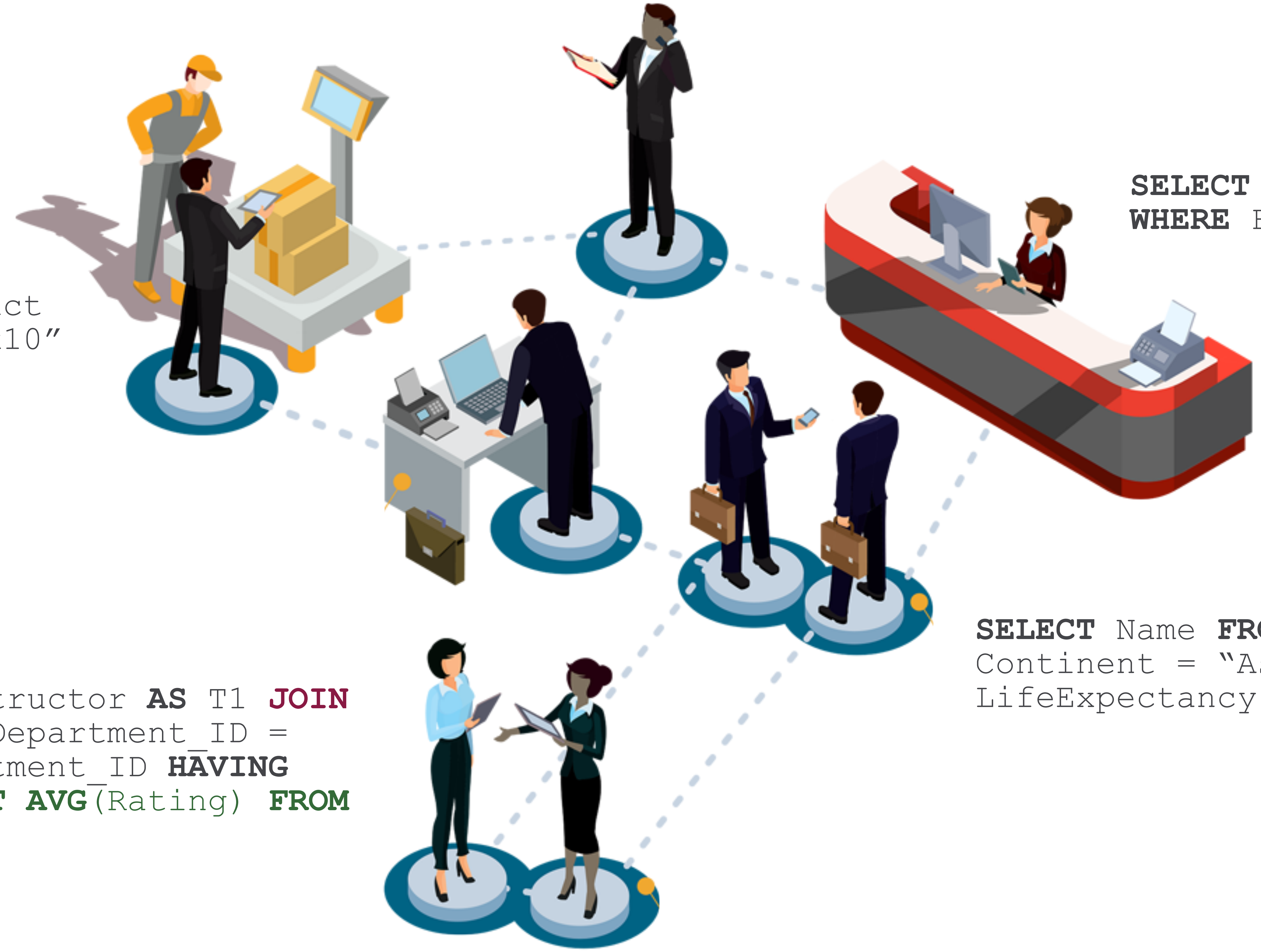
The **Internet** contains billions of public tables. And more tabular data can be found on **corporate intranets** and **personal devices**.

Natural Language Interface to Databases

Traditionally, users access databases using **structured query language (SQL)**.

Natural Language Interface to Databases

Traditionally, users access databases using **structured query language (SQL)**.



```
SELECT Arriving_Time FROM Flights
WHERE Flight_Number = “CZ327”
```

```
SELECT Quantity FROM Product
WHERE Name = “Hoverboard x10”
```

```
SELECT Name FROM Country WHERE
Continent = “Asia” ORDER BY
LifeExpectancy LIMIT 1
```

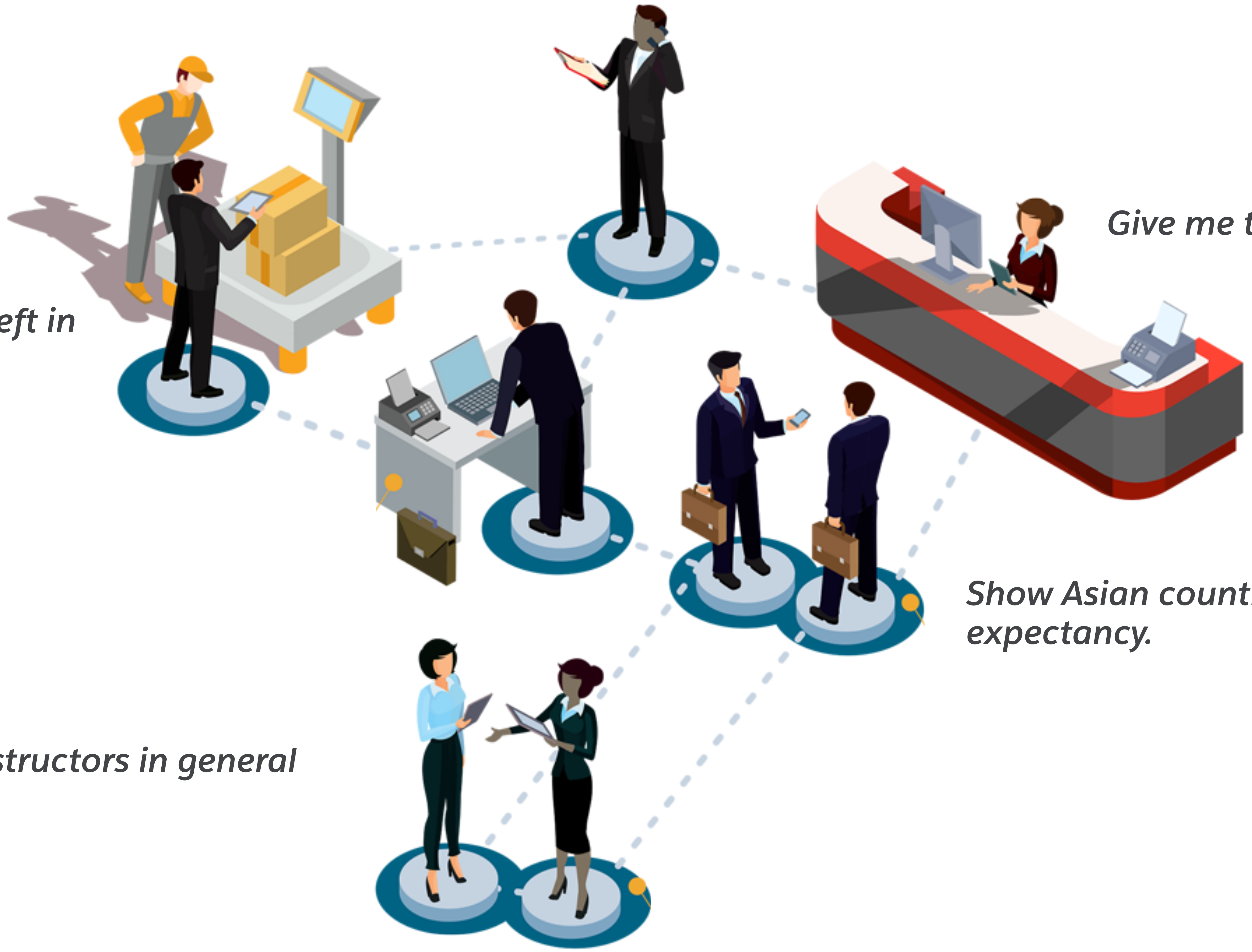
```
SELECT T2.name FROM Instructor AS T1 JOIN
Department AS T2 ON T1.Department_ID =
T2.ID GROUP BY T1.Department_ID HAVING
AVG(T1.Rating) > (SELECT AVG(Rating) FROM
Instructor)
```

Natural Language Interface to Databases

Our goal is to learn semantic parsers that map natural language utterances to **executable** SQL queries for **any database.**

Natural Language Interface to Databases

Our goal is to learn semantic parsers that map natural language utterances to **executable** SQL queries for **any database.**



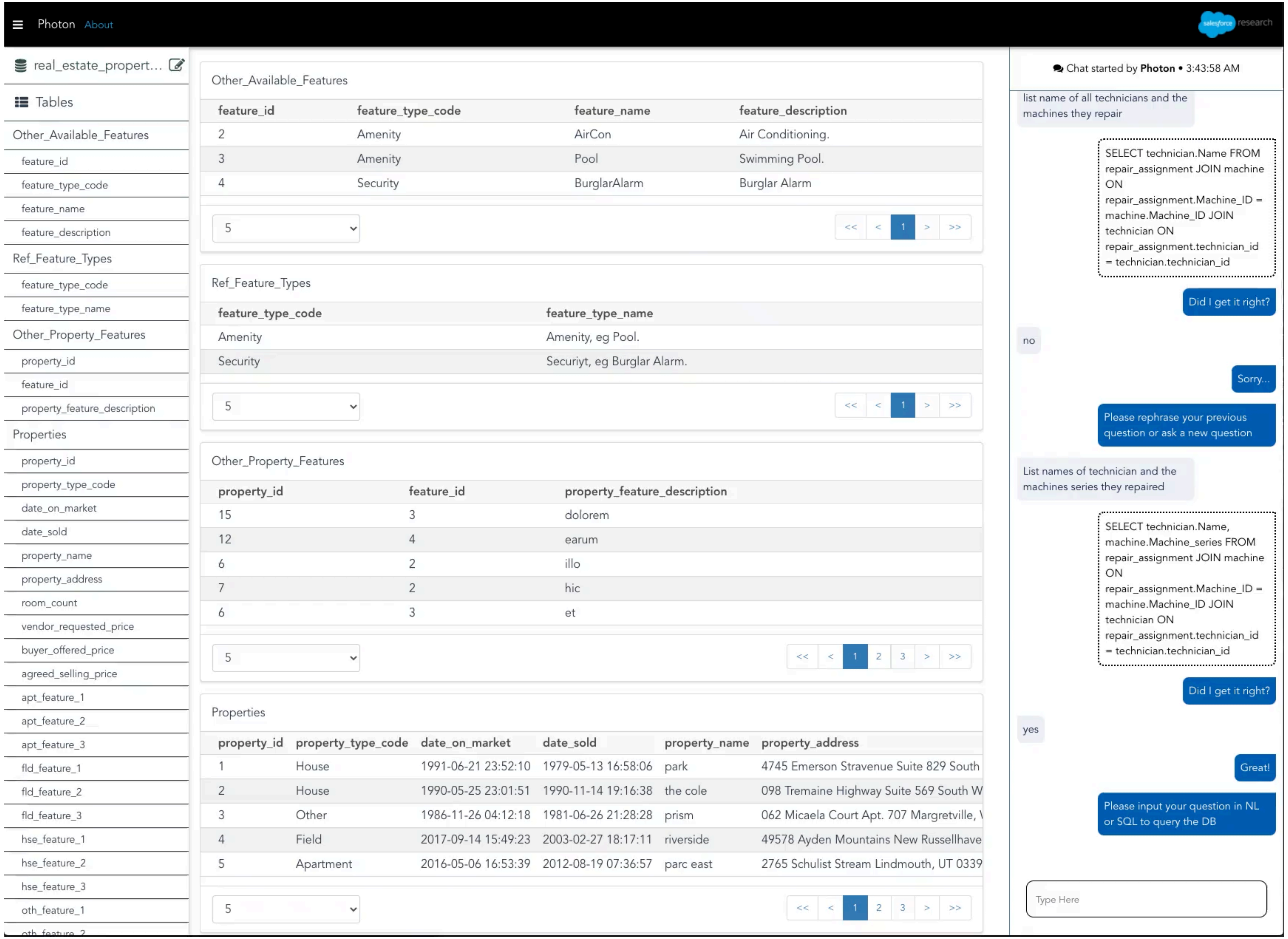
Give me the arriving time of “CZ327”.

How many “Hoverboard x10” are left in stock?

Show Asian countries ordered by life expectancy.

Which departments have instructors in general rated above average?

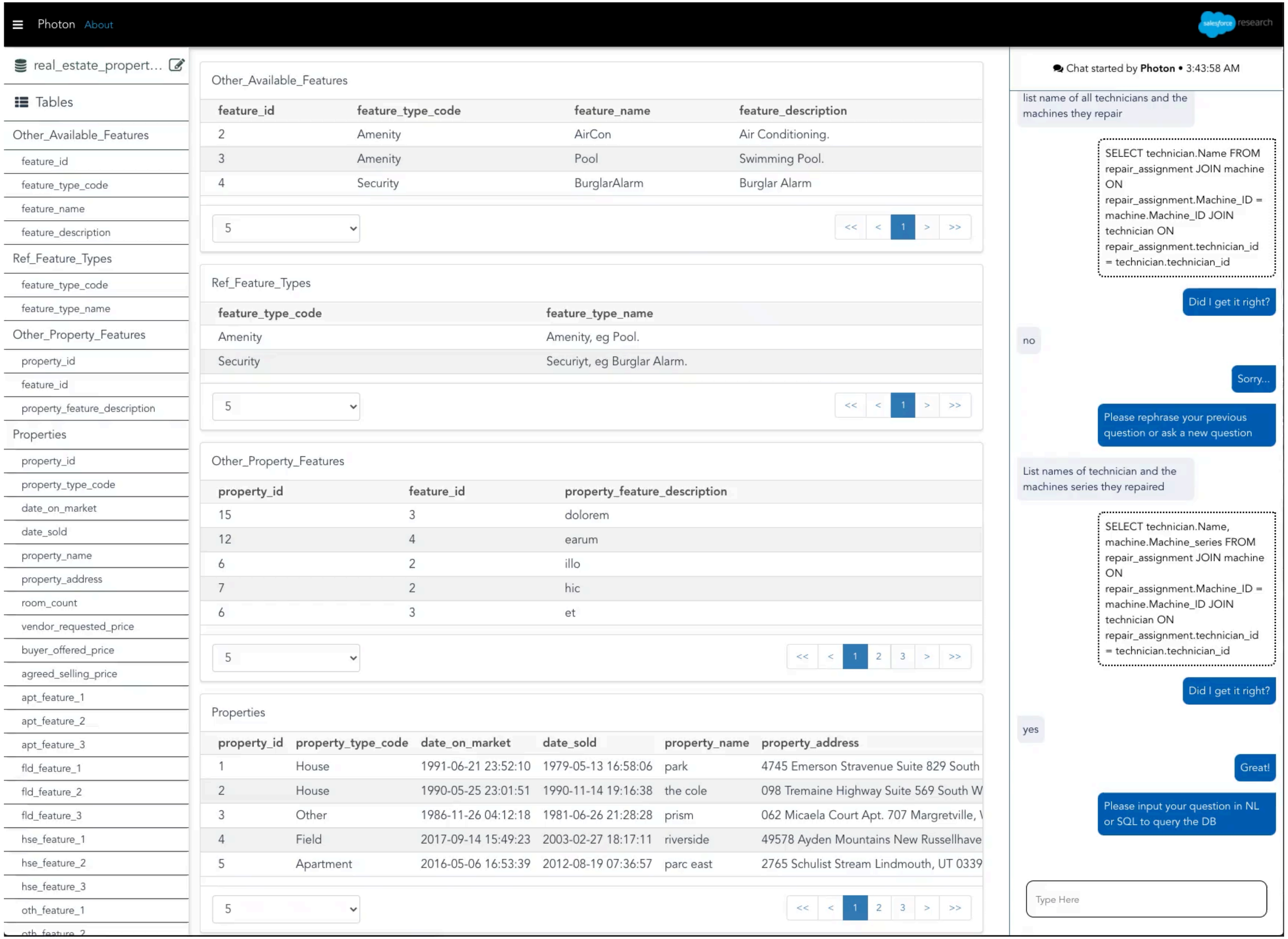
salesforce research https://naturalsql.com



- Text-to-SQL translation
- Query arbitrary subsets of tables in any database
- Support table join and other complex SQL operators

research

https://naturalsql.com



- Text-to-SQL translation
- Query arbitrary subsets of tables in any database
- Support table join and other complex SQL operators

Relational Database Terms

User Profile

UserID	Name	Nationality	PartitionID	# Followers	…
103041	Smith, John	Canada	P10	3	…
103042	Hanks, Tom	United States	P11	16.6M	…
103043	Obama, Barack	United States	P11	127M	…
…	…	…	…	…	…

💡 A table represents a entity type (or event type).

Relational Database Terms

User Profile ← Table Name

UserID	Name	Nationality	PartitionID	# Followers	…
103041	Smith, John	Canada	P10	3	…
103042	Hanks, Tom	United States	P11	16.6M	…
103043	Obama, Barack	United States	P11	127M	…
…	…	…	…	…	…

Relational Database Terms

User Profile

Table Header

UserID	Name	Nationality	PartitionID	# Followers	...
103041	Smith, John	Canada	P10	3	…
103042	Hanks, Tom	United States	P11	16.6M	…
103043	Obama, Barack	United States	P11	127M	…
…	…	…	…	…	…

Relational Database Terms

User Profile

UserID	**Name**	**Nationality**	**PartitionID**	**# Followers**	**...**
103041	Smith, John	Canada	P10	3	…
103042	Hanks, Tom	United States	P11	16.6M	…
103043	Obama, Barack	United States	P11	127M	…
…	…	…	…	…	…

Column Name

Relational Database Terms

User Profile

Row

UserID	**Name**	**Nationality**	**PartitionID**	**# Followers**	**...**
103041	Smith, John	Canada	P10	3	...
103042	Hanks, Tom	United States	P11	16.6M	...
103043	Obama, Barack	United States	P11	127M	...
...	...	...	...	...	...

💡 A row is an instantiation of the entity/event.

Relational Database Terms

User Profile

Column/field

UserID	**Name**	**Nationality**	**PartitionID**	**# Followers**	**...**
103041	Smith, John	Canada	P10	3	…
103042	Hanks, Tom	United States	P11	16.6M	…
103043	Obama, Barack	United States	P11	127M	…
…	…	…	…	…	…

Relational Database Terms

User Profile

UserID	**Name**	**Nationality**	**PartitionID**	**# Followers**	**...**
103041	Smith, John	Canada	P10	3	…
103042	Hanks, Tom	United States	P11	16.6M	…
103043	Obama, Barack	United States	P11	127M	…
…	…	…	…	…	…

Integer String Integer

Relational Database Terms

User Profile

UserID	**Name**	**Nationality**	**PartitionID**	**# Followers**	**…**
103041	Smith, John	Canada	P10	3	…
103042	Hanks, Tom	United States	P11	16.6M	…
103043	Obama, Barack	United States	P11	127M	…
…	…	…	…	…	…

Primary Key

Relational Database Terms

User Profile

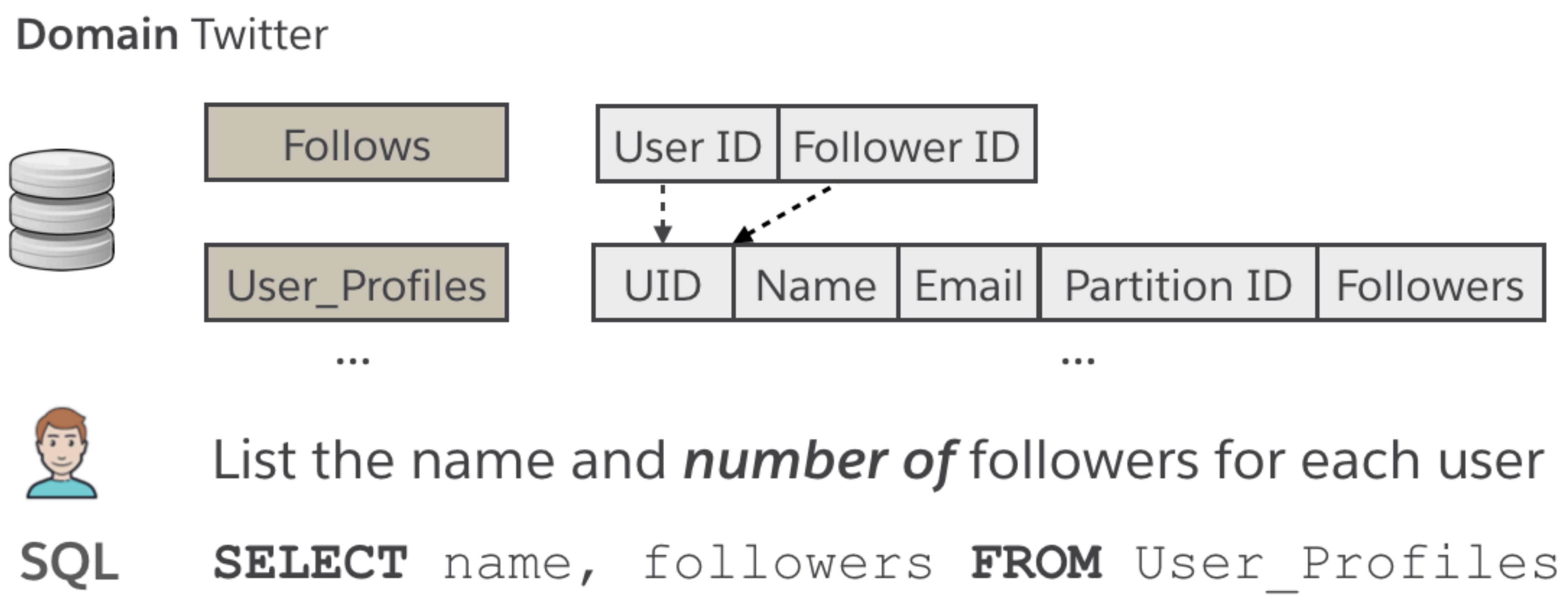
UserID	**Name**	**Nationality**	**PartitionID**	**# Followers**	**…**
103041	Smith, John				
103042	Hanks, Tom				
103043	Obama, Barack				
…	…				

Follows

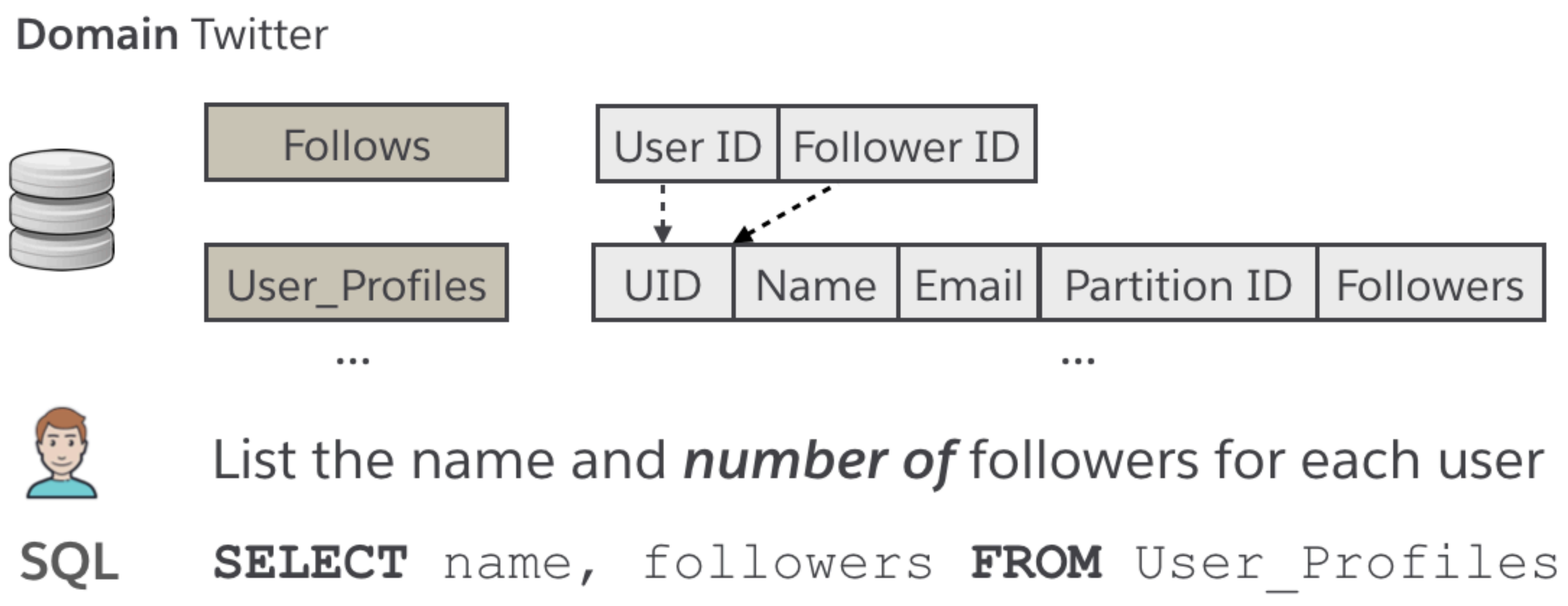
UserID	**FollowerID**	**…**
103041	103042	…
103042	103041	…
…	…	…

Foreign Key

Text-to-SQL Semantic Parsing

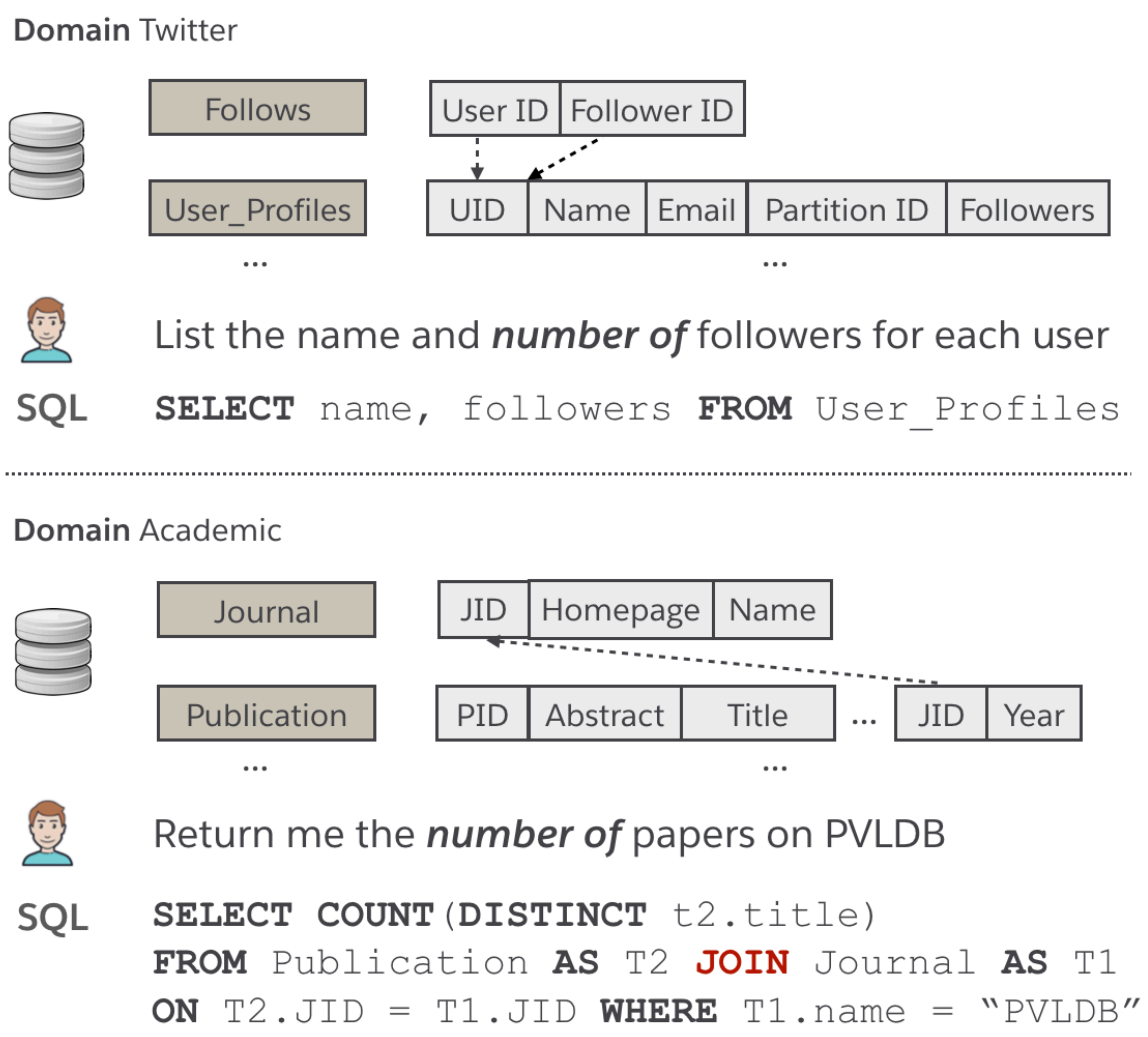


Text-to-SQL Semantic Parsing



Strong/Full supervision

Cross Domain Text-to-SQL Semantic Parsing



Challenges

Challenge 1: Questions with similar intent may map to very different SQL logical forms when issued to different DBs.

Domain Twitter

Follows	User ID	Follower ID			
User_Profiles	UID	Name	Email	Partition ID	Followers
...	...				

List the name and ***number of*** followers for each user

SQL
```
SELECT name, followers FROM User_Profiles
```

Cross-Database

Domain Academic

Journal	JID	Homepage	Name			
Publication	PID	Abstract	Title	...	JID	Year
...	...					

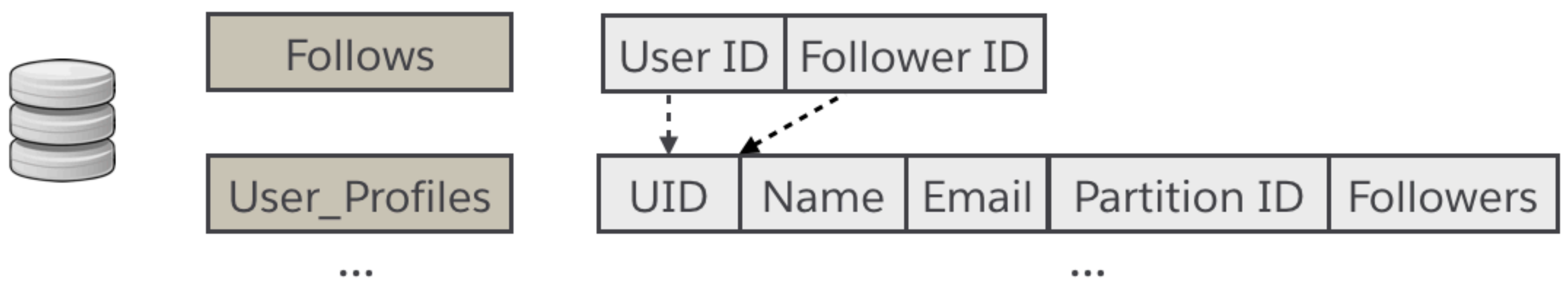
Return me the ***number of*** papers on PVLDB

SQL
```
SELECT COUNT(DISTINCT t2.title)
FROM Publication AS T2 JOIN Journal AS T1
ON T2.JID = T1.JID WHERE T1.name = "PVLDB"
```

Challenges

Challenge 2: The questions often mention domain-specific entities.

Domain Twitter

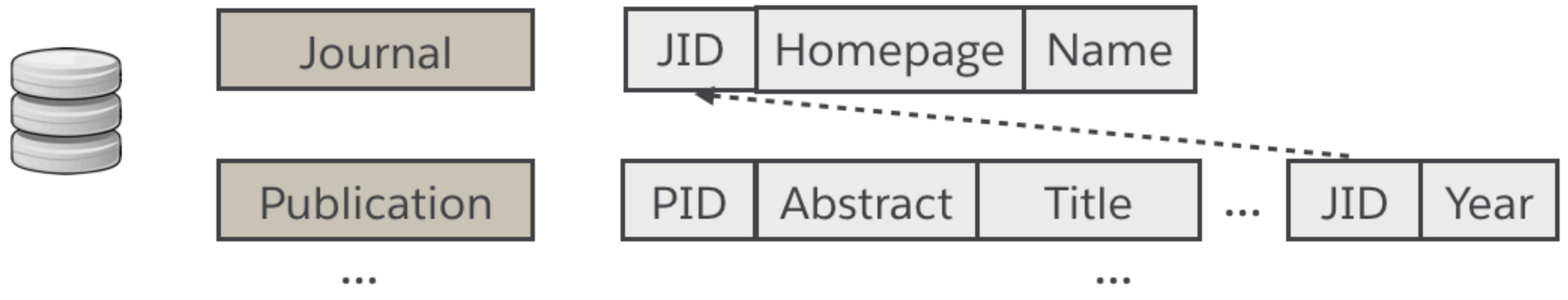


List the name and ***number of*** followers for each user

SQL

```
SELECT name, followers FROM User_Profiles
```

Domain Academic



Return me the ***number of*** papers on PVLDB

SQL

```
SELECT COUNT(DISTINCT t2.title)
FROM Publication AS T2 JOIN Journal AS T1
ON T2.JID = T1.JID WHERE T1.name = "PVLDB"
```

Domain Diversity

Observations

Observation 1: It is important to contextualize the question and the database (DB), similar to the setup in machine reading comprehension

Domain Twitter

Follows	User ID	Follower ID			
User_Profiles	UID	Name	Email	Partition ID	Followers
...	...				

List the name and ***number of*** followers for each user

SQL
```
SELECT name, followers FROM User_Profiles
```

Domain Academic

Journal	JID	Homepage	Name			
Publication	PID	Abstract	Title	...	JID	Year
...	...					

Return me the ***number of*** papers on PVLDB

SQL
```
SELECT COUNT(DISTINCT t2.title)
FROM Publication AS T2 JOIN Journal AS T1
ON T2.JID = T1.JID WHERE T1.name = "PVLDB"
```

Observations

Domain Twitter

Follows	User ID	Follower ID

User_Profiles	UID	Name	Email	Partition ID	Followers

... ...

List the name and ***number of*** followers for each user

SQL

```
SELECT name, followers FROM User_Profiles
```

Domain Academic

Journal	JID	Homepage	Name

Publication	PID	Abstract	Title	...	JID	Year

... ...

Return me the ***number of*** papers on PVLDB

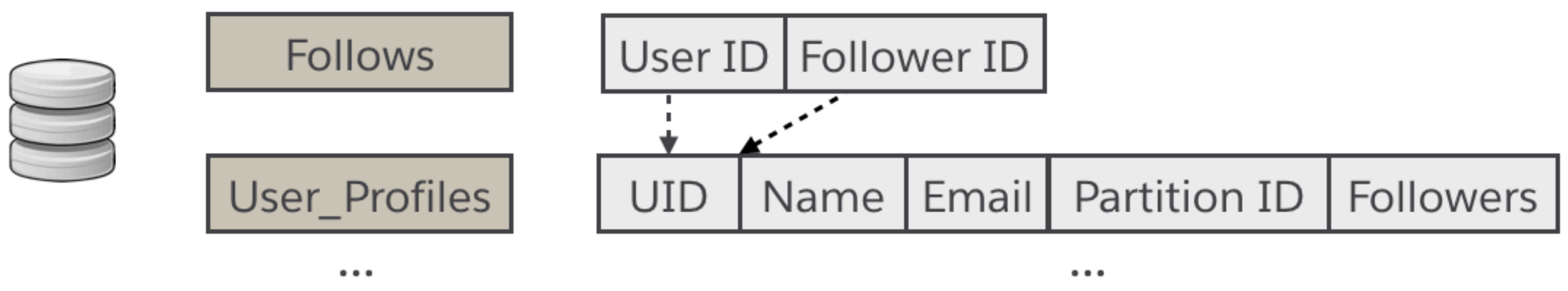
SQL

```
SELECT COUNT(DISTINCT t2.title)
FROM Publication AS T2 JOIN Journal AS T1
ON T2.JID = T1.JID WHERE T1.name = "PVLDB"
```

Observation 2: Database understanding should take into account both the DB schema and the DB content

Observations

Domain Twitter

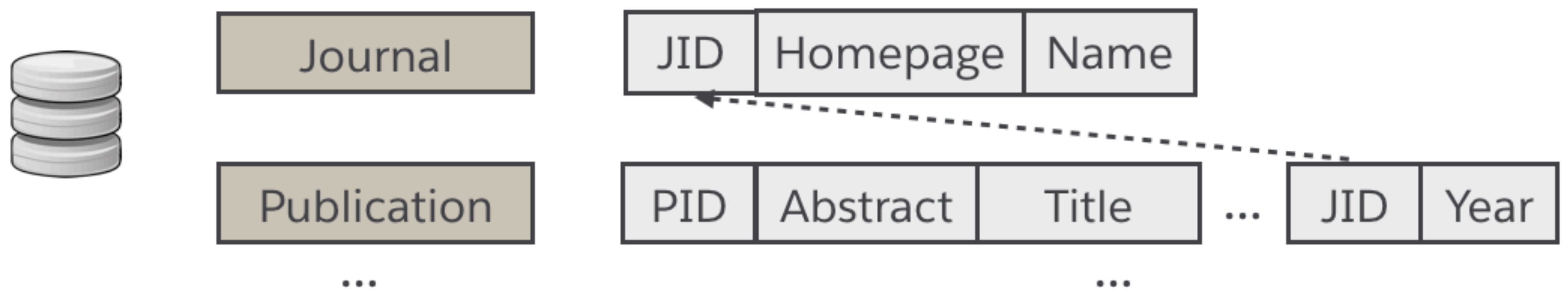


List the name and ***number of*** followers for each user

SQL

```
SELECT name, followers FROM User_Profiles
```

Domain Academic



Return me the ***number of*** papers on PVLDB

SQL

```
SELECT COUNT(DISTINCT t2.title)
FROM Publication AS T2 JOIN Journal AS T1
ON T2.JID = T1.JID WHERE T1.name = "PVLDB"
```

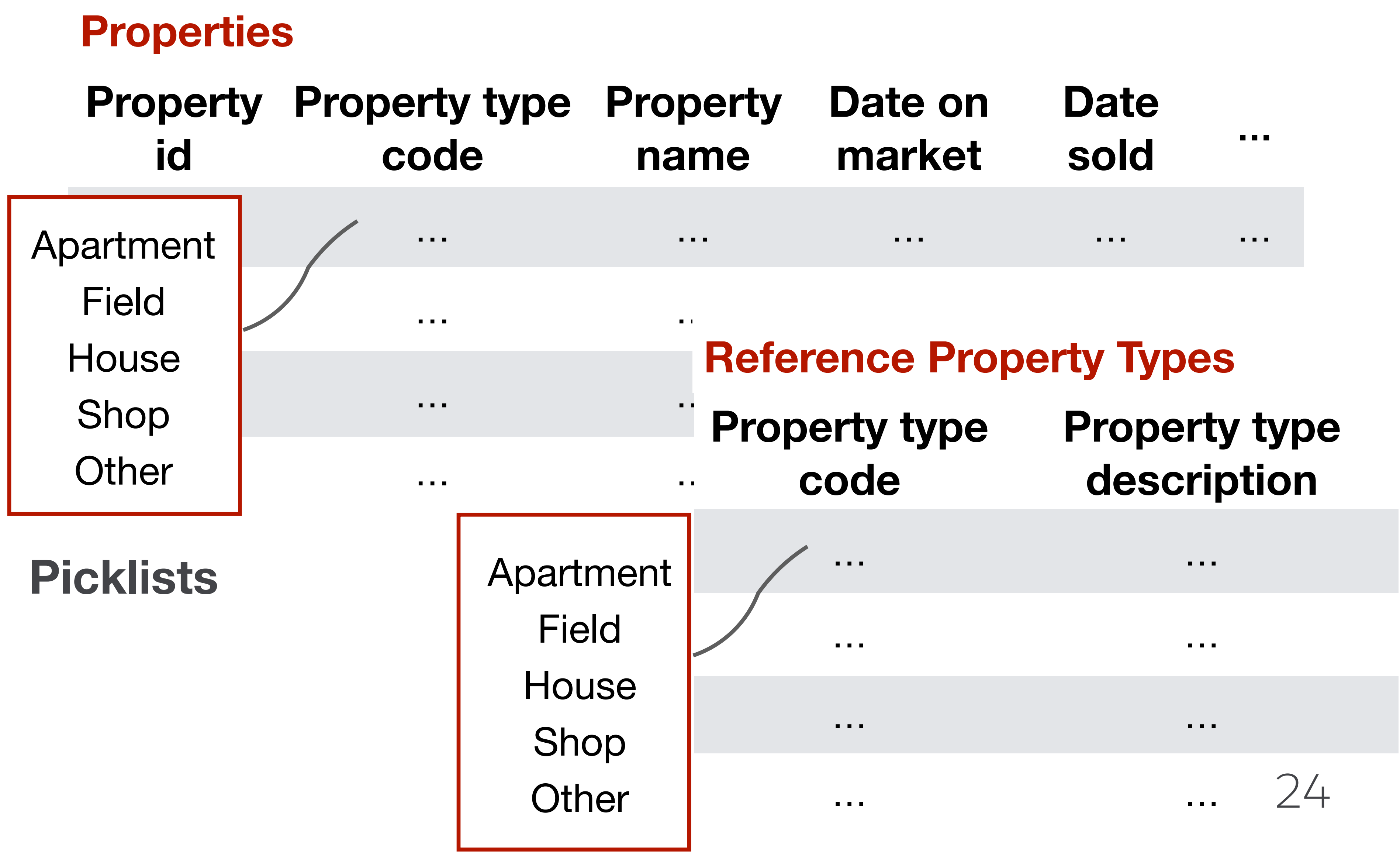
Observation 3: Most "rare entities" mentioned in the question correspond to **tables, fields,** or **DB cells**

Problem Setup

Question

Database Schema

Show names of properties that are either houses or apartments



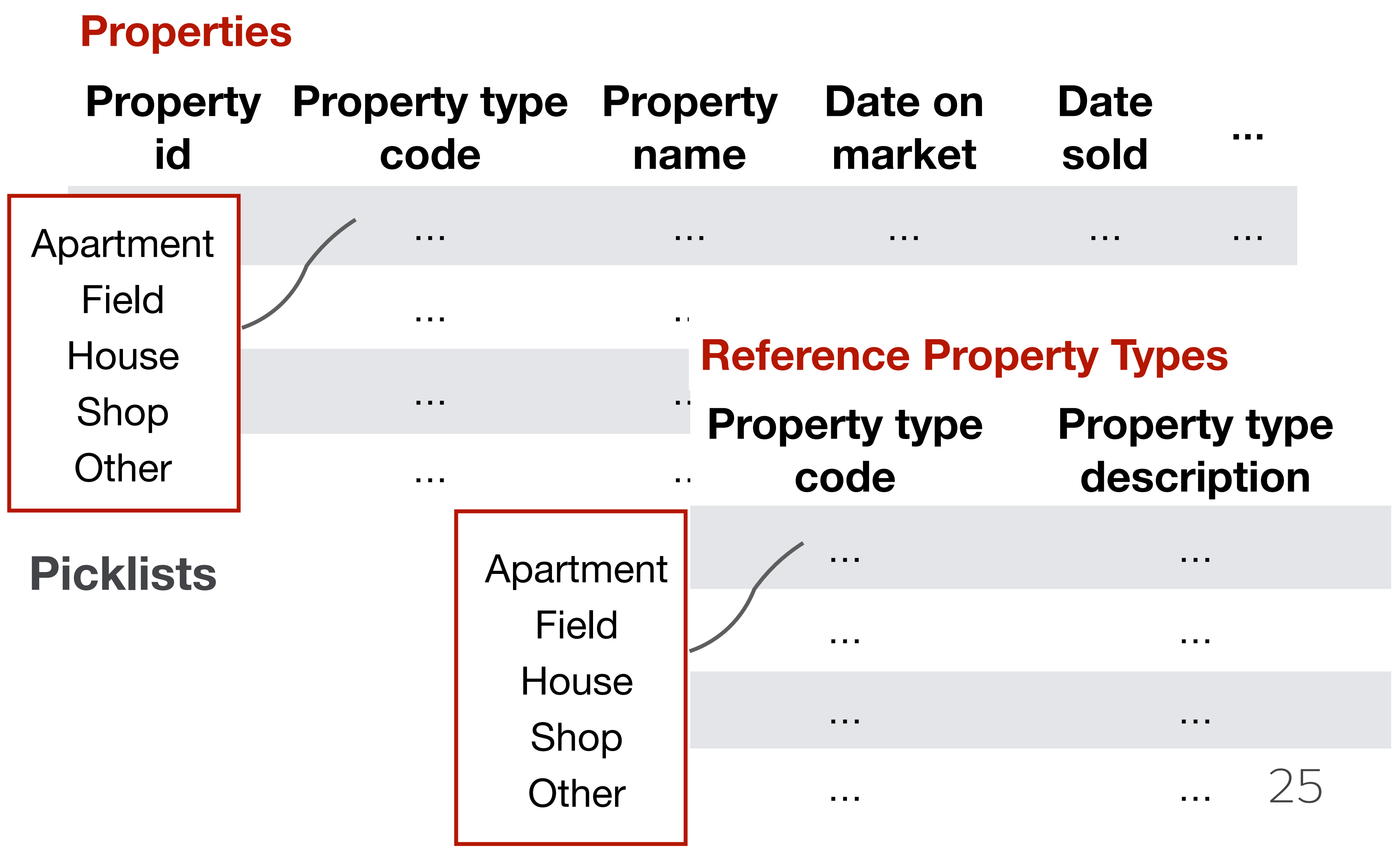
Problem Setup

Research Question:

How can we learn a representation that effectively captures the language grounding of an input question, the DB schema and the DB content?

Question | Database Schema

Show names of properties that are either houses or apartments

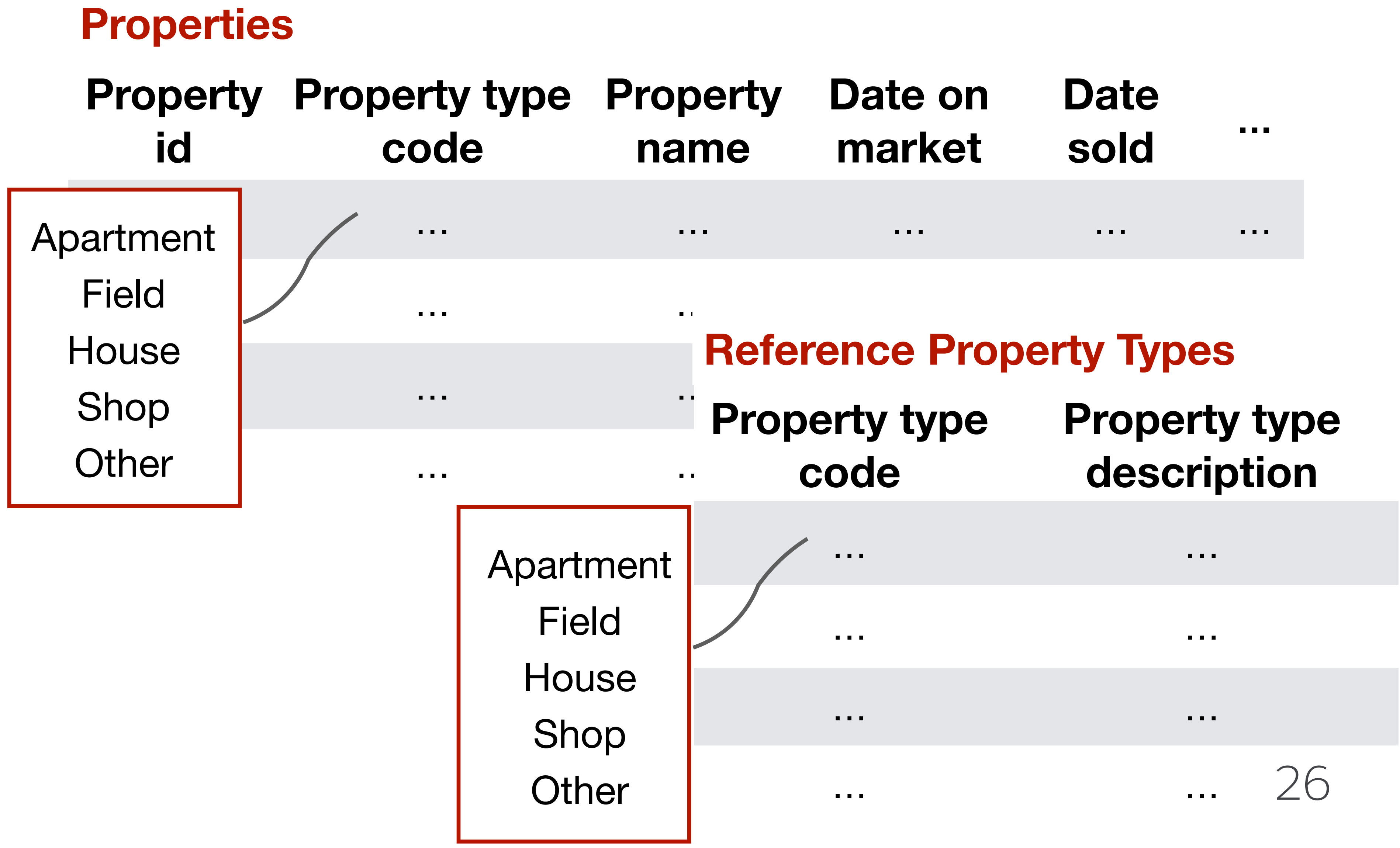


Joint Textual-Tabular Data Encoding

Serialize Table Header/DB Schema

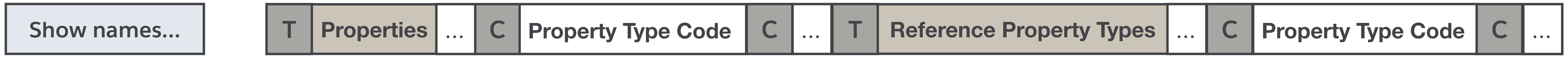
Show names... | Properties | ... | Property Type Code | ... | Reference Property Types | ... | Property Type Code | ...

Show names of properties that are either houses or apartments

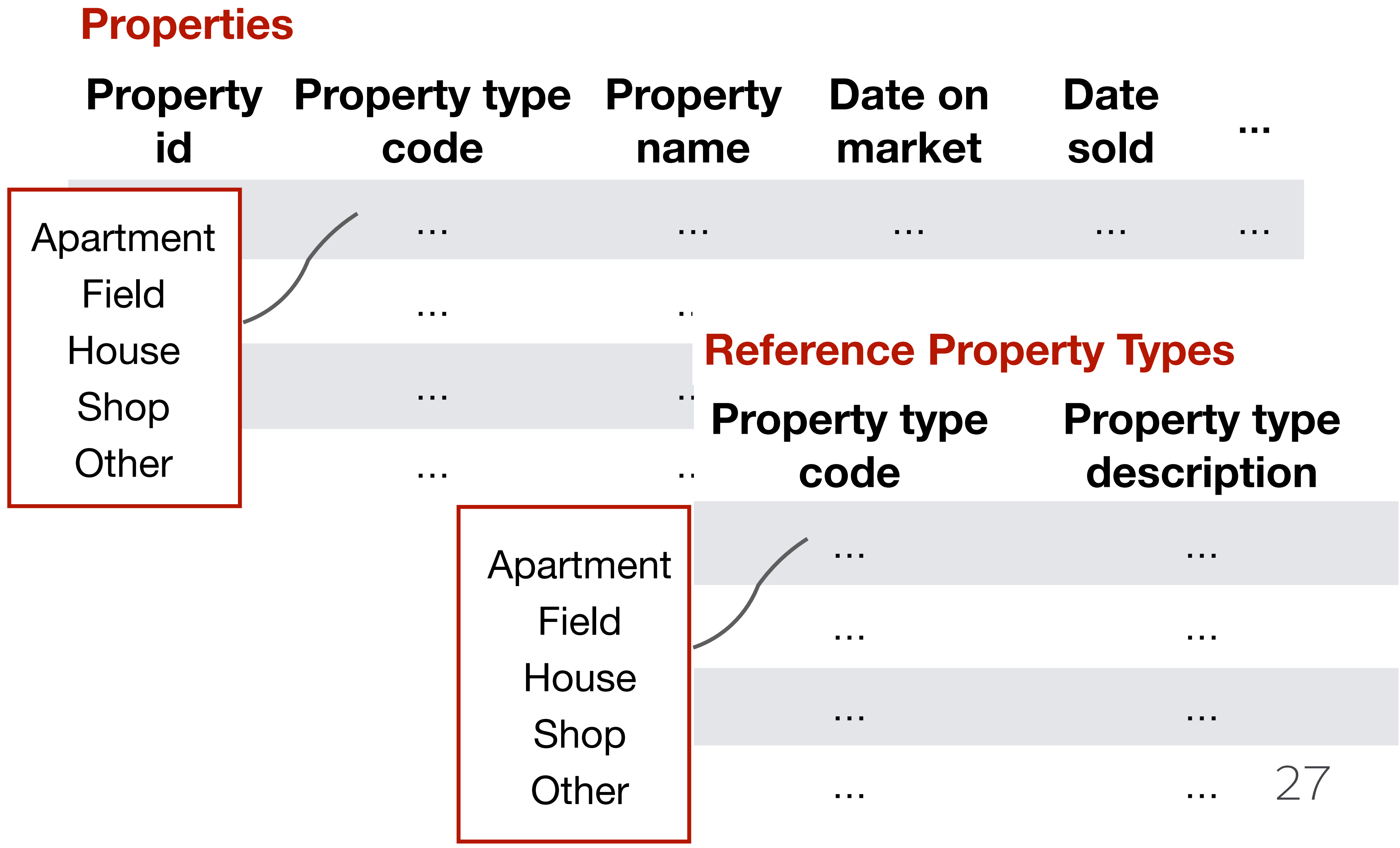


Joint Textual-Tabular Data Encoding

Serialize Table Header/DB Schema

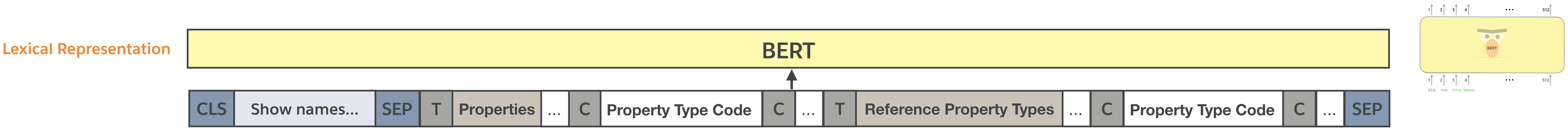


Show names of properties that are either houses or apartments



Joint Textual-Tabular Data Encoding

Serialize Table Header/DB Schema



Show names of properties that are either houses or apartments

Properties

Property id	Property type code	Property name	Date on market	Date sold	...
	...	...	...	...	...
	...	..			
	...	..			
	...	..			

Apartment
Field
House
Shop
Other

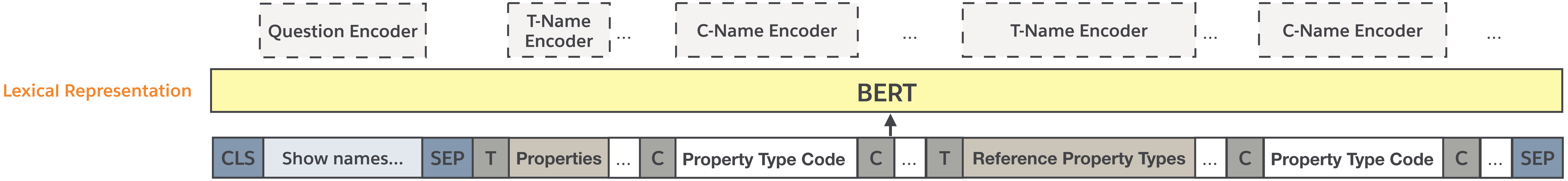
Reference Property Types

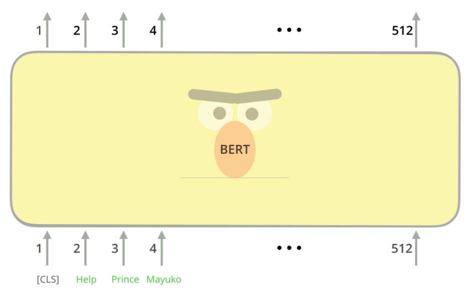
Property type code	Property type description
...	...
...	...
...	...
...	...

Apartment
Field
House
Shop
Other

Joint Textual-Tabular Data Encoding

Component Encoding Layers





Show names of properties that are either houses or apartments

Properties

Property id	Property type code	Property name	Date on market	Date sold	...
	...	...	...	...	...
	...	..			
	...	..			
	...	..			

Apartment
Field
House
Shop
Other

Reference Property Types

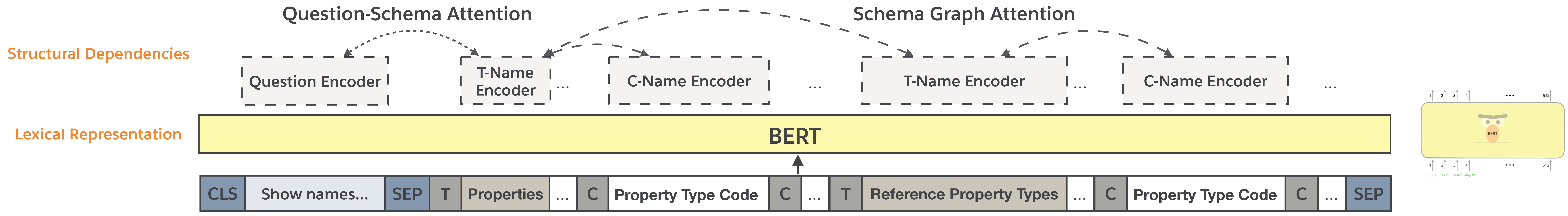
Property type code	Property type description
...	...
...	...
...	...
...	...

Apartment
Field
House
Shop
Other

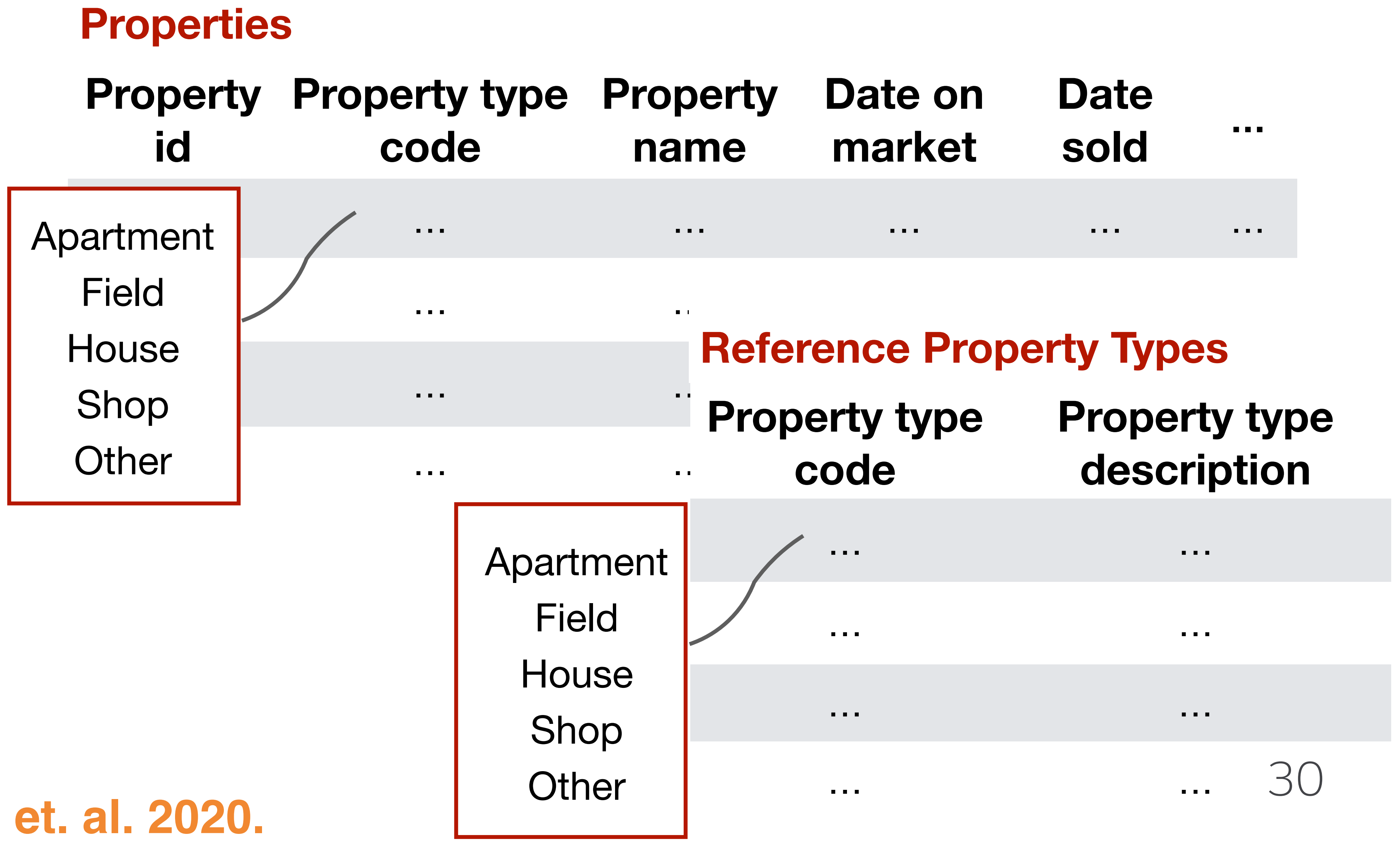
Previous work: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. Wang et. al. 2020.

Joint Textual-Tabular Data Encoding

Attention Layers



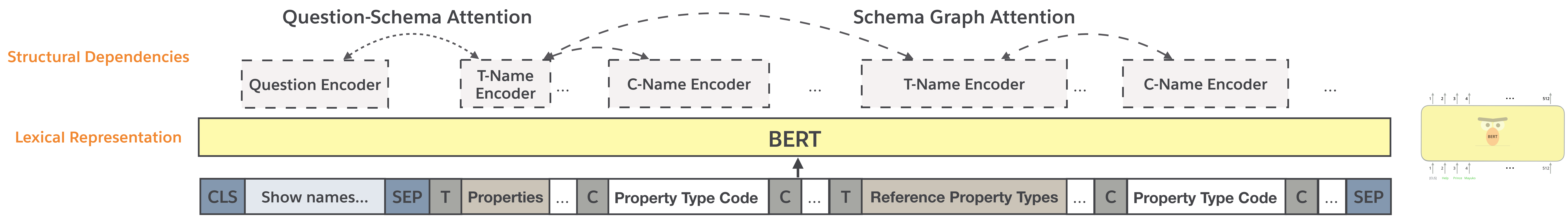
Show names of properties that are either houses or apartments



Previous work: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. Wang et. al. 2020.

Joint Textual-Tabular Data Encoding

Attention Layers



Show names of properties that are either houses or apartments

Properties

Property id	Property type code	Property name	Date on market	Date sold	...
	...	...	...	...	...
	...				
	...				
	...				

Apartment
Field
House
Shop
Other

Reference Property Types

Property type code	Property type description
...	...
...	...
...	...
...	...

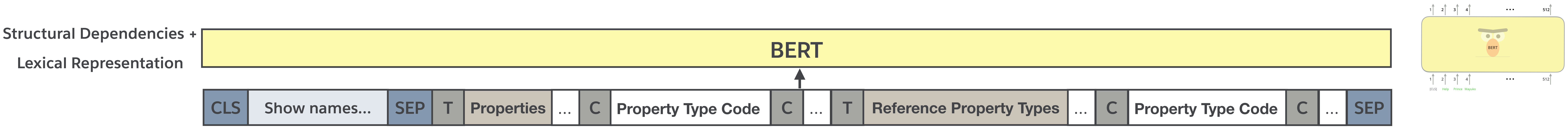
Apartment
Field
House
Shop
Other

Model complexity quickly increases.

+6 Relational Self-Attention Layers on top of BERT-large. Architecture redundancy

Previous work: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. Wang et. al. 2020.

Joint Textual-Tabular Data Encoding



Show names of properties that are either houses or apartments

Properties

Property id	Property type code	Property name	Date on market	Date sold	...
	...	...	...	...	...
	...	..			
	...	..			
	...	..			

Property type code values: Apartment, Field, House, Shop, Other

Reference Property Types

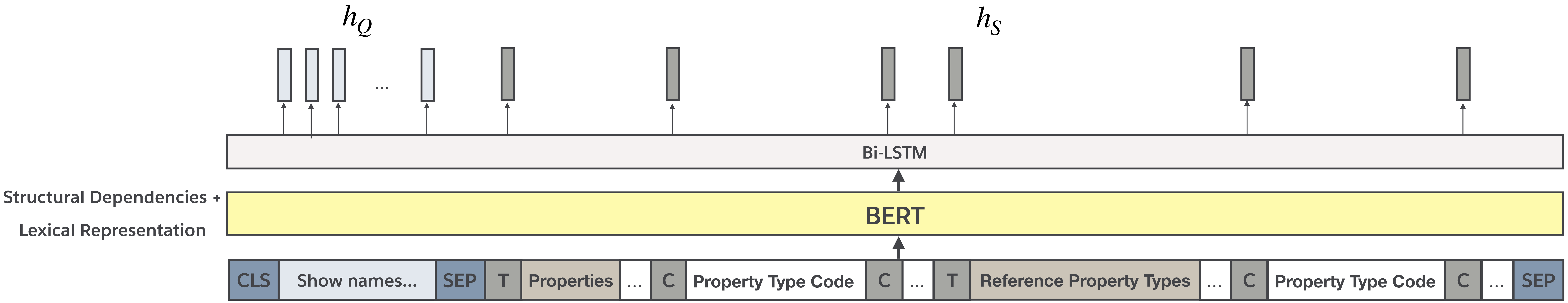
Property type code	Property type description
...	...
...	...
...	...
...	...

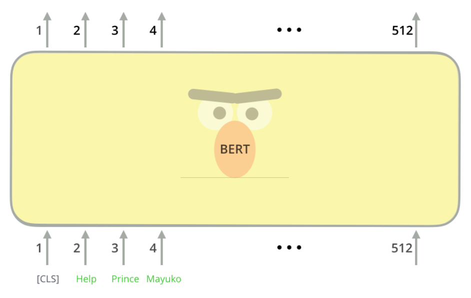
Property type code values: Apartment, Field, House, Shop, Other

Leveraging *just* the deep attention architecture in BERT to encode both lexical information as well as intra- and inter- modality dependencies.

Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin et. al. 2020.

Joint Textual-Tabular Data Encoding





Show names of properties that are either houses or apartments

Properties

Property id	Property type code	Property name	Date on market	Date sold	...
	...	...	...	...	...
	...	..			
	...	..			
	...	..			

Apartment
Field
House
Shop
Other

Reference Property Types

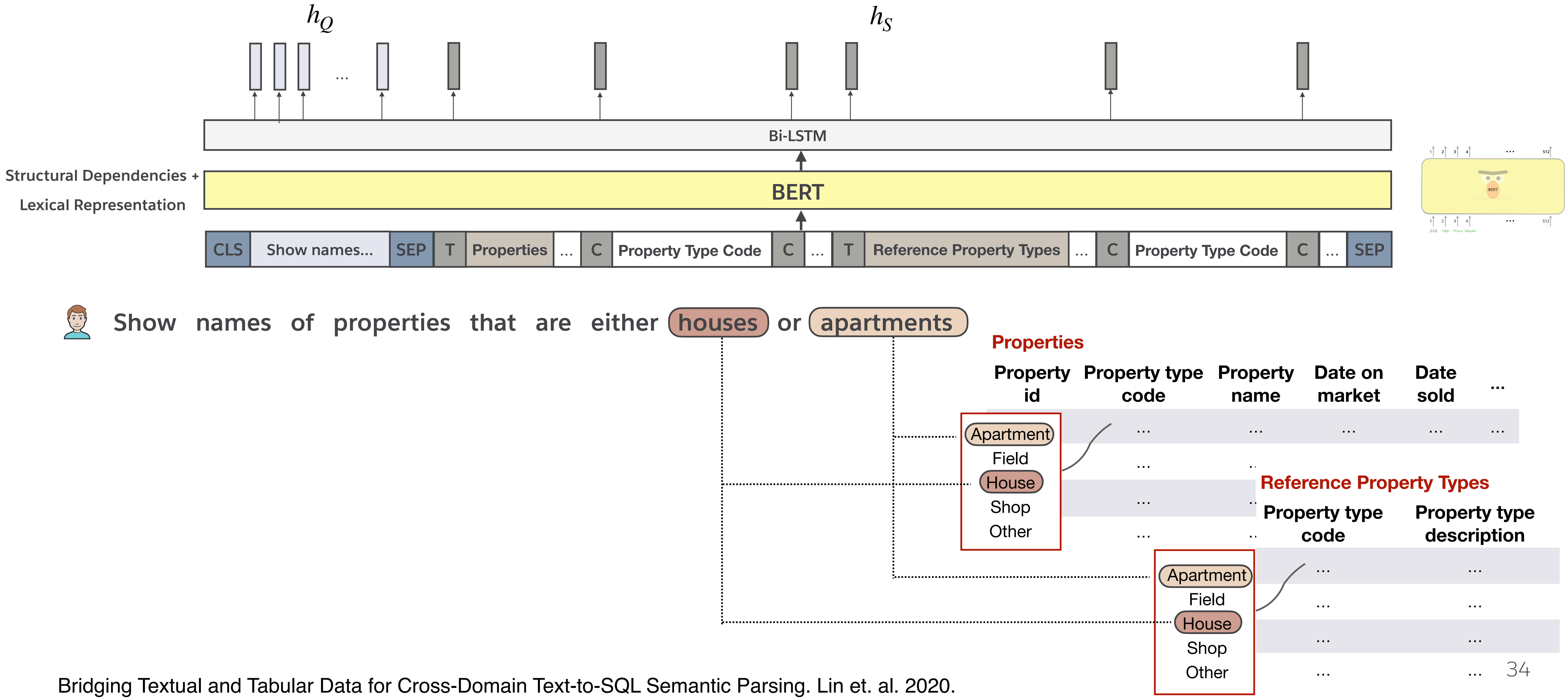
Property type code	Property type description
...	...
...	...
...	...
...	...

Apartment
Field
House
Shop
Other

Leveraging *just* the deep attention architecture in BERT to encode both lexical information as well as intra- and inter- modality dependencies.

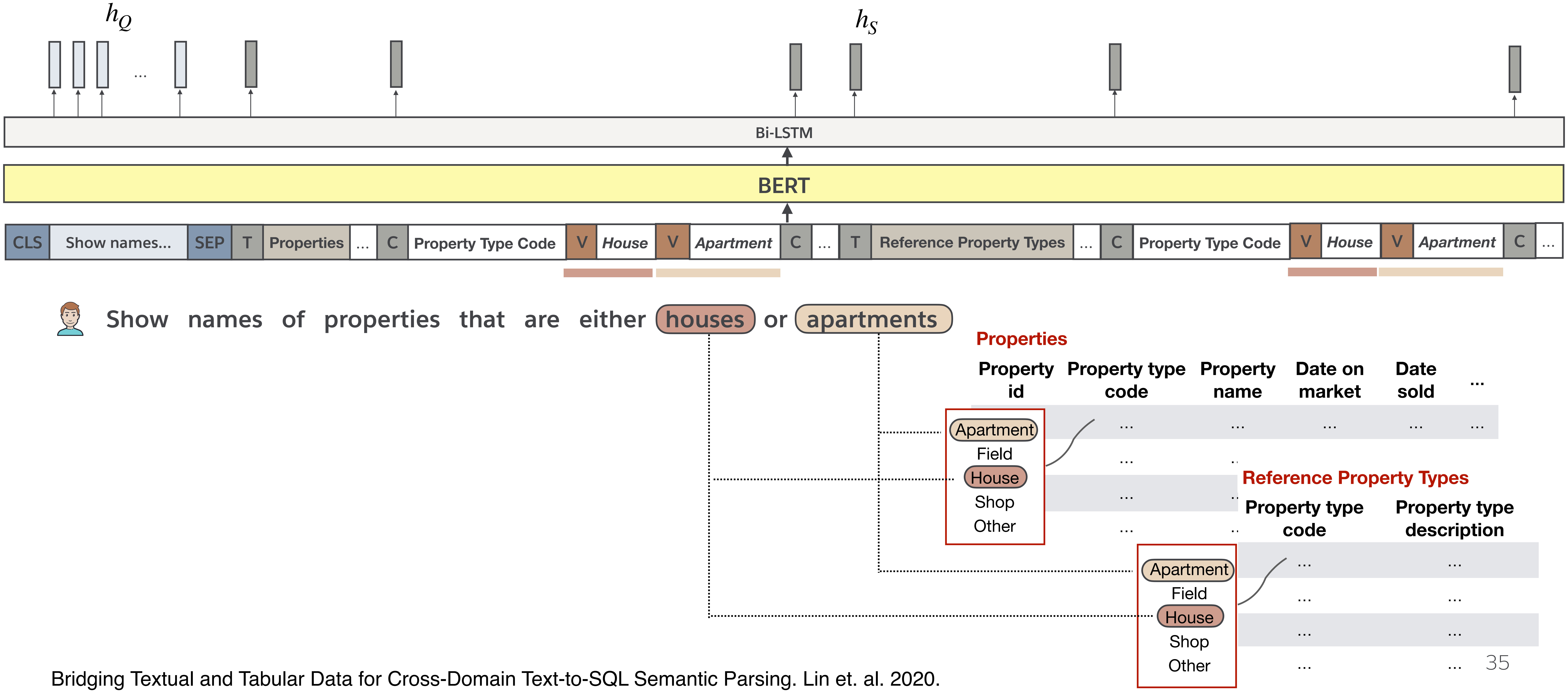
Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin et. al. 2020.

Joint Textual-Tabular Data Encoding



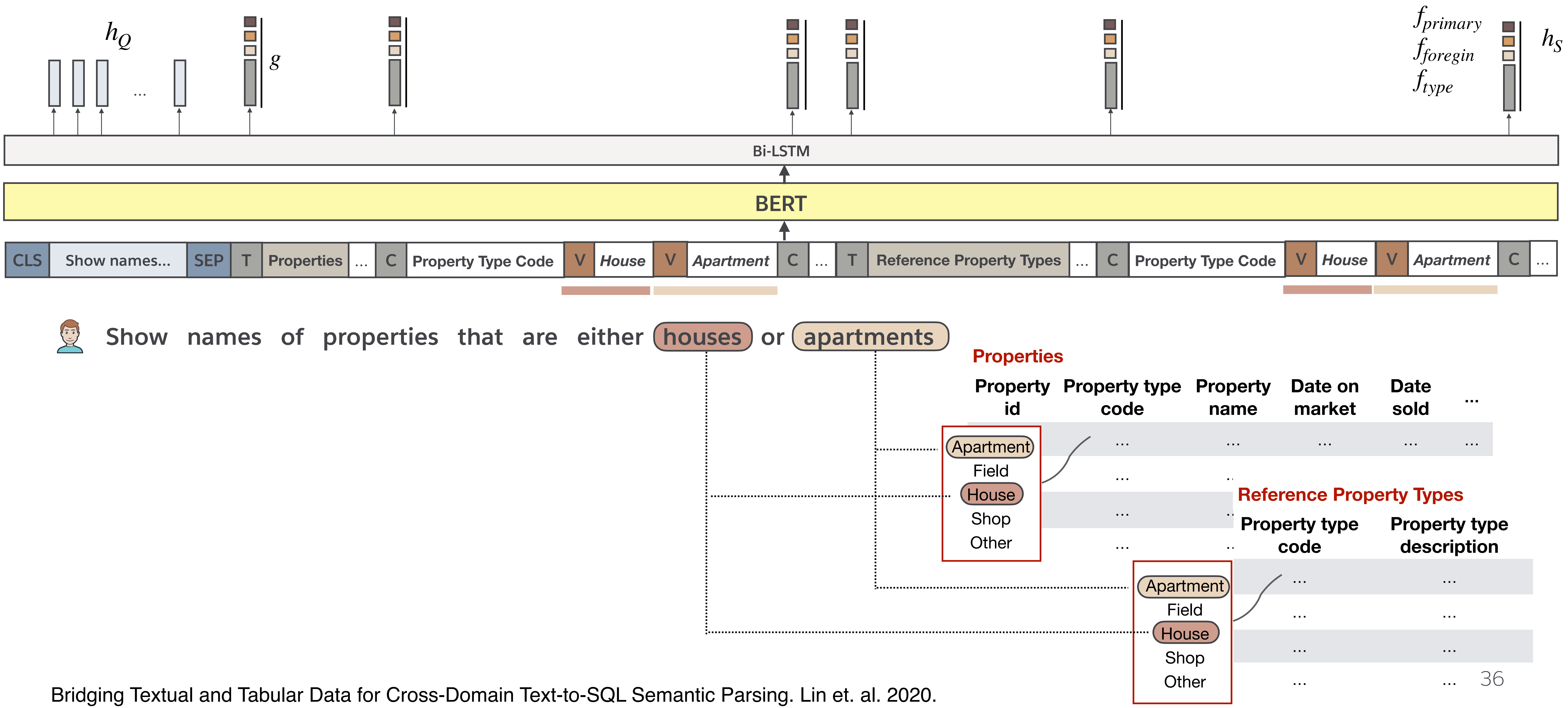
Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin et. al. 2020.

Bridging



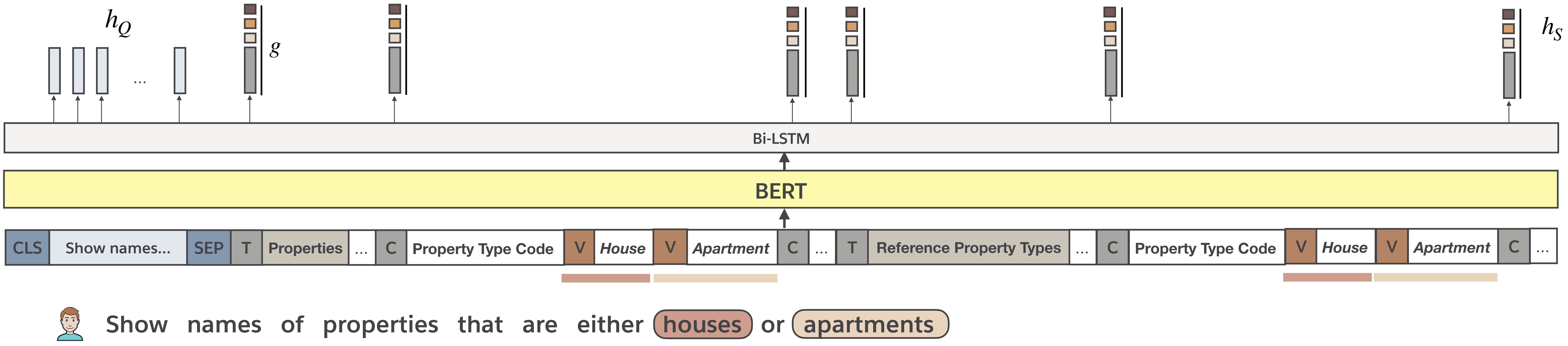
Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin et. al. 2020.

Meta-Data Encoding



Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin et. al. 2020.

Decoder



Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin et. al. 2020.

Decoder

LSTM-based pointer-generator (See et al. 2017)

p^t_{out}
- SQL Keywords and digits
- Question words
- Tables and columns

Cross-entropy loss

Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin et. al. 2020.

Schema-Consistency Guided Decoding

Pruning the search space of a sequential pointer-generator decoder

- **Observation:** The FROM clauses set the scope of a SQL query and the table fields appeared in the rest of the clauses can only belong to the tables in FROM

```
SELECT T2.name FROM Instructor AS T1 JOIN Department AS T2 ON T1.Department_ID = T2.ID
GROUP BY T1.Department_ID HAVING AVG(T1.Rating) > (SELECT AVG(Rating) FROM Instructor)
```

Schema-Consistency Guided Decoding

Pruning the search space of a sequential pointer-generator decoder

- **Observation:** The FROM clauses set the scope of a SQL query and the table fields appeared in the rest of the clauses can only belong to the tables in FROM

```
SELECT T2.name FROM Instructor AS T1 JOIN Department AS T2 ON T1.Department_ID = T2.ID
GROUP BY T1.Department_ID HAVING AVG(T1.Rating) > (SELECT AVG(Rating) FROM Instructor)
```

💡 Rewrite a SQL query in execution order, with FROM clause at the beginning of each sub-query

```
FROM Instructor AS T1 JOIN Department AS T2 ON T1.Department_ID = T2.ID SELECT T2.name
GROUP BY T1.Department_ID HAVING AVG(T1.Rating) > (FROM Instructor SELECT AVG(Rating))
```

Schema-Consistency Guided Decoding

Pruning the search space of a sequential pointer-generator decoder

- **Observation:** The FROM clauses set the scope of a SQL query and the table fields appeared in the rest of the clauses can only belong to the tables in FROM

```
SELECT T2.name FROM Instructor AS T1 JOIN Department AS T2 ON T1.Department_ID = T2.ID
GROUP BY T1.Department_ID HAVING AVG(T1.Rating) > (SELECT AVG(Rating) FROM Instructor)
```

💡 Rewrite a SQL query in execution order, with FROM clause at the beginning of each sub-query

```
FROM Instructor AS T1 JOIN Department AS T2 ON T1.Department_ID = T2.ID SELECT T2.name
GROUP BY T1.Department_ID HAVING AVG(T1.Rating) > (FROM Instructor SELECT AVG(Rating))
```

Lemma: Let Y_{exec} be a SQL query with clauses arranged in execution order, then any table field in Y_{exec} will appear after the corresponding table token.

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically

FROM

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically


```
FROM Instructor
```

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically


```
FROM Instructor JOIN
```

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically


```
FROM Instructor JOIN Department
```

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically


```
FROM Instructor JOIN Department ON
```

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically


```
FROM Instructor JOIN Department ON Instructor.Department_ID = Department.ID SELECT
Department.name GROUP BY Instructor.Department_ID HAVING AVG(Instructor.Rating) >
(FROM Instructor SELECT AVG(Instructor.Rating))
```

Schema-Consistency Guided Decoding

- Generate SQL queries in execution order and unmask DB fields dynamically

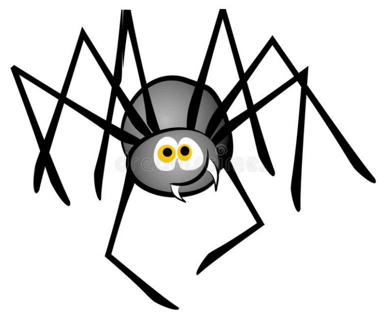

```
FROM Instructor JOIN Department ON Instructor.Department_ID = Department.ID SELECT
Department.name GROUP BY Instructor.Department_ID HAVING AVG(Instructor.Rating) >
(FROM Instructor SELECT AVG(Instructor.Rating))
```

✓ Implemented via vector space computation

✓ Applied during inference only

✓ Cannot guarantee schema consistency, used in combination with static SQL correctness check

✓ Can be applied to other types of decoders

Dataset - Spider 

(Yu et al. 2018)

Expert-annotated, **cross-domain**, **complex** text-to-SQL dataset

No overlap between Train/Dev/Test databases, enabling the development of text-to-SQL models which generalize to unseen DBs

Hidden

	Train	Dev	Test
# DBs	146	20	40
# Examples	8,659	1,034	2,147

Database

Primary key Foreign key

Instructor	ID	Name	Department_ID	Salary	...

Primary key

Department	ID	Name	Building	Budget	...

... ...

Question What are the name and budget of the departments with average instructor salary above the overall average?

SQL

```
SELECT T2.name, T2.budget
FROM Instructor AS T1 JOIN Department AS T2 ON
T1.Department_ID = T2.ID
GROUP BY T1.Department_ID
HAVING AVG(T1.salary) >
    (SELECT AVG(Salary) FROM Instructor)
```

Dataset - WikiSQL

(Zhong et al. 2017)

Generated over **Wikipedia tables** using the **semantic-parsing-overnight** approach (Wang et al. 2015)

SQL Template:

```
SELECT $AGG $COLUMN
WHERE $COLUMN $OP $VALUE
(AND $COLUMN $OP $VALUE)*
```

Train/Dev/Test tables overlap, but **49.6%** of dev tables are not in the train set and **45.1%** of test tables are not in the train set.

	Train	Dev	Test
# Tables	17,984	1,621	2,787
# Examples	56,355	8,421	15,878

WikiTable

Player	No.	Nationality	Position	Years in Toronto	School/Club

Question Who is the player that wears number 45?

SQL `SELECT Player WHERE No. = 42`

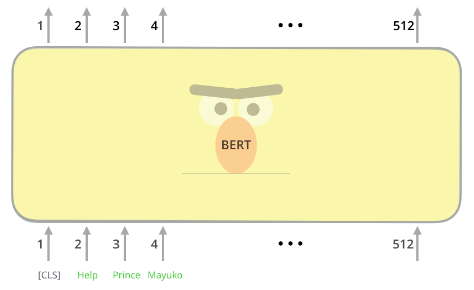
Experiment Setup

Pre-processing

- Compute fuzzy string match between the input question and the picklists of each DB field to obtain value mentions
- For each DB field, use the top-K matches in the DB schema representation ($K = 2$)

Decoding

- Beam search with length penalty
  - beam size = 16 for ablation study; beam size = 64 for leaderboard results



BERT-large-uncased, 24 layers (Devlin et al. 2019)

Evaluation

- Exact set match
  - Logical form match ignoring values and SQL component order invariance
- Execution accuracy
  - Check if the execution results of the predicted SQL query matches the executions results of the ground-truth SQL query

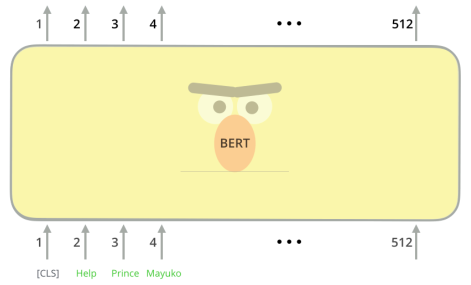
Experiment Setup

Pre-processing

- Compute fuzzy string match between the input question and the picklists of each DB field to obtain value mentions
- For each DB field, use the top-K matches in the DB schema representation ($K = 2$)

Decoding

- Beam search with length penalty
  - beam size = 16 for ablation study; beam size = 64 for leaderboard results



BERT-large-uncased, 24 layers (Devlin et al. 2019)

Evaluation

- Exact set match
  - Logical form match ignoring values and SQL component order invariance
- Execution accuracy
  - Check if the execution results of the predicted SQL query matches the executions results of the ground-truth SQL query

Better evaluation for text-to-SQL is still an open research problem

Ablation Study

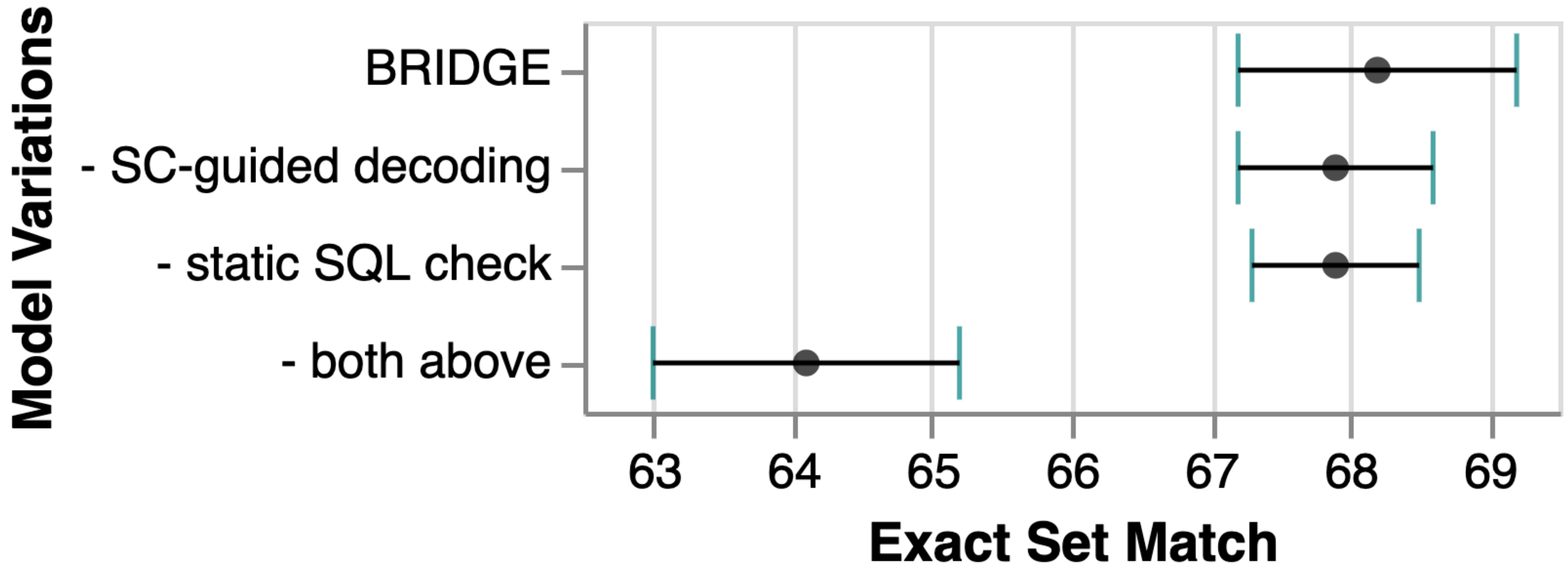


Figure 1. Ablation study of BRIDGE decoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study

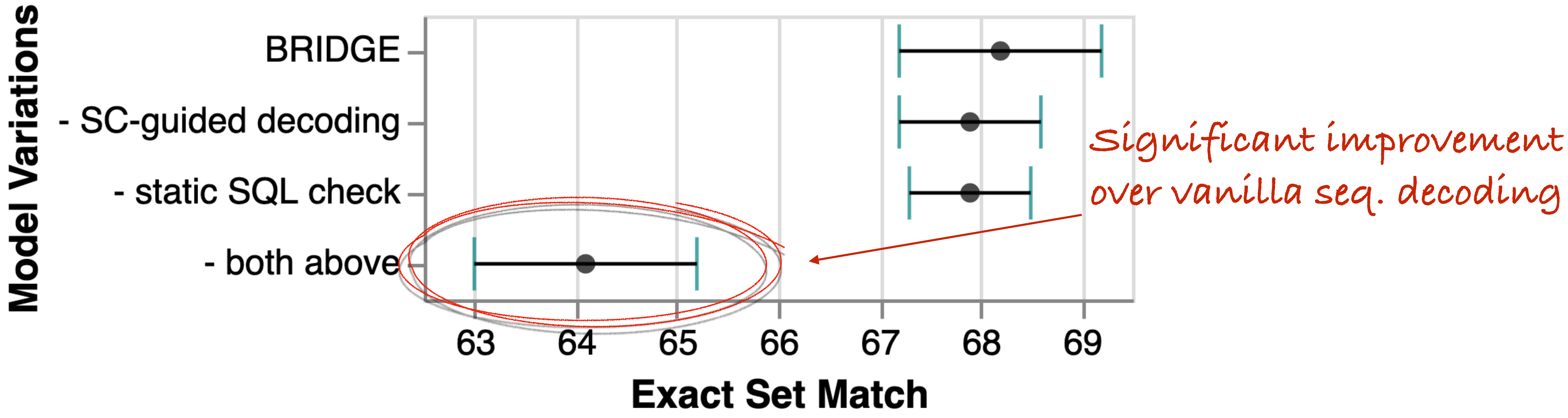


Figure 1. Ablation study of BRIDGE decoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study

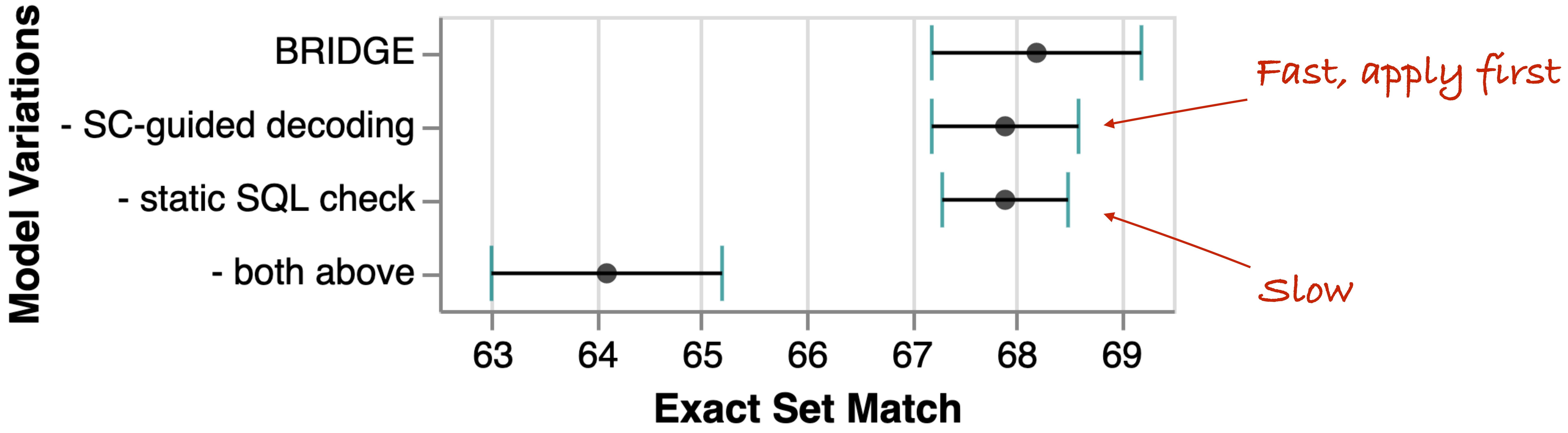


Figure 1. Ablation study of BRIDGE decoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study

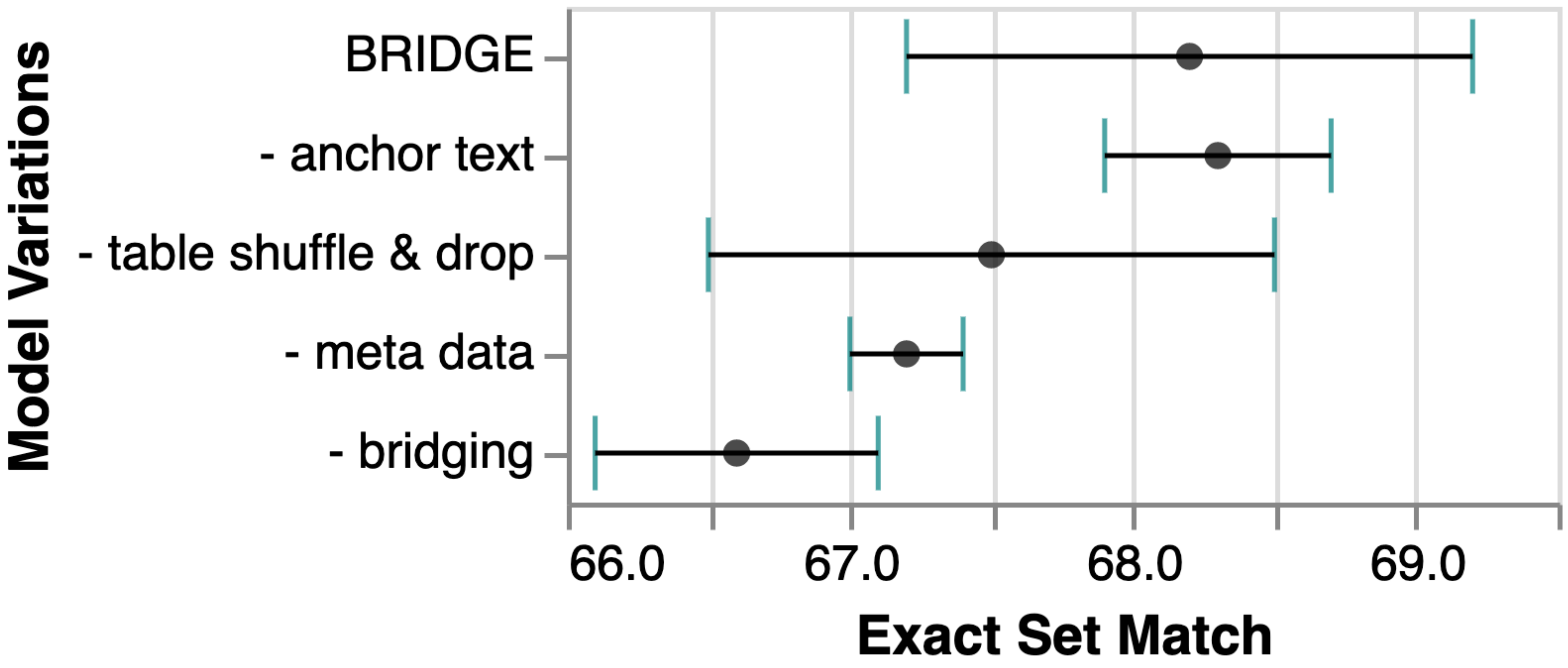


Figure 2. Ablation study of BRIDGE encoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study

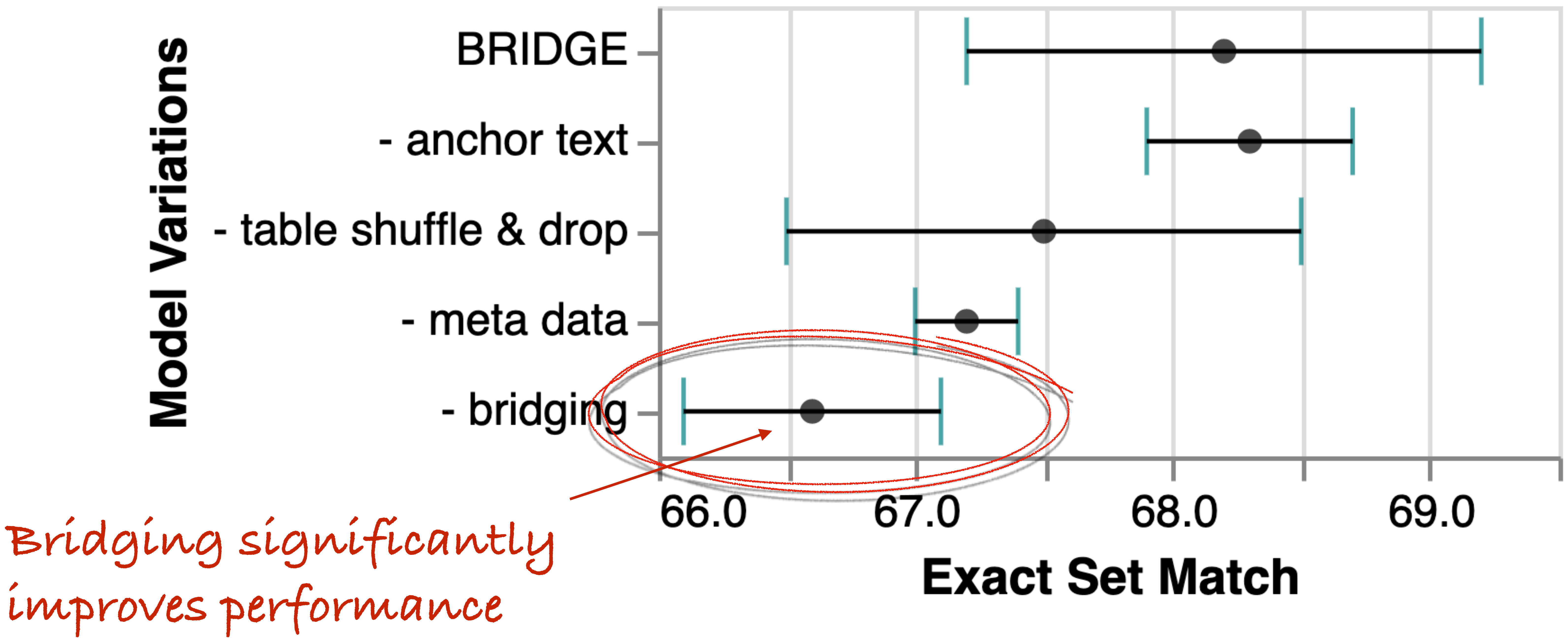
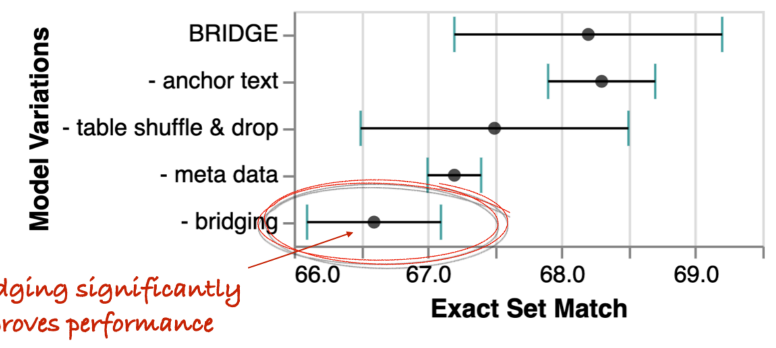


Figure 2. Ablation study of BRIDGE encoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Bridging Ablation Performance by Difficulty Level



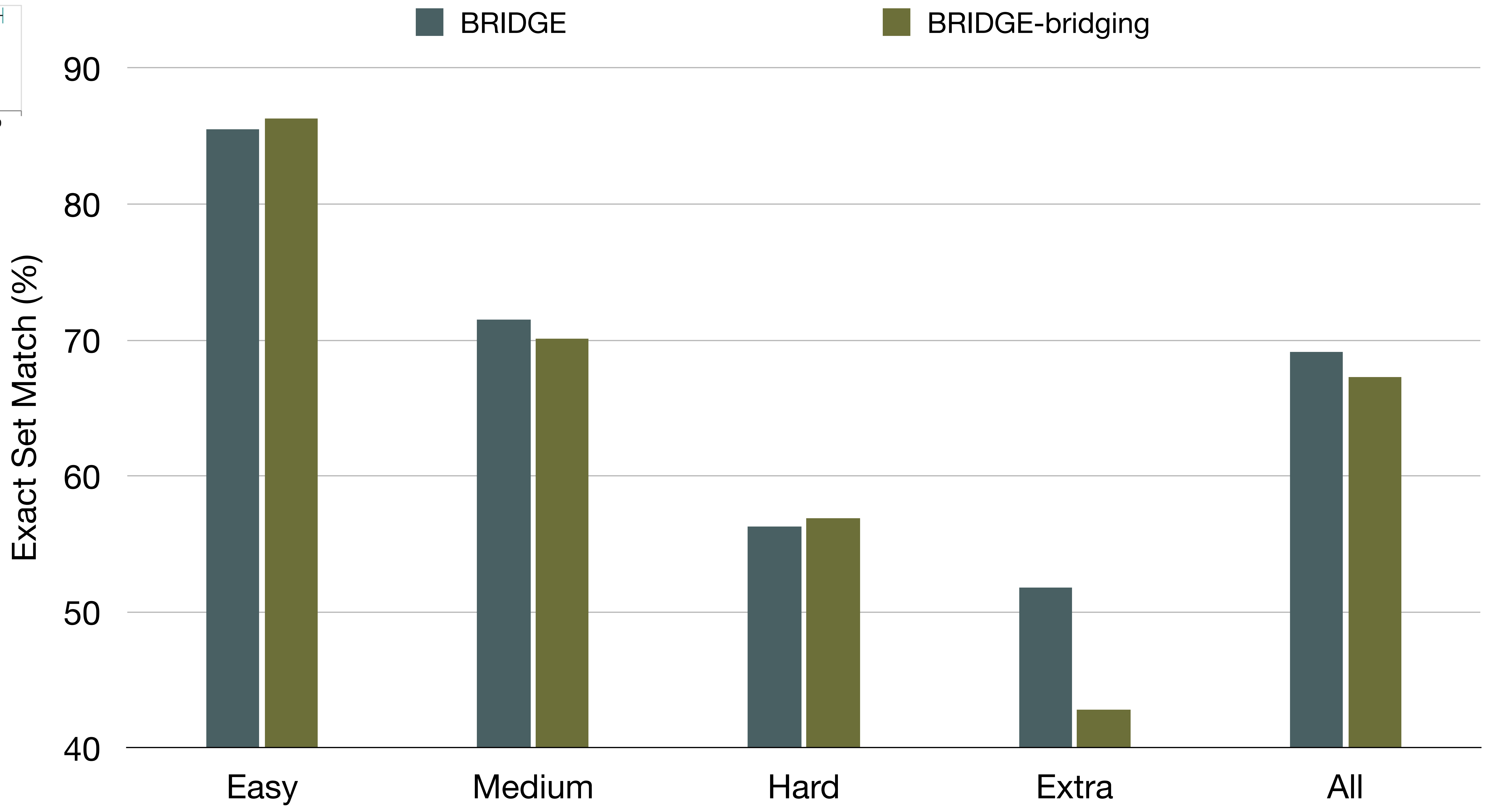


Figure 2.1. Performance of BRIDGE vs. BRIDGE - bridging on the Spider dev set.

Ablation Study

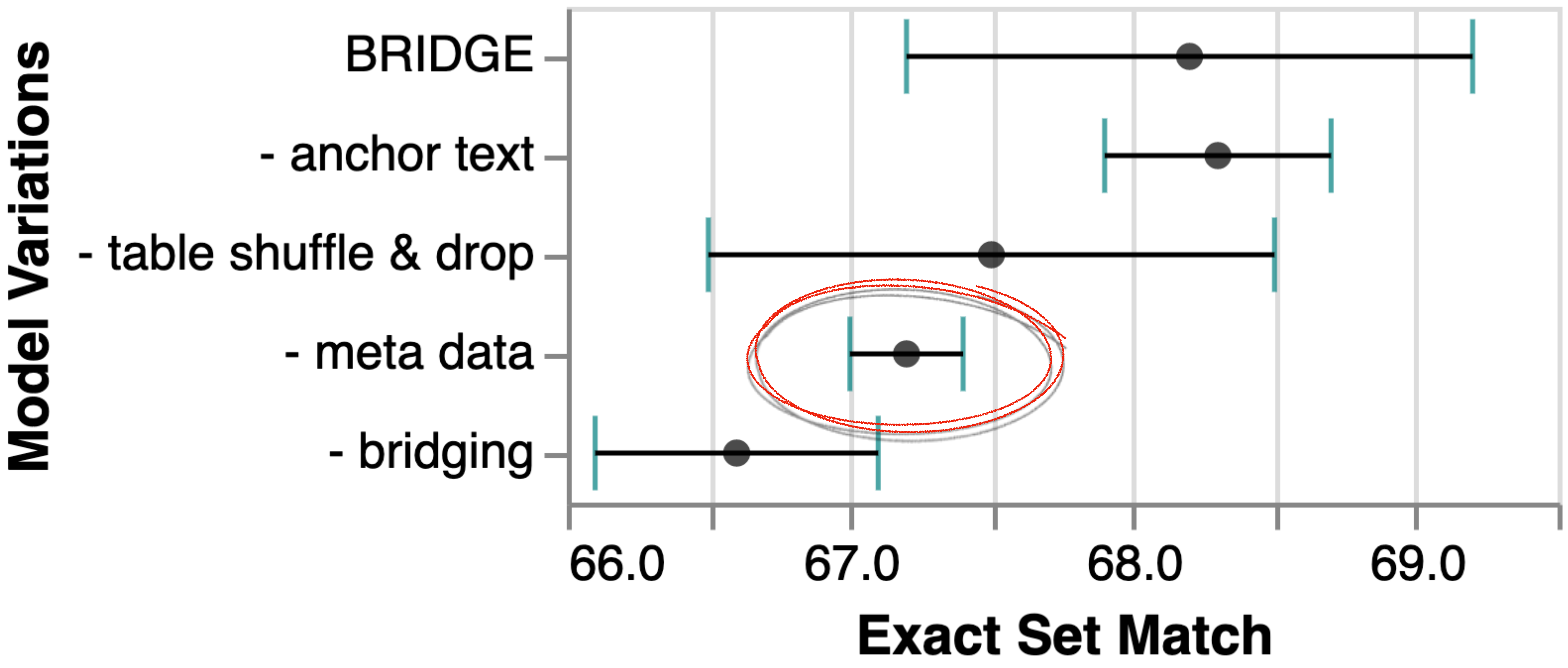


Figure 2. Ablation study of BRIDGE encoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study

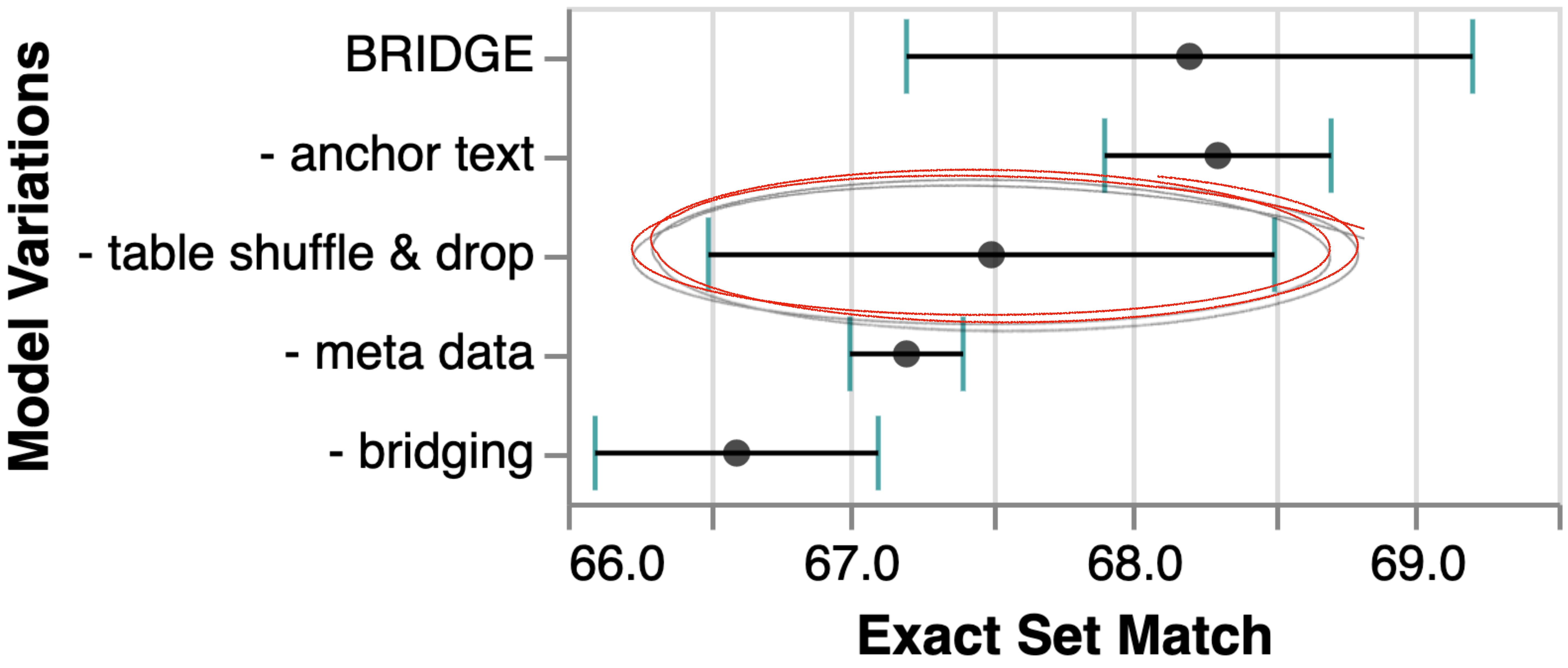
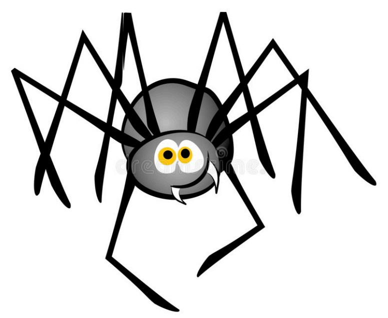


Figure 2. Ablation study of BRIDGE encoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study



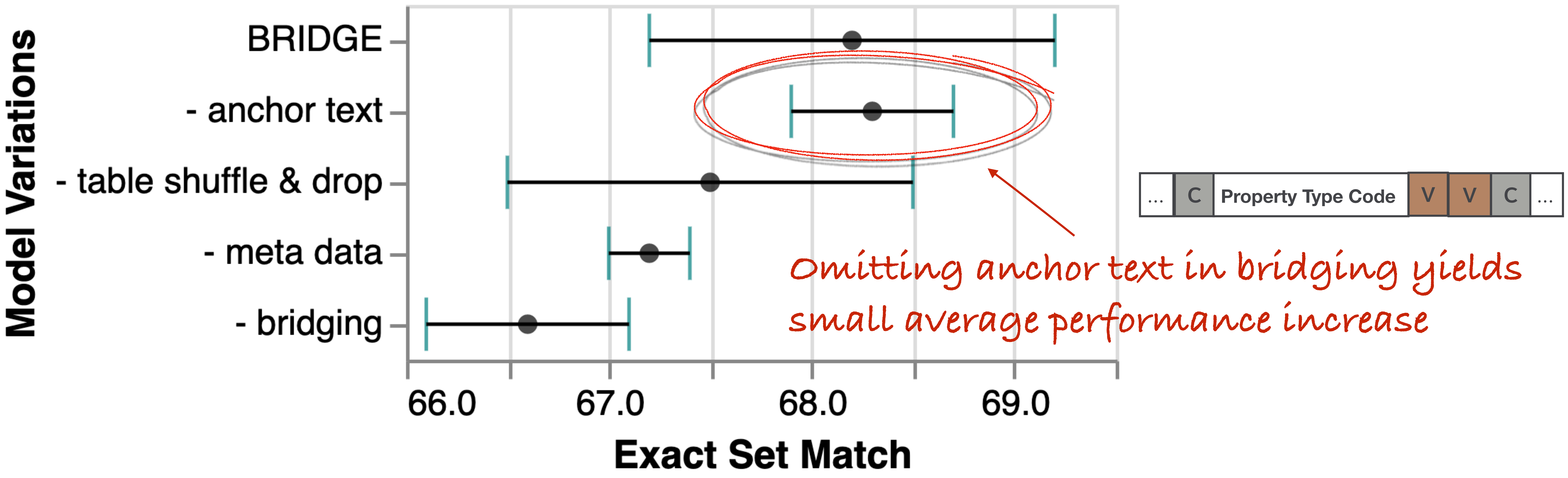
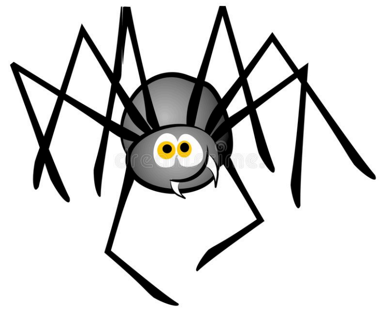


Figure 2. Ablation study of BRIDGE encoding strategies on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study



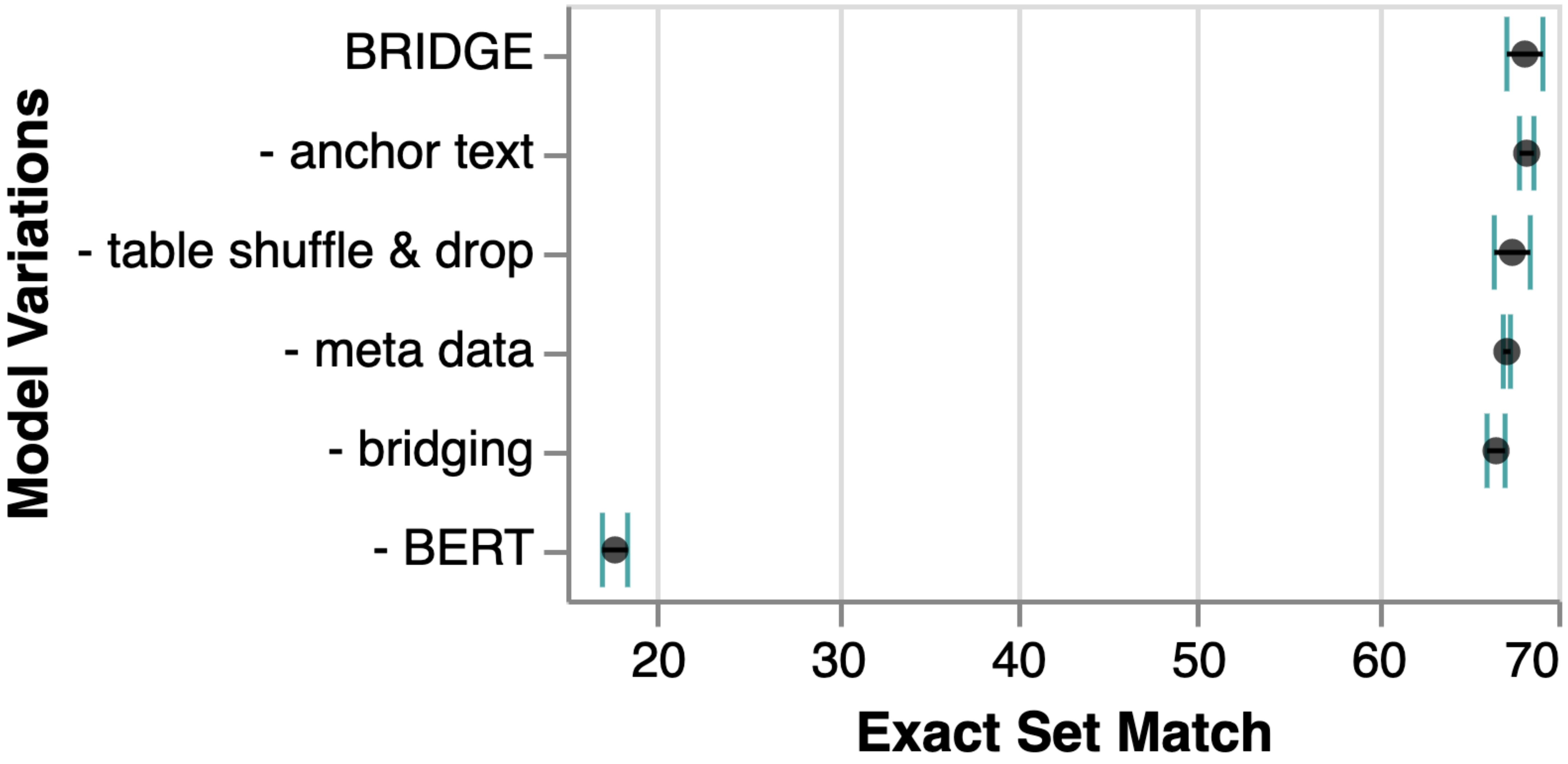
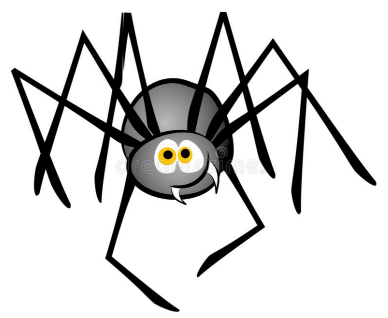
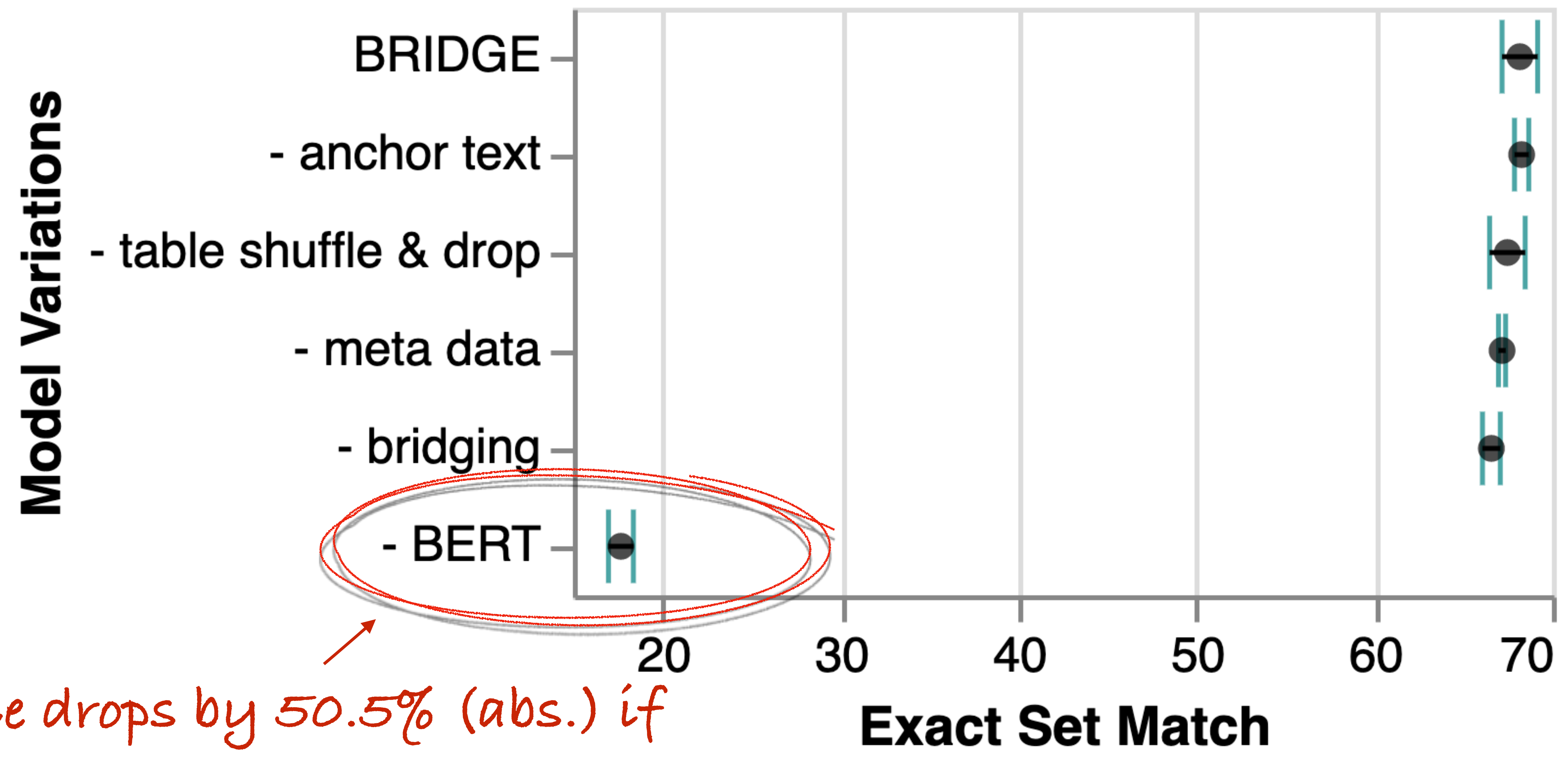


Figure 3. Ablation study of BRIDGE vs. BRIDGE - BERT on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study





Performance drops by 50.5% (abs.) if BERT is removed

Figure 3. Ablation study of BRIDGE vs. BRIDGE - BERT on the Spider Dev set. We train 3 models using different random seeds for each model variation and average the performances.

Ablation Study

Model	w/o EG		w/ EG	
	EM	EX	EM	EX
BRIDGE $_L$	**86.2**	**91.7**	**86.8**	**92.6**
-anchor text	84.2	90.0	85.2	91.3
-bridging	82.6	88.5	84.5	90.8

Figure 4. Ablation study of BRIDGE WikiSQL Dev set. We train only 1 model for each model variation since model variation on WikiSQL is very small. EG refers to “execution guided decoding”.

Ensemble Model

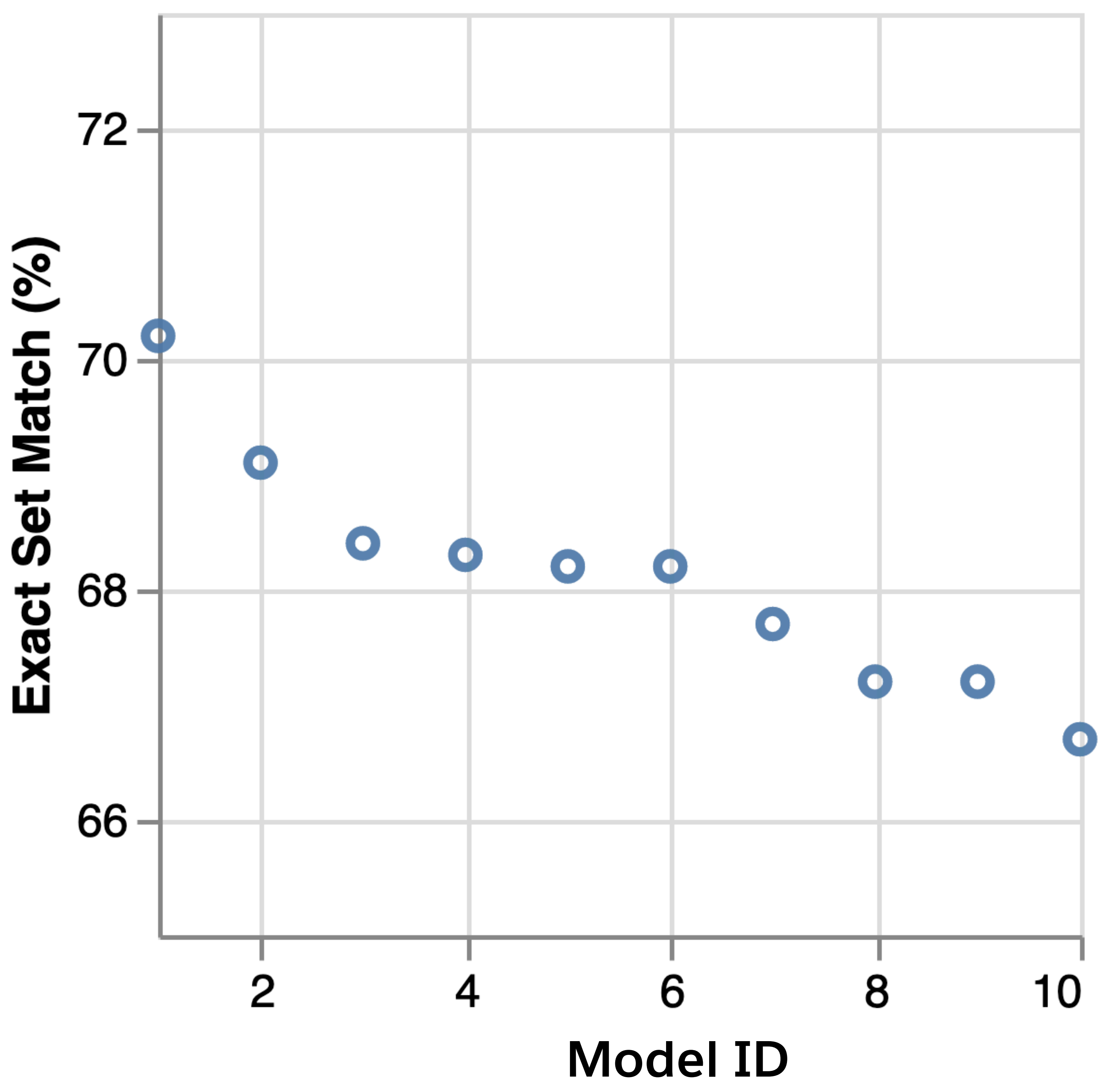
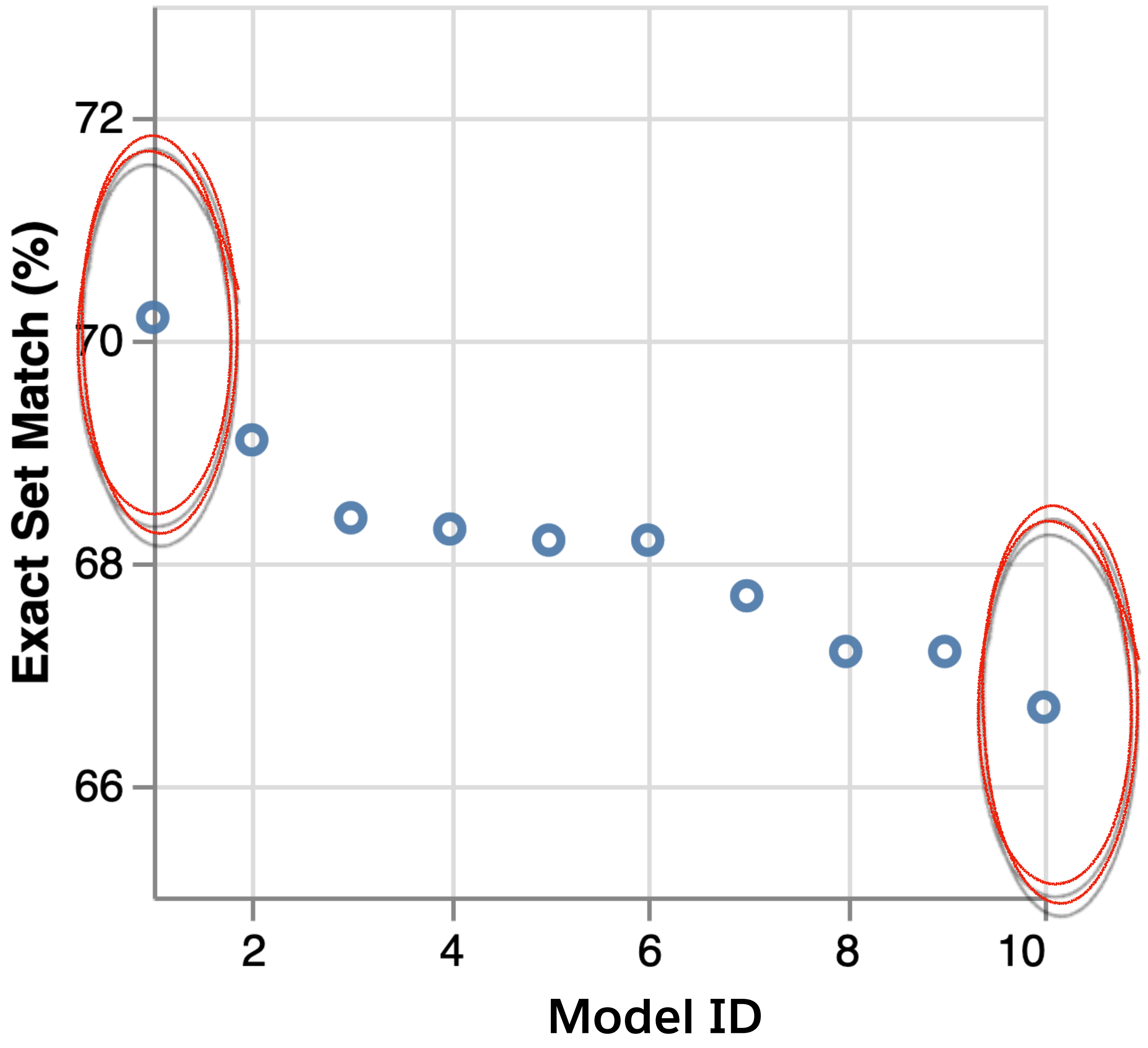


Figure 5. Performance comparison of BRIDGE model trained using 10 different random seeds on the Spider dev set.

Ensemble Model



	Best ✓	Best ✗
Worst ✓	61.2%	5.5%
Worst ✗	8.9%	24.4%

Figure 5. Performance comparison of BRIDGE model trained using 10 different random seeds on the Spider dev set.

Ensemble Model

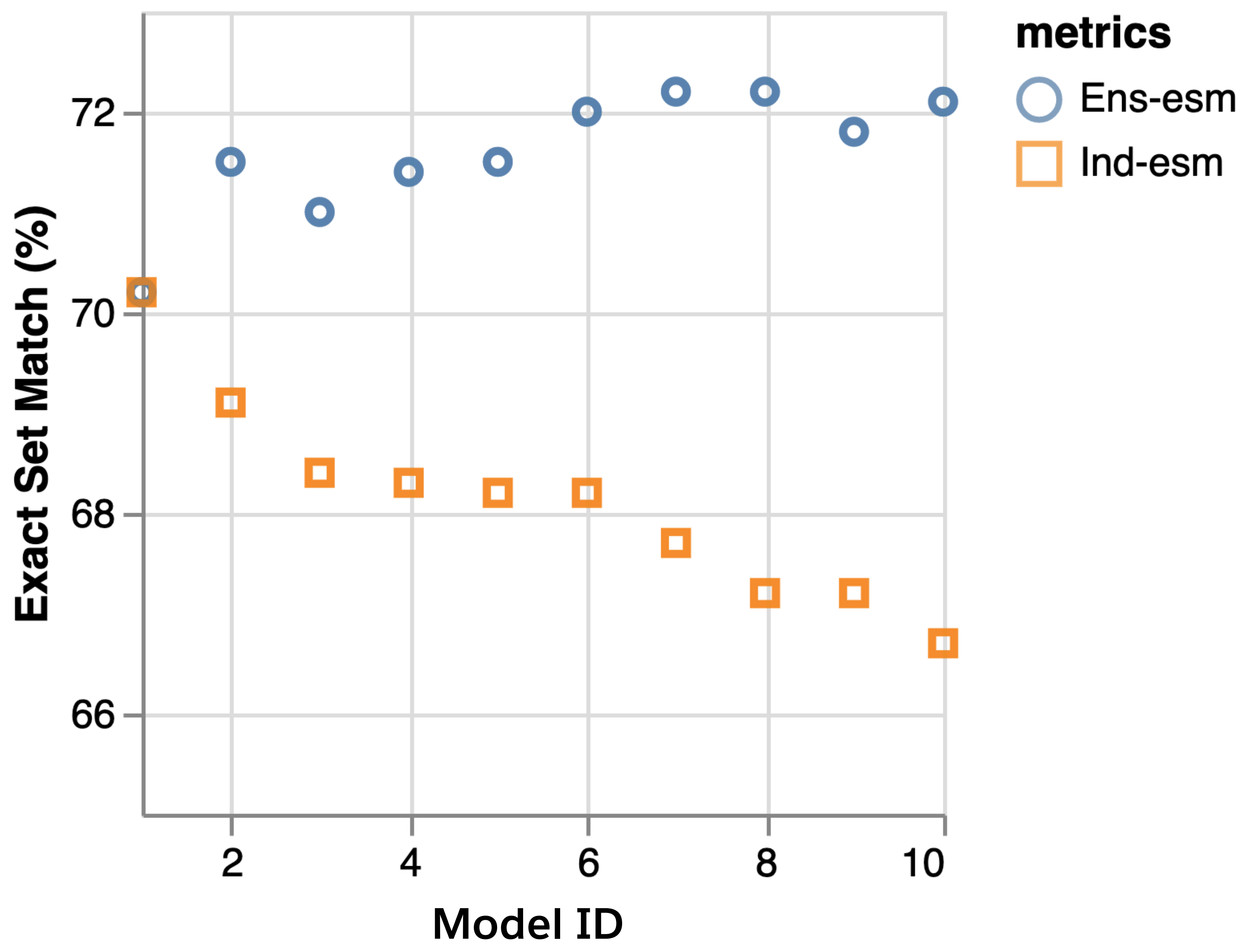
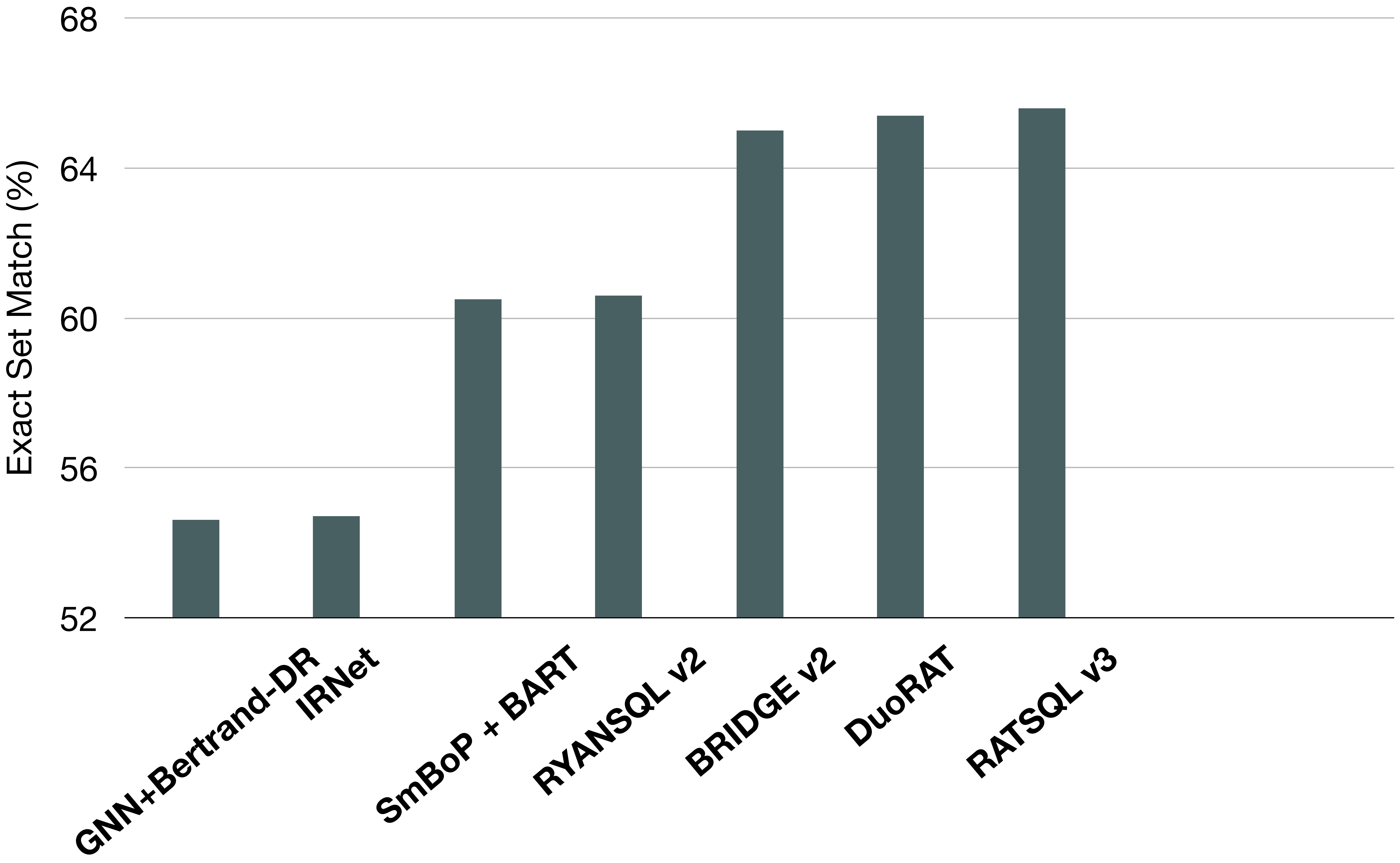


Figure 6. Performance of model ensemble (using step-wise output distribution average) on the Spider dev set.

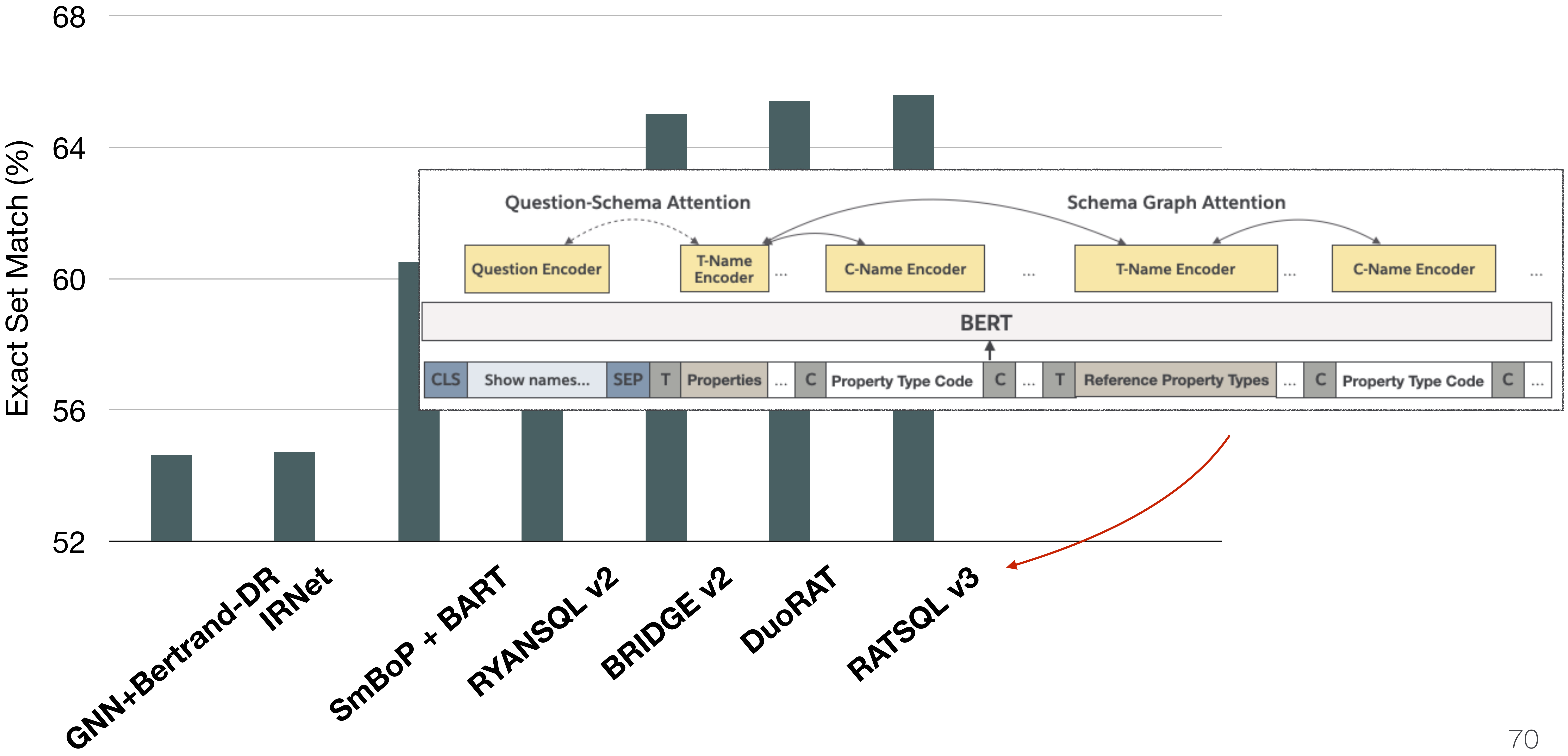
Performance on Spider Leaderboard

Figure 7. Comparison to other top-performing text-to-SQL models on the Spider leaderboard (Jan 31, 2021).



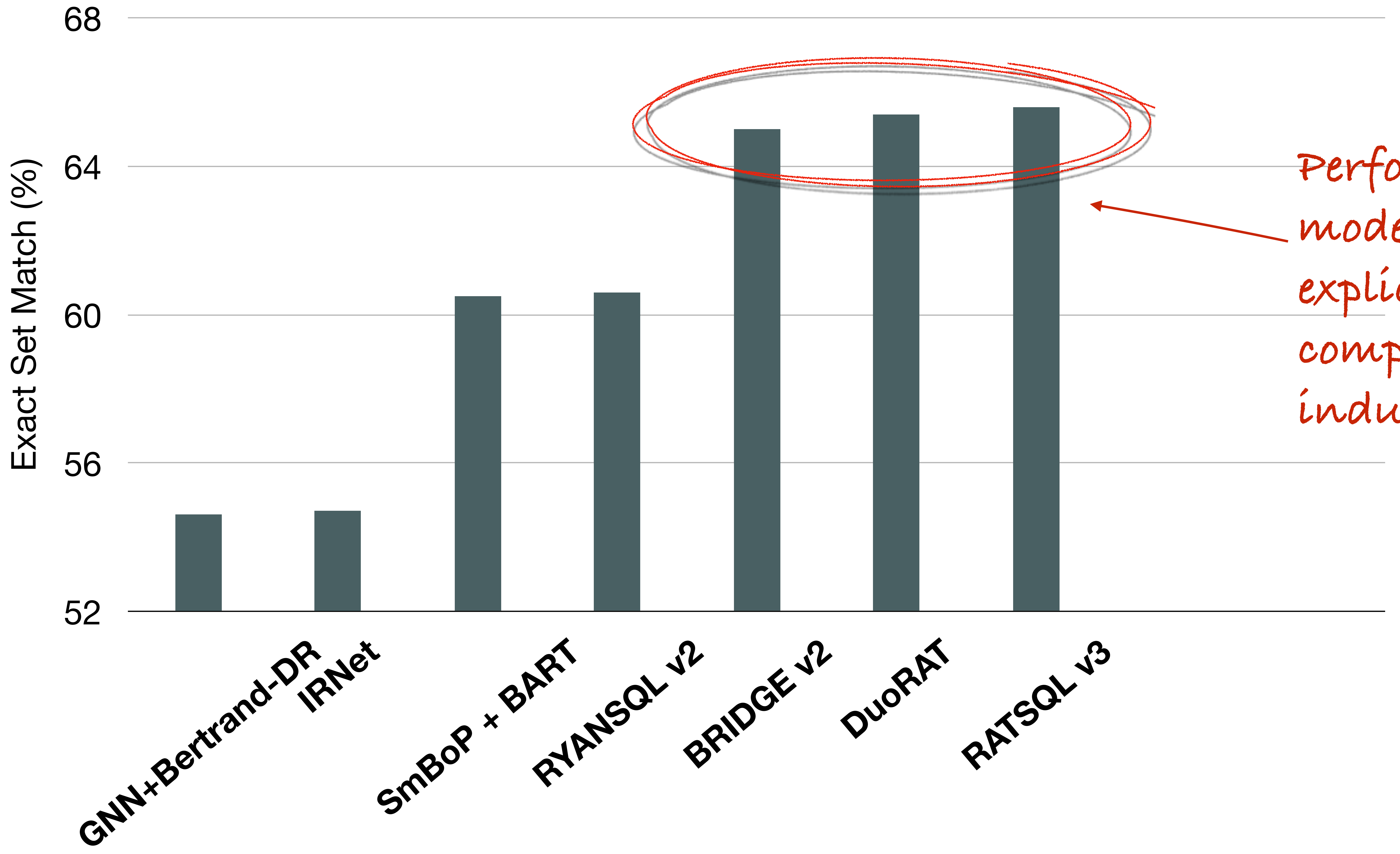
Performance on Spider Leaderboard

Figure 7. Comparison to other top-performing text-to-SQL models on the Spider leaderboard (Jan 31, 2021).



Performance on Spider Leaderboard

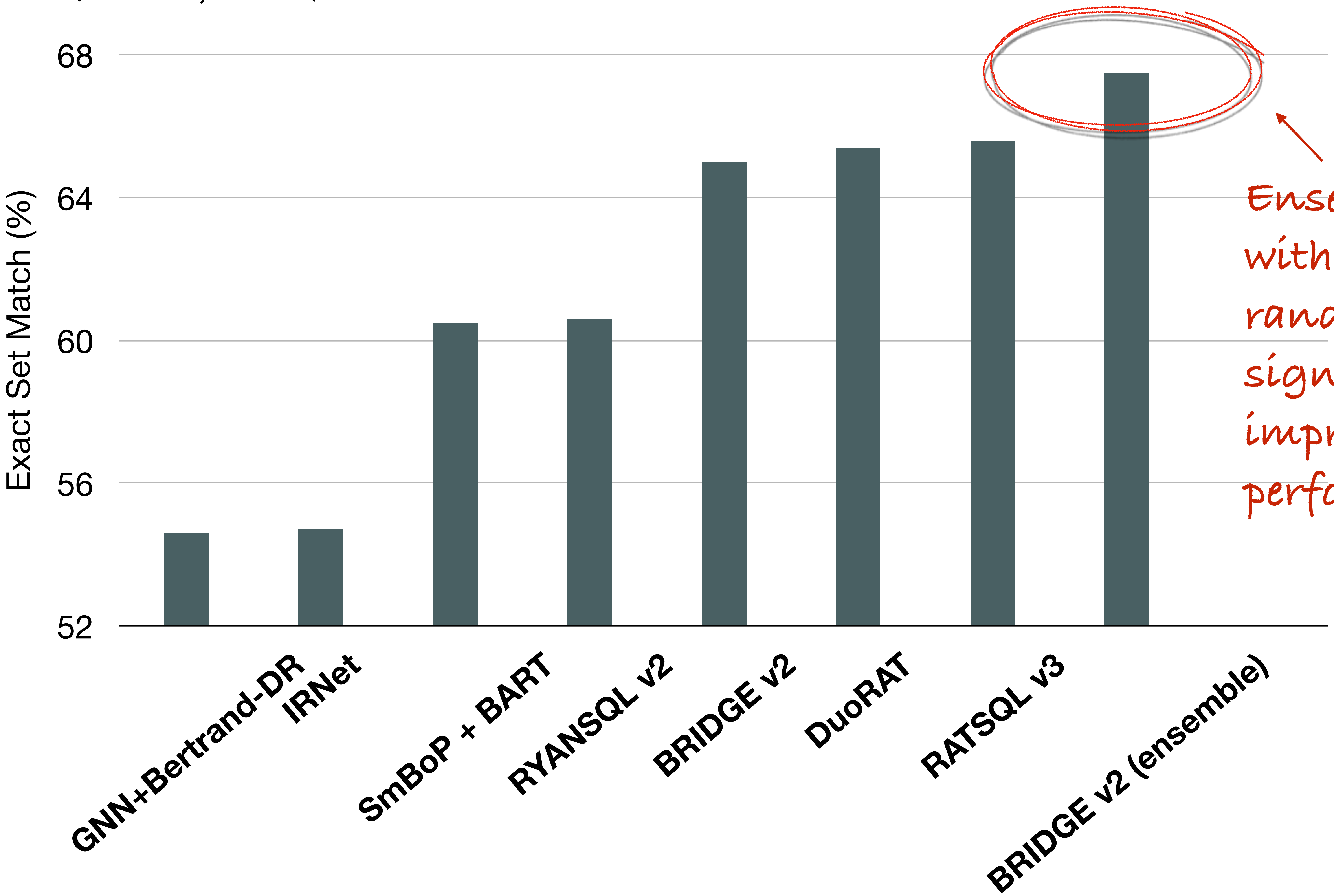
Figure 7. Comparison to other top-performing text-to-SQL models on the Spider leaderboard (Jan 31, 2021).



Performance matches models that explicitly model compositional inductive biases

Performance on Spider Leaderboard

Figure 7. Comparison to other top-performing text-to-SQL models on the Spider leaderboard (Jan 31, 2021).



Ensembling models with 3 different random seeds significantly improves test performance

Performance Comparison by Difficulty Level

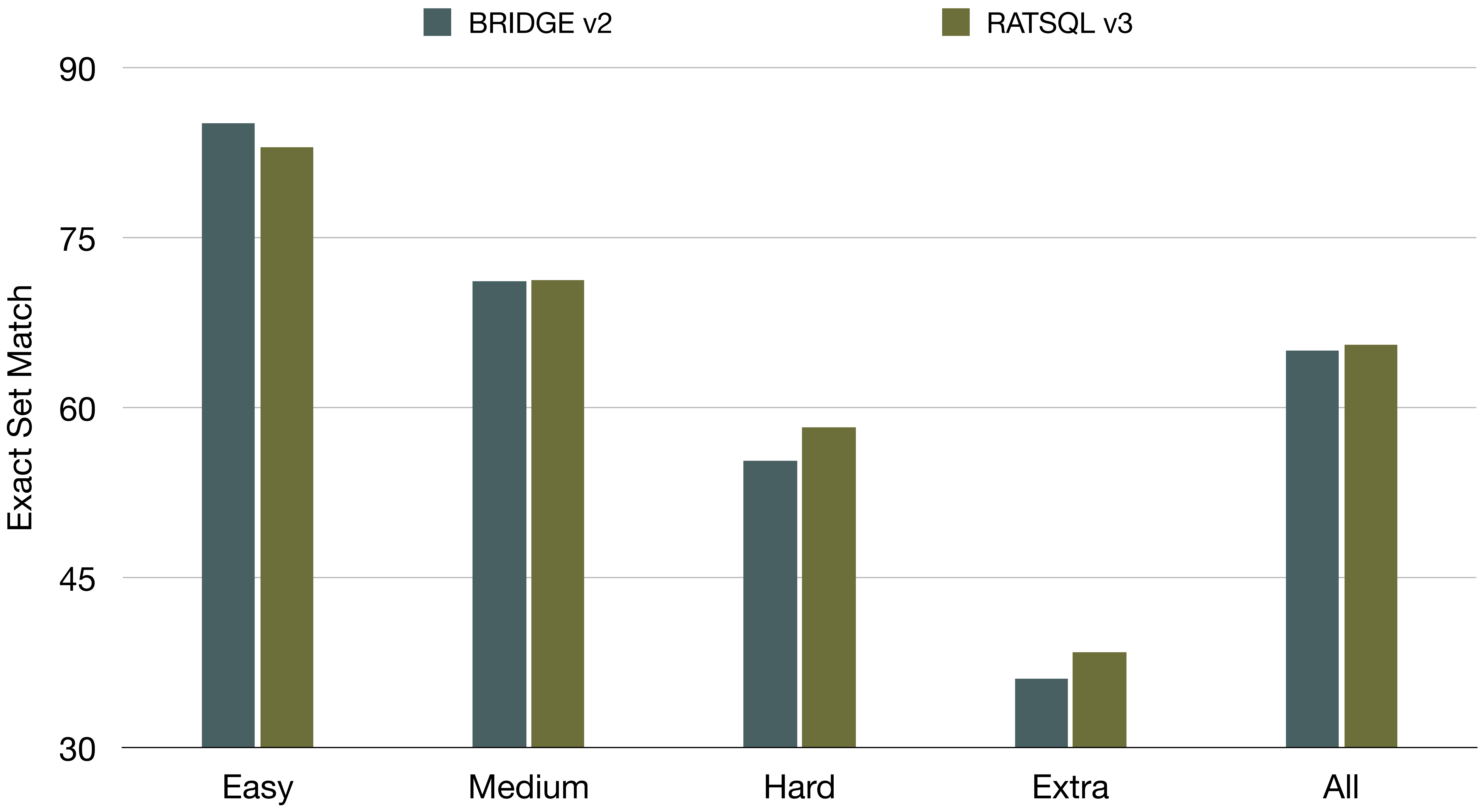


Figure 8. Performance of BRIDGE v2 compared to RATSQL v3 on the Spider Test set.

Performance Comparison by Difficulty Level

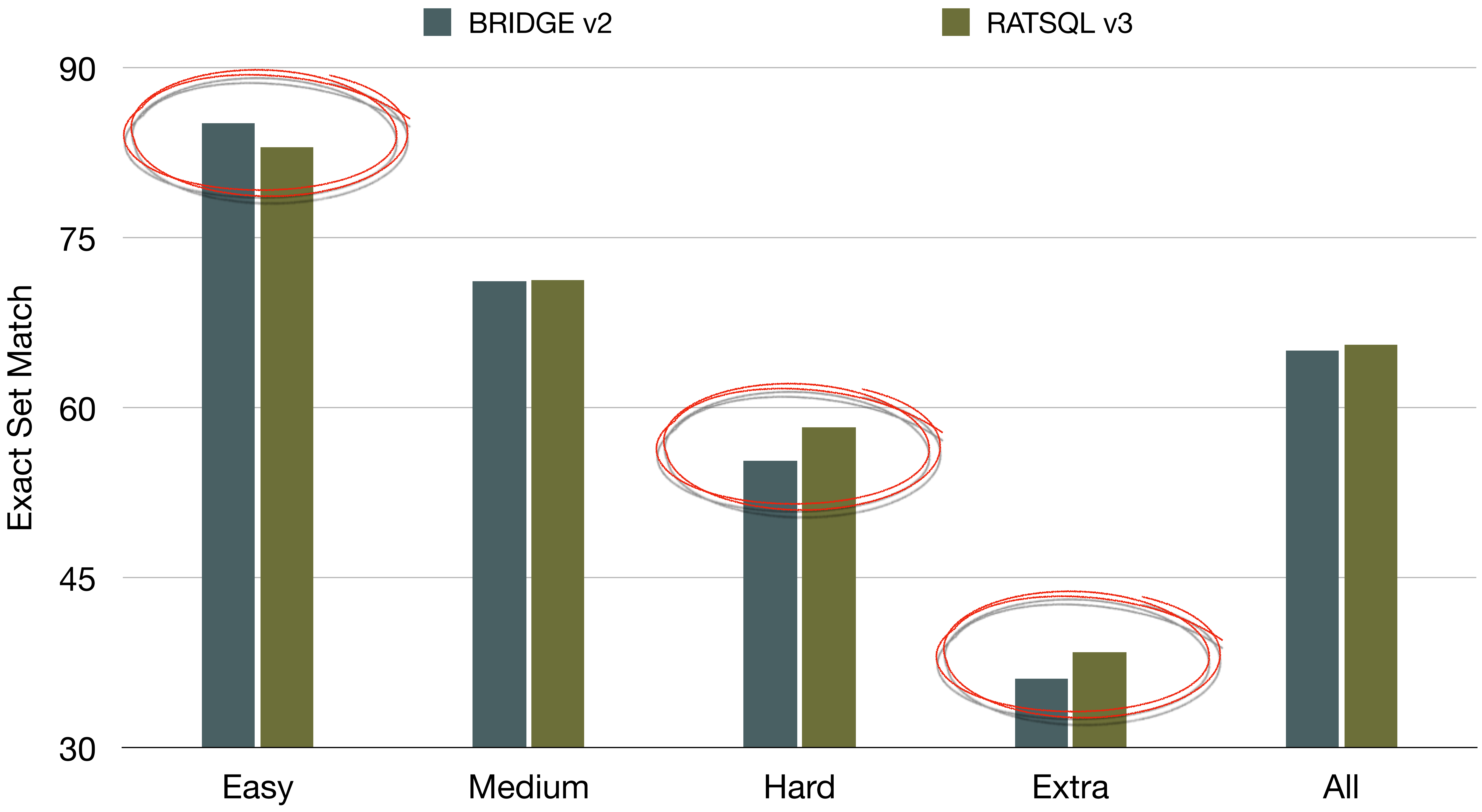
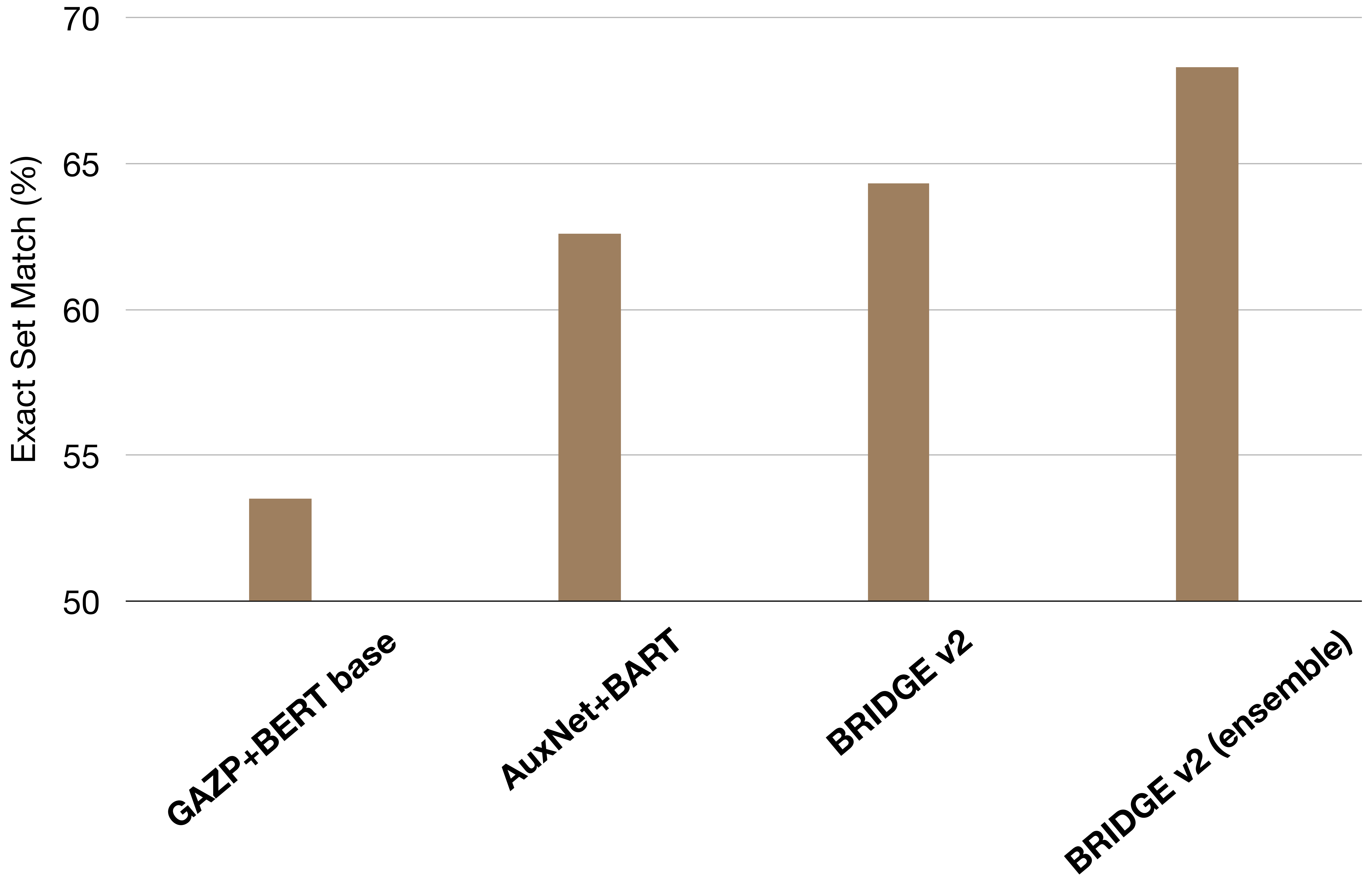


Figure 8. Performance of BRIDGE v2 compared to RATSQL v3 on the Spider Test set.

Performance on Spider Leaderboard - Execution Accuracy

Figure 7.1. Comparison to other top-performing text-to-SQL models on the Spider leaderboard based on execution accuracy (Jan 31, 2021).



Performance on WikiSQL Leaderboard

Figure 9. Comparison to other top text-to-SQL models on the WikiSQL leaderboard (Jan 31, 2020). ♠ denotes approaches that use table content during training. EG refers to "execution guided decoding".

Model	Dev		Test	
	EM	EX	EM	EX
SQLova (Hwang et al., 2019)	81.6	87.2	80.7	86.2
X-SQL (He et al., 2019b)	83.8	89.5	83.3	88.7
IE-SQL (Ma et al., 2020)	84.6	88.7	84.6	88.8
NL2SQL ♠ (Guo and Gao, 2019)	84.3	90.3	83.7	89.2
HydraNet (Lyu et al., 2020)	83.6	89.1	83.8	89.2
BRIDGE $_L$ ♠	**86.2**	**91.7**	**85.7**	**91.1**
SQLova+EG (Hwang et al., 2019)	84.2	90.2	83.6	89.6
NL2SQL+EG ♠ (Guo and Gao, 2019)	85.4	91.1	84.5	90.1
X-SQL+EG (He et al., 2019b)	86.2	92.3	86.0	91.8
BRIDGE $_L$+EG ♠	86.8	**92.6**	86.3	91.9
HydraNet+EG (Lyu et al., 2020)	86.6	92.4	86.5	92.2
IE-SQL+EG (Ma et al., 2020)	**87.9**	**92.6**	**87.8**	**92.5**

Performance on WikiSQL Leaderboard

Figure 9. Comparison to other top text-to-SQL models on the WikiSQL leaderboard (Jan 31, 2020). ♠ denotes approaches that use table content during training. EG refers to "execution guided decoding".

Model	Dev		Test	
	EM	EX	EM	EX
SQLova (Hwang et al., 2019)	81.6	87.2	80.7	86.2
X-SQL (He et al., 2019b)	83.8	89.5	83.3	88.7
IE-SQL (Ma et al., 2020)	84.6	88.7	84.6	88.8
NL2SQL ♠ (Guo and Gao, 2019)	84.3	90.3	83.7	89.2
HydraNet (Lyu et al., 2020)	83.6	89.1	83.8	89.2
BRIDGE $_L$ ♠	**86.2**	**91.7**	**85.7**	**91.1**
SQLova+EG (Hwang et al., 2019)	84.2	90.2	83.6	89.6
NL2SQL+EG ♠ (Guo and Gao, 2019)	85.4	91.1	84.5	90.1
X-SQL+EG (He et al., 2019b)	86.2	92.3	86.0	91.8
BRIDGE $_L$+EG ♠	86.8	**92.6**	86.3	91.9
HydraNet+EG (Lyu et al., 2020)	86.6	92.4	86.5	92.2
IE-SQL+EG (Ma et al., 2020)	**87.9**	**92.6**	**87.8**	**92.5**

Best model without execution guided decoding

Performance on WikiSQL Leaderboard

Figure 9. Comparison to other top text-to-SQL models on the WikiSQL leaderboard (Jan 31, 2020). ♠ denotes approaches that use table content during training. EG refers to "execution guided decoding".

Model	Dev EM	Dev EX	Test EM	Test EX
SQLova (Hwang et al., 2019)	81.6	87.2	80.7	86.2
X-SQL (He et al., 2019b)	83.8	89.5	83.3	88.7
IE-SQL (Ma et al., 2020)	84.6	88.7	84.6	88.8
NL2SQL ♠ (Guo and Gao, 2019)	84.3	90.3	83.7	89.2
HydraNet (Lyu et al., 2020)	83.6	89.1	83.8	89.2
BRIDGE $_L$ ♠	**86.2**	**91.7**	**85.7**	**91.1**
SQLova+EG (Hwang et al., 2019)	84.2	90.2	83.6	89.6
NL2SQL+EG ♠ (Guo and Gao, 2019)	85.4	91.1	84.5	90.1
X-SQL+EG (He et al., 2019b)	86.2	92.3	86.0	91.8
BRIDGE $_L$+EG ♠	86.8	**92.6**	86.3	91.9
HydraNet+EG (Lyu et al., 2020)	86.6	92.4	86.5	92.2
IE-SQL+EG (Ma et al., 2020)	**87.9**	**92.6**	**87.8**	**92.5**

Model	w/o EG EM	w/o EG EX	w/ EG EM	w/ EG EX
BRIDGE $_L$	**86.2**	**91.7**	**86.8**	**92.6**
-anchor text	84.2	90.0	85.2	91.3
-bridging	82.6	88.5	84.5	90.8

Best model without execution guided decoding

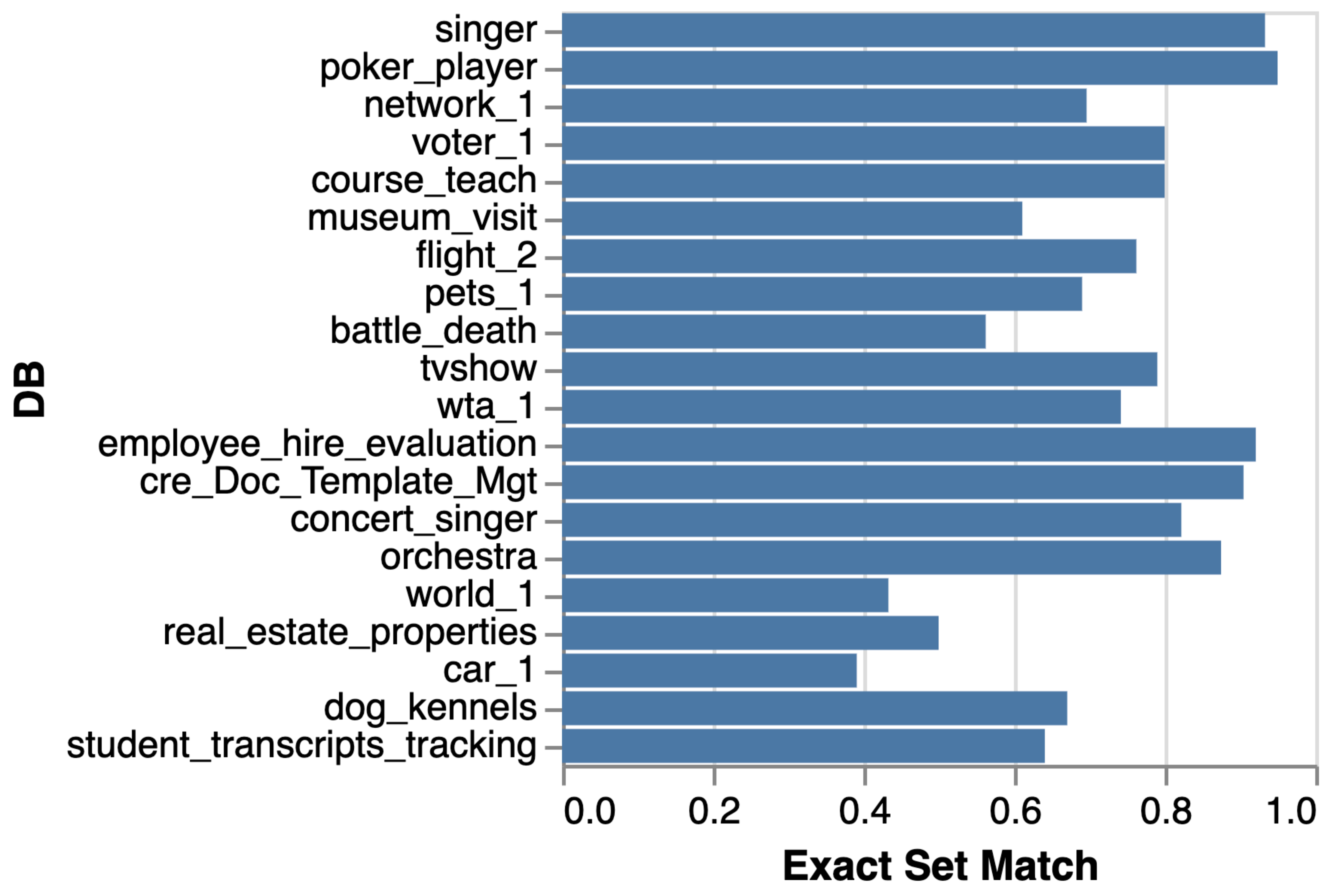
Performance on WikiSQL Leaderboard

Figure 9. Comparison to other top text-to-SQL models on the WikiSQL leaderboard (Jan 31, 2020). ♠ denotes approaches that use table content during training. EG refers to "execution guided decoding".

Model	Dev EM	Dev EX	Test EM	Test EX
SQLova (Hwang et al., 2019)	81.6	87.2	80.7	86.2
X-SQL (He et al., 2019b)	83.8	89.5	83.3	88.7
IE-SQL (Ma et al., 2020)	84.6	88.7	84.6	88.8
NL2SQL ♠ (Guo and Gao, 2019)	84.3	90.3	83.7	89.2
HydraNet (Lyu et al., 2020)	83.6	89.1	83.8	89.2
BRIDGE $_L$ ♠	**86.2**	**91.7**	**85.7**	**91.1**
SQLova+EG (Hwang et al., 2019)	84.2	90.2	83.6	89.6
NL2SQL+EG ♠ (Guo and Gao, 2019)	85.4	91.1	84.5	90.1
X-SQL+EG (He et al., 2019b)	86.2	92.3	86.0	91.8
BRIDGE $_L$+EG ♠	86.8	**92.6**	86.3	91.9
HydraNet+EG (Lyu et al., 2020)	86.6	92.4	86.5	92.2
IE-SQL+EG (Ma et al., 2020)	**87.9**	**92.6**	**87.8**	**92.5**

Top-3 model using execution guided decoding

Cross-Database Performance



Performance varies significantly w.r.t. different DBs

Figure 10. Performance of BRIDGE on each database on the Spider dev set.

Error Analysis

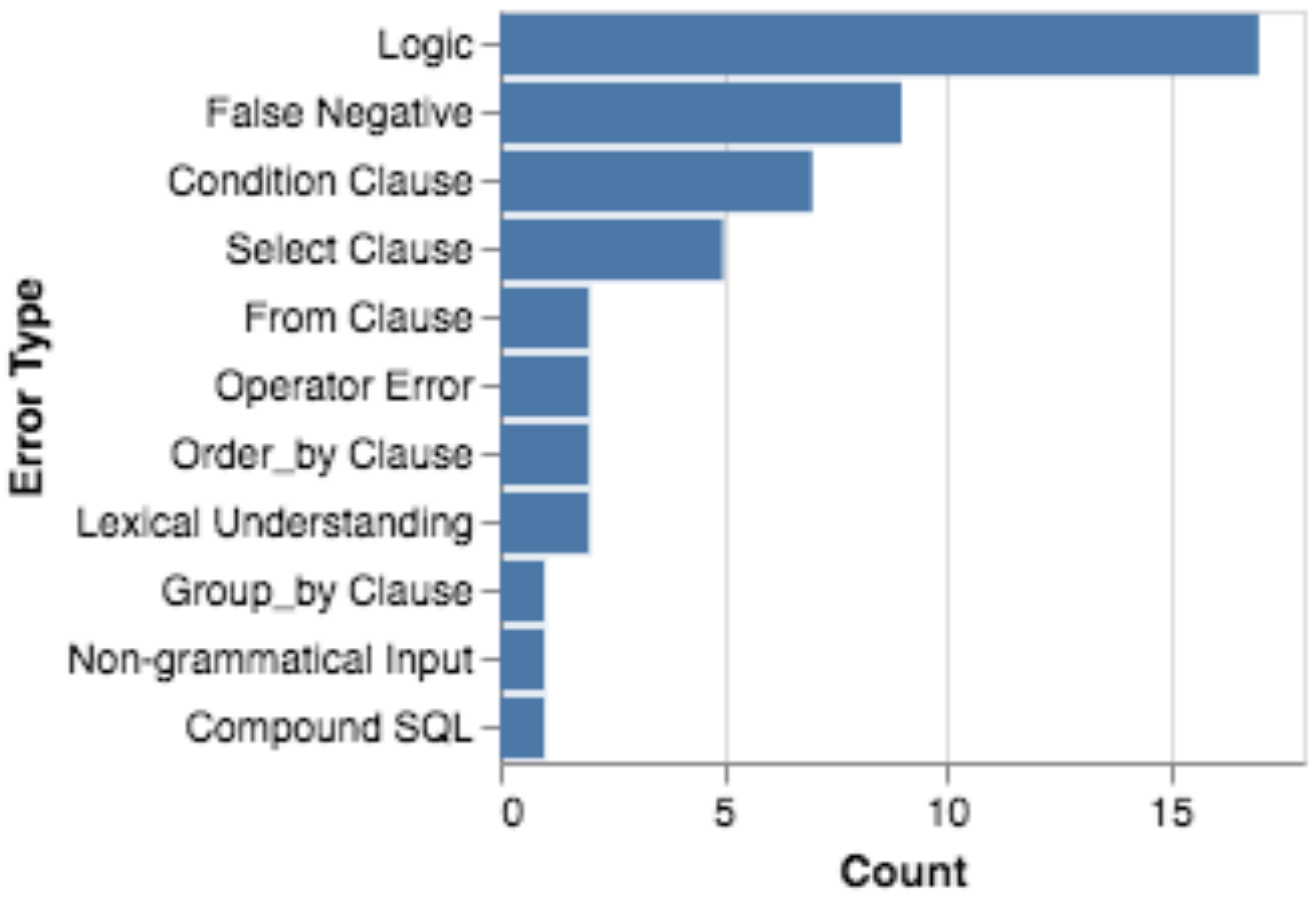


Figure 11. Manual error categorization for 50 wrong predictions on the Spider dev set.

Error Analysis

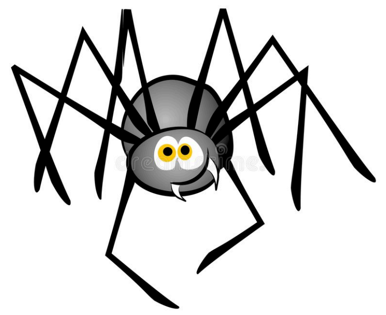


Figure 11. Manual error categorization for 50 wrong predictions on the Spider dev set.

Error Analysis

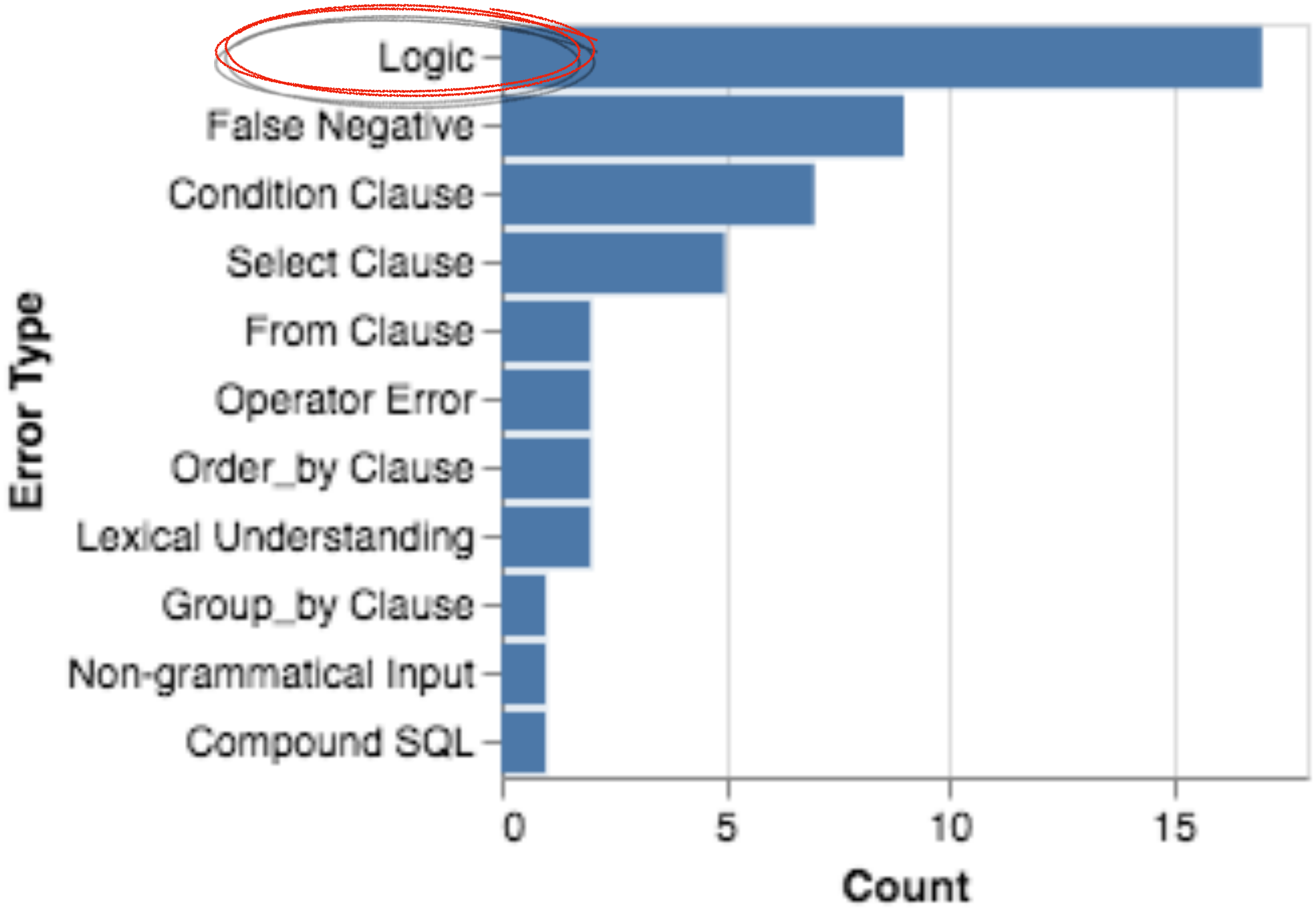


Figure 11. Manual error categorization for 50 wrong predictions on the Spider dev set.

Qualitative Examples

Error Category I - Logic

Logic	👤	*Find the number of concerts happened in the stadium with the highest capacity.* **concert_singer**
	✗	`SELECT COUNT(*) FROM stadium JOIN concert ON stadium.Stadium_ID = concert.Stadium_ID ORDER BY stadium.Capacity DESC LIMIT 1`
	✓	`SELECT COUNT(*) FROM concert WHERE stadium_id = (SELECT stadium_id FROM stadium ORDER BY capacity DESC LIMIT 1)`
	👤	*Show the names of all of the high schooler Kyle's friends.* **network_1**
	✗	`SELECT Highschooler.name FROM Friend JOIN Highschooler ON Friend.friend_id = Highschooler.ID WHERE Highschooler.name = "Kyle"`
	✓	`SELECT T3.name FROM Friend AS T1 JOIN Highschooler AS T2 ON T1.student_id = T2.id JOIN Highschooler AS T3 ON T1.friend_id = T3.id WHERE T2.name = "Kyle"`

Qualitative Examples

Error Category II - Lexical Understanding

Lexical Understanding

Count the number of countries for which Spanish is the predominantly spoken language. world_1

✗
```
SELECT COUNT(*) FROM countrylanguage WHERE countrylanguage.Language = "Spanish"
```

✓
```
SELECT COUNT(*), MAX(Percentage) FROM countrylanguage WHERE LANGUAGE = "Spanish"
GROUP BY CountryCode
```

What are the full names of all left handed players, in order of birth date? WTA_1

✗
```
SELECT first_name, last_name FROM players ORDER BY birth_date
```

✓
```
SELECT first_name, last_name FROM players WHERE hand = 'L' ORDER BY birth_date
```

Qualitative Examples

Error Category III - Robustness

Robustness

What is the model of the car with the smallest amount of horsepower? car_1

✗
```
SELECT cars_data.Horsepower FROM cars_data ORDER BY cars_data.Horsepower LIMIT 1
```

✓
```
SELECT T1.Model FROM CAR_NAMES AS T1 JOIN CARS_DATA AS T2 ON T1.MakeId = T2.Id
ORDER BY T2.horsepower ASC LIMIT 1
```

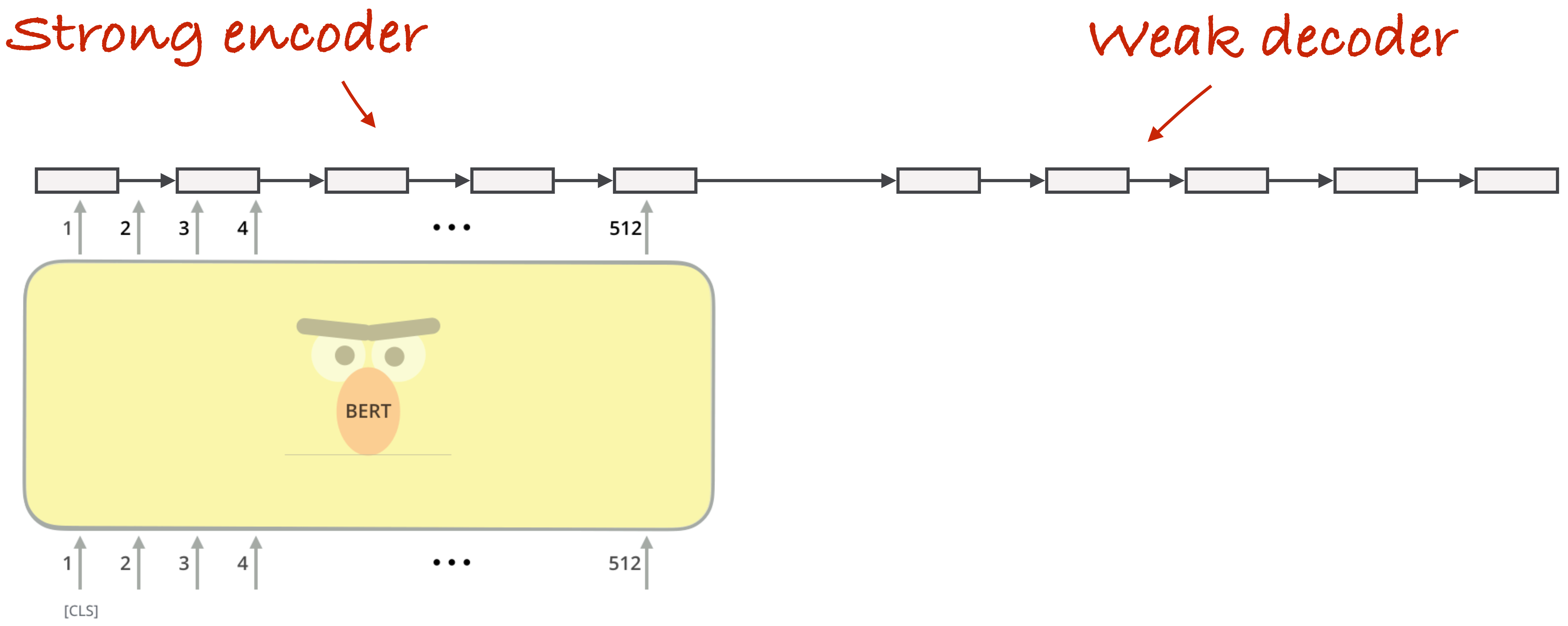
What is the total population and average area of countries in the continent of North America whose area is bigger than 3000? concert_singer

✗
```
SELECT SUM(country.Population), AVG(country.Population) FROM country WHERE
country.Continent = "North America" AND country.SurfaceArea 3000>
```

✓
```
SELECT SUM(country.population), AVG(country.surfacearea) FROM country WHERE
country.Continent = "north america" and country.SurfaceArea 3000>
```

Discussion

1. BRIDGE uses a sequential encoder for jointly encoding text, DB schema and relevant DB cells, and a sequential decoder for generating SQL queries. The decoder has significantly fewer parameters than the encoder.



LM Alternatives:

BART (Lewis et al. 2020)

T5 (Raffel et al. 2020)

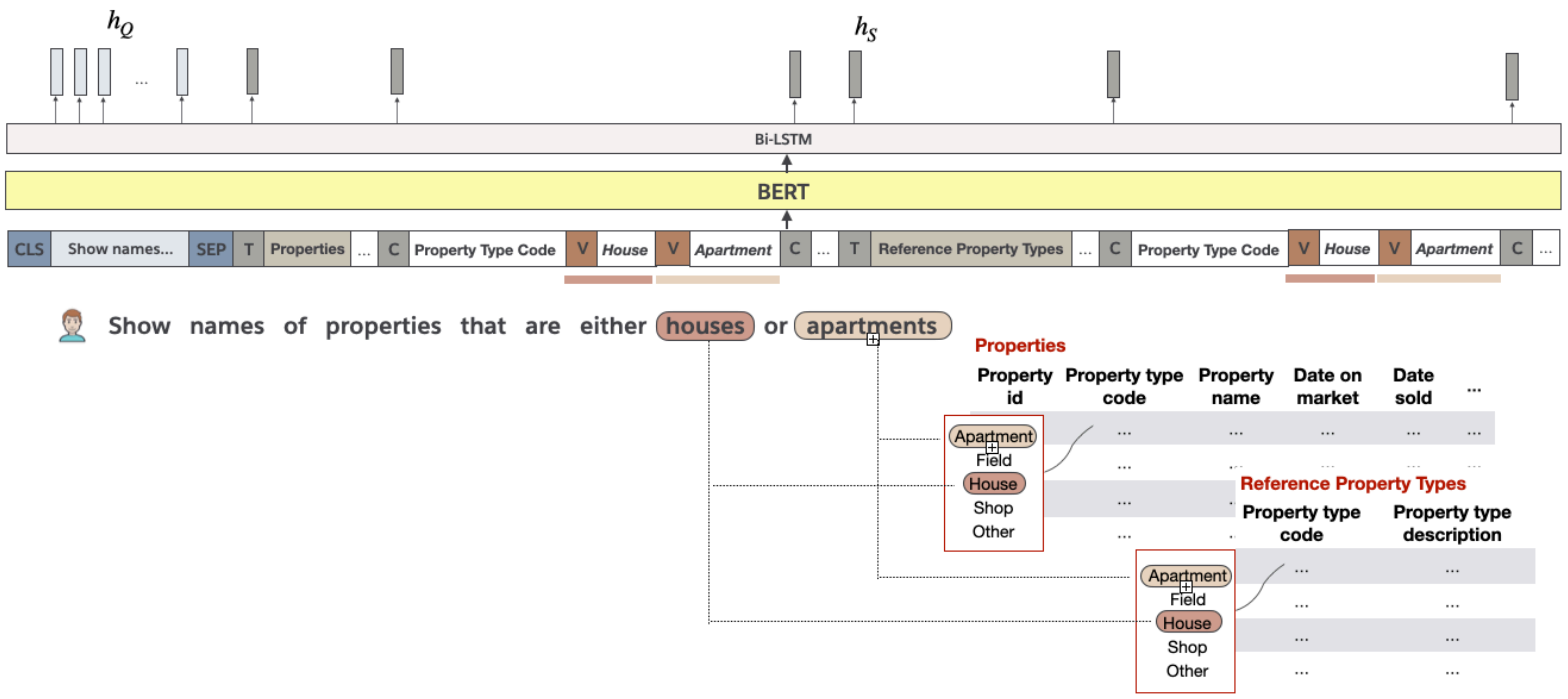
GPT-2 (Radford et al. 2019)

GPT-3 (Brown et al. 2020)

...

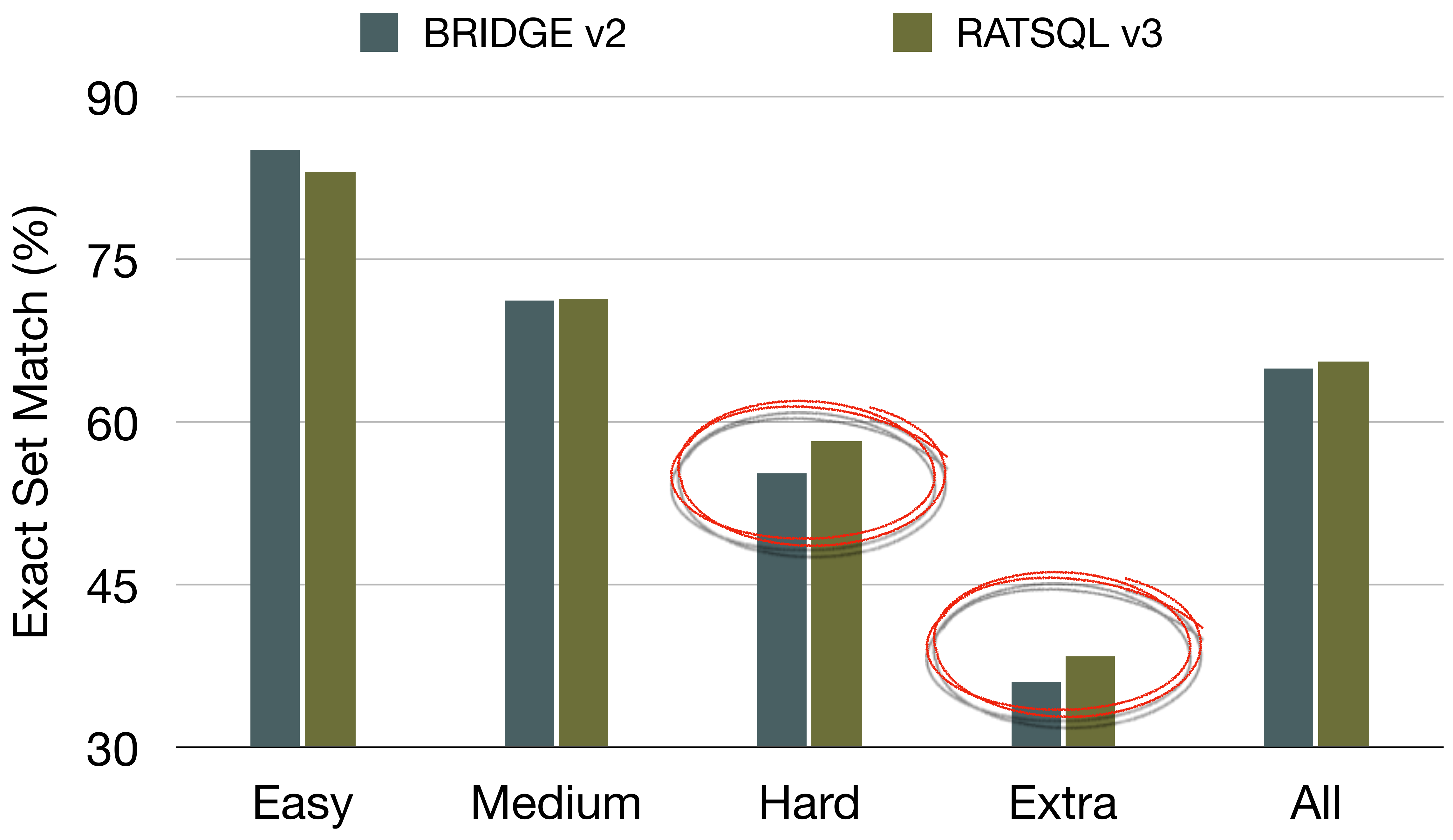
Discussion

II. Learning to recognize relevant cells (addressing acronyms and other lexical variations)



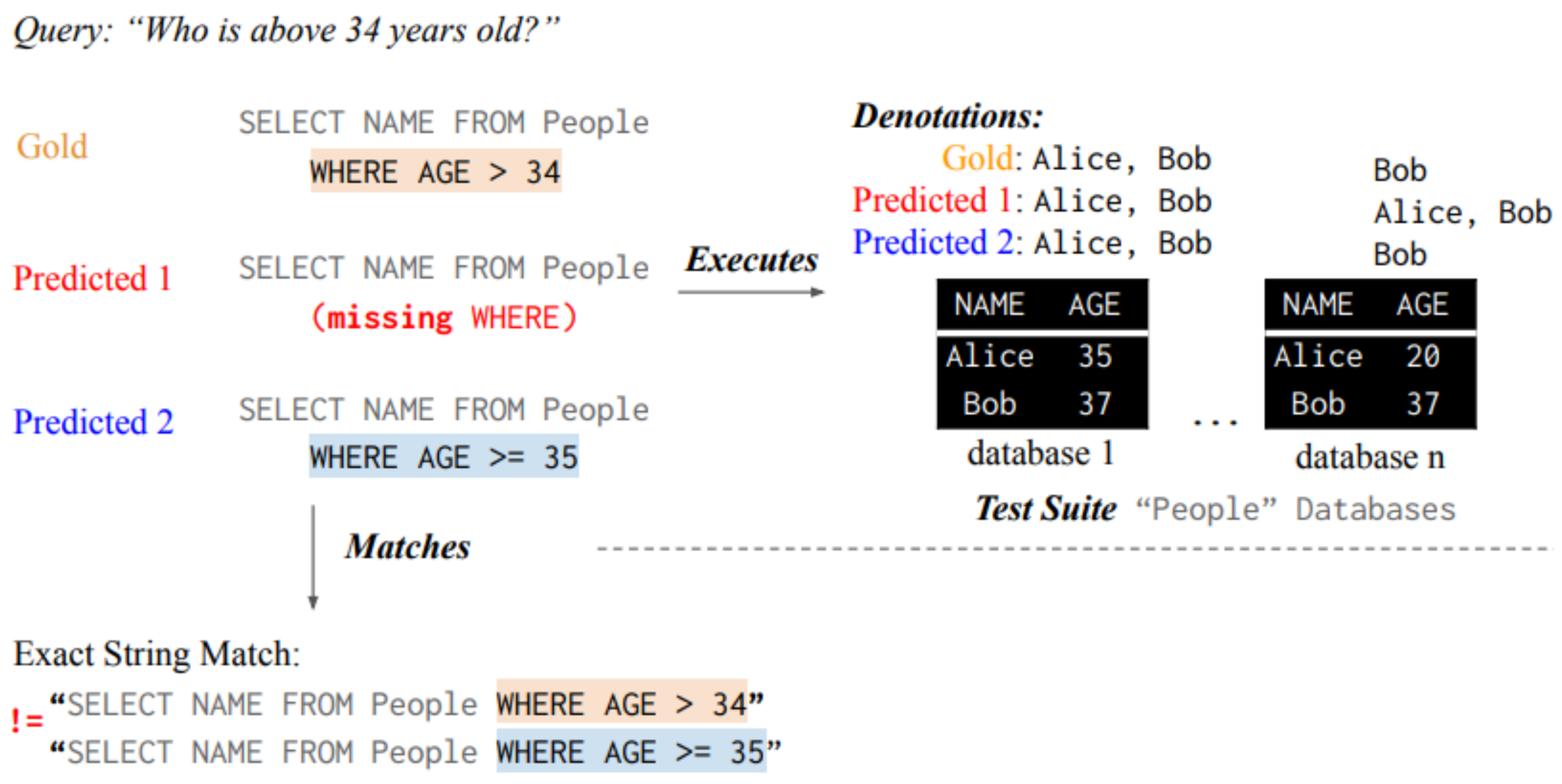
Discussion

III. Compositional inductive biases (CIBs) show significant benefit for synthesizing hard SQL queries. Previous work have shown that CIBs are effective for improving model's compositional generalization and overcoming data sparsity.



Discussion

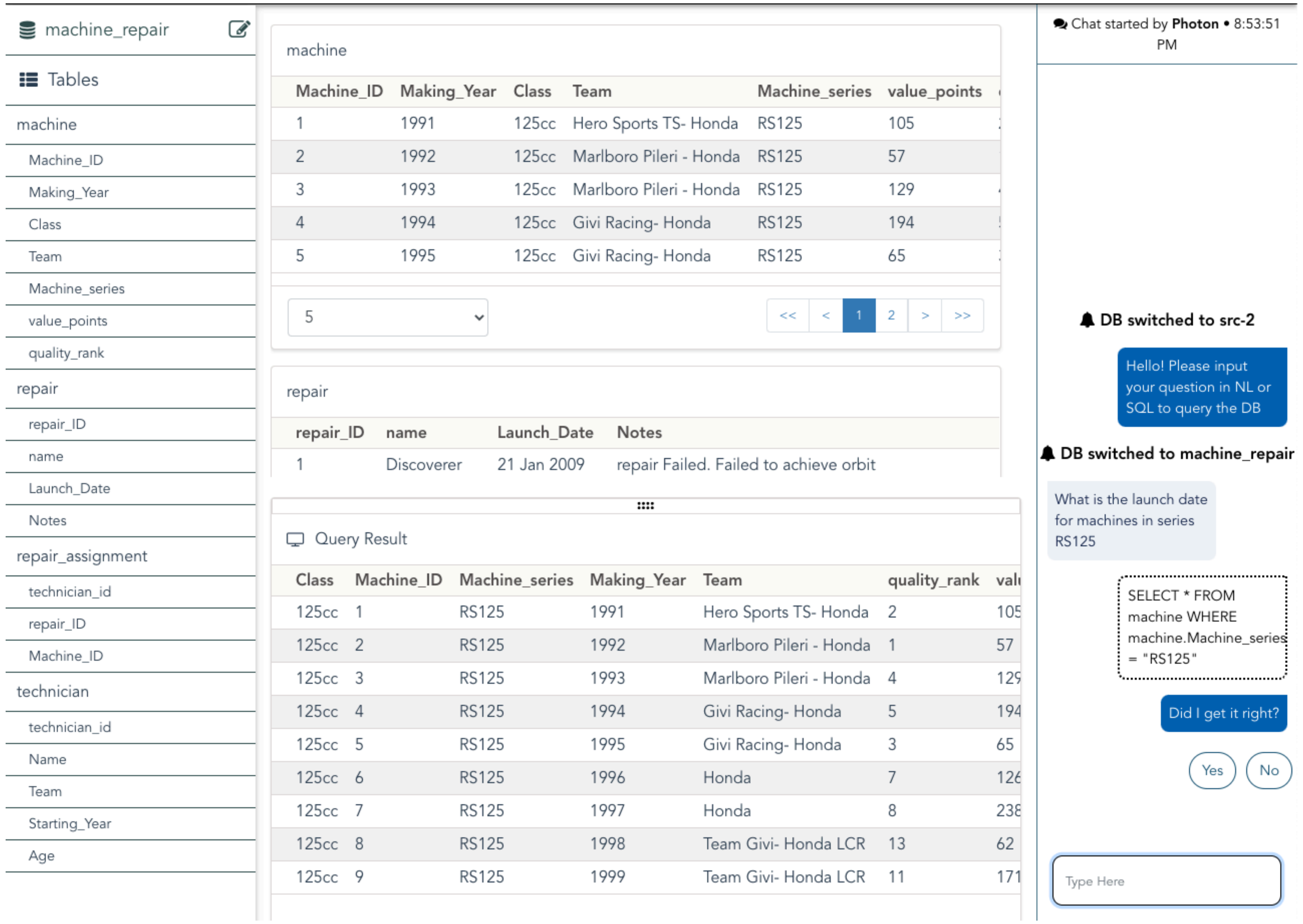
IV. More comprehensive model evaluation and benchmarking.



Distilled Test Suites by (Zhong et al. 2020)

Discussion

V. Existing benchmark datasets are not perfect.
- Sparse schema component coverage
- Sparse logic relation coverage
- Data synthesis?
- Interpolation?



Discussion

VI. More future directions

- Train with execution feedback
- Overcome data sparsity
- Interpretability and Explainability
- Process context and pragmatics
- Question answering over DBs, documents, and other modality of information
- …

https://github.com/salesforce/TabularSemanticParsing

Star 51 Fork 11

Pre-trained transformer LMs can effectively capture language-database grounding when the cross-modal data are serialized and tagged with special tokens

Two strategies significantly contribute to the overall text-to-SQL performance

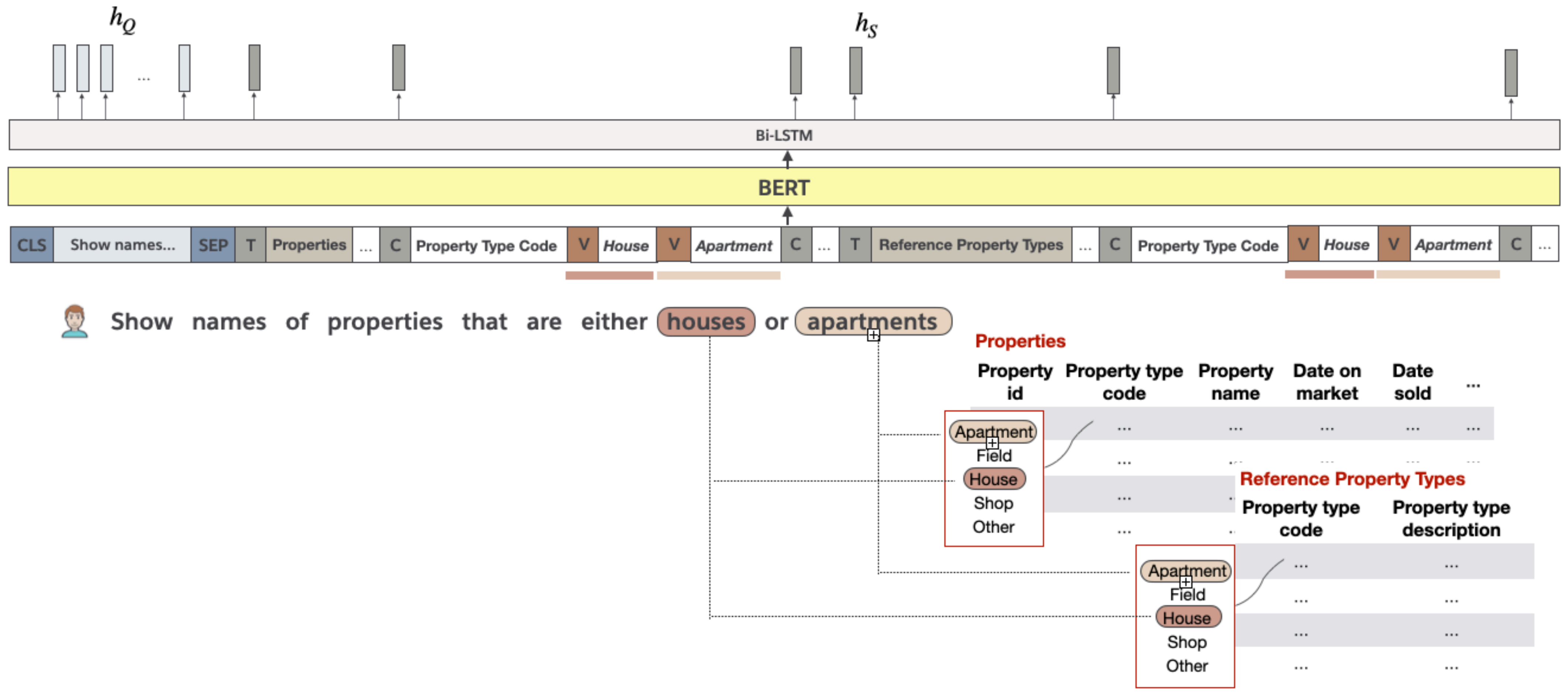
- The bridging mechanism that appends field values mentions (anchor texts) to the corresponding field names in the serialized representation
- Search-space pruning based on SQL syntax and schema consistency

Co-authors

Richard Socher
You.com

Caiming Xiong
Salesforce AI Research

salesforce research



Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Lin, Socher and Xiong. EMNLP Findings 2020.