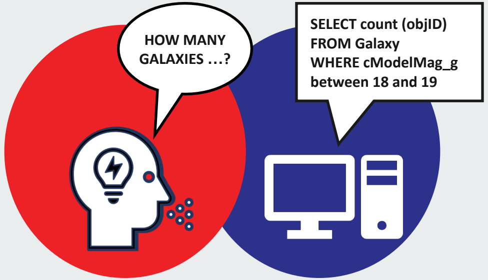
Data Democratisation with Deep Learning: An Analysis of Text-to-SQL Systems

The Web Conference 2022 Tutorial

George Katsogiannis-Meimarakis (katso@athenarc.gr)
Georgia Koutrika (georgia@athenarc.gr)

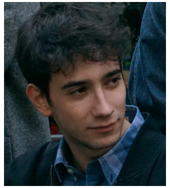
DARE Lab

ATHENA



Presenters

George Katsogiannis



- **Research Assistant** at Athena Research Center, Greece
  - Text-to-SQL
  - Data Democratisation
  - INODE Project
- MSc Student - Data Science and Information Technologies
  - Artificial Intelligence and Big Data specialisation

Georgia Koutrika

- **Research Director** at Athena Research Center, Greece
- Research interests:
  - data exploration, including natural language interfaces, and recommendation systems
  - big data analytics
  - large-scale information extraction, entity resolution and information integration

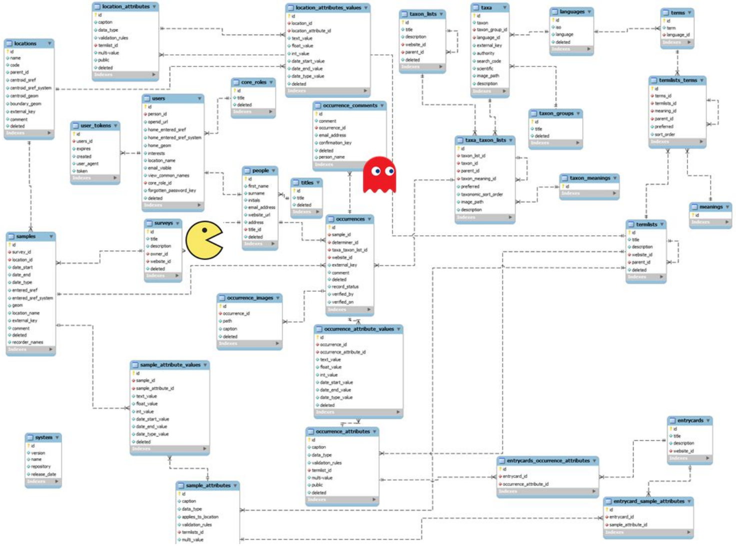
Why Text-to-SQL Systems?

- The imminent **age of information** has made data an indispensable part of all human activities
- Many different **data sets are being generated** by users, systems and sensors
- Data repositories can benefit **many types of users** looking for insights, patterns, information, etc.
- However, not all users have **equal access to data**

Why Text-to-SQL Systems?

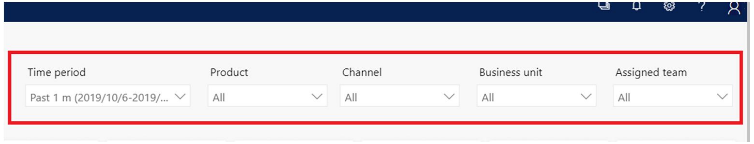
Databases are complex

- Data **volume and complexity** make it difficult to query data



Why Text-to-SQL Systems?

Data Query Interfaces are user-unfriendly

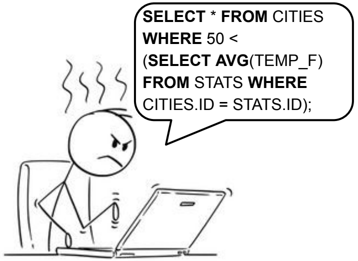


Form-based interfaces have limited query capabilities

Low-level query interfaces are intended for programmers

Why Text-to-SQL Systems?

- Data **volume and complexity** make it difficult to query data
- Database query interfaces are notoriously **user-UNFRIENDLY**



What is **data democratisation**?

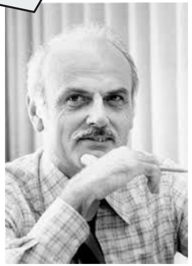
- **Empower everyone** to access, use, understand and derive value from data
- Lift the **technical barriers** that impede access to data and **eliminate dependency** to IT experts
- Design tools that are aimed for the **casual user**
- An organization-wide cultural stance

Why Text-to-SQL Systems?

Which cities have year-round average temperature above 50 degrees?

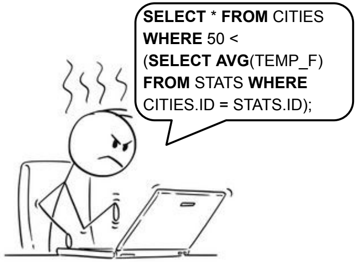
To satisfy the needs of casual users of databases,
we must break through the barriers that presently prevent
these users from freely **employing their native languages**

Ted Codd (circa: 1974)



Expressing queries in natural language can open up data access to everyone

Tutorial Outline

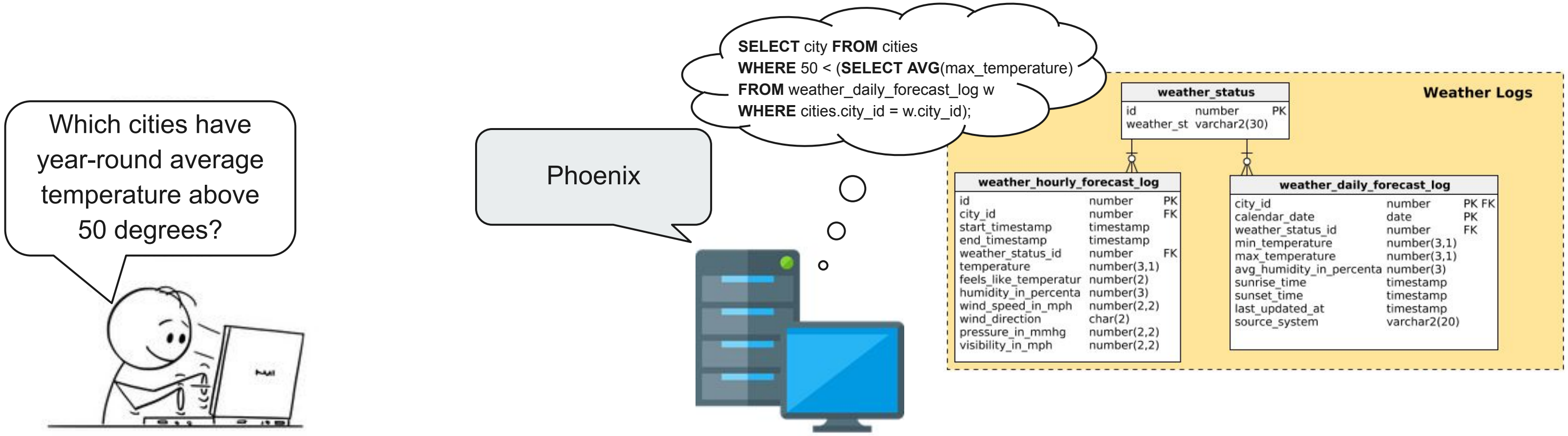


1. The Text-to-SQL Problem - 5’
2. Available Benchmarks - 5’
3. A Taxonomy of Text-to-SQL Deep Learning Systems - 35’
4. Key Text-to-SQL Systems - 20’
5. Challenges and Research Opportunities - 10’

1. Schema Linking
2. Language Processing
3. Input Encoding
4. Output Decoding
5. Neural Training
6. Output Refinement

The Text-to-SQL Problem

The Text-to-SQL Problem



Challenges: From the NL side

- Complexity of NL
  - Ambiguity
  - References - Schema Linking
  - Inferences
  - Vocabulary Gap
- User Mistakes
  - Spelling mistakes
  - Syntactical/Grammatical mistakes

City or person?

"Show information about Paris"

"model" refers to car.model OR engine.model ?

"President *(of the USA)* before Obama?"

"composer" vs "songwriter"

Grammys

"Which singer won the most Grammies?"

"Show most actor played movies "

??

Challenges: From the SQL side

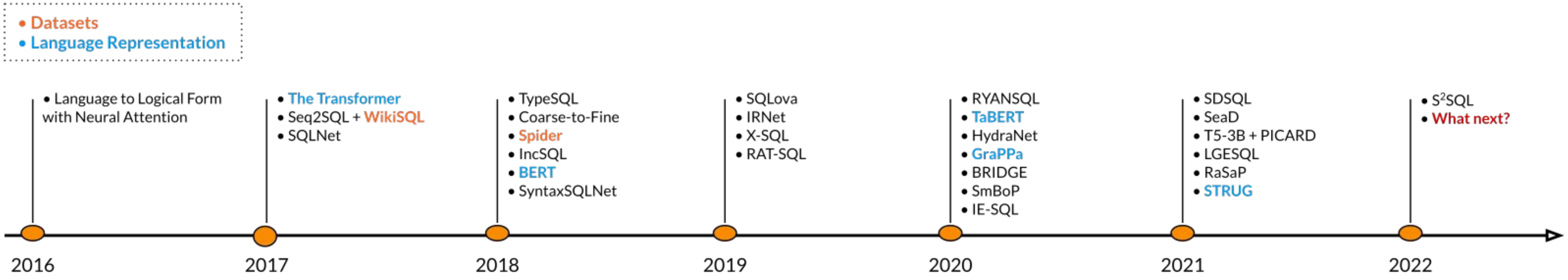
- Complex Syntax
    - SQL is a structured language with a strict grammar and limited expressivity
- Database Structure
    - The user's data model may not match the data schema

"Which countries have a GDP higher than the EU average?"

Sounds simple but needs a complex nested query

"Find directors who released a movie this year"

Simple NLQ that might need 3,4 or 5 JOINs



A brief timeline of deep learning text-to-SQL research

Available Benchmarks

Text-to-SQL Benchmarks

Several pain points:

✘ **No common datasets**
- System evaluations have used different datasets of varying size and complexity.

✘ **Small or proprietary datasets**
- e.g., TPC-H (100MB) and DBLP (56MB)

✘ **No standard, small query sets**
- Different test queries, often not available to reproduce the experiments.

✘ **Incomparable effectiveness evaluations**
- none, user study, manual evaluation, comparison to gold standard queries

Year	Dataset	Examples	Databases
1994	ATIS	275	1
1996	GeoQuery	525	1
2003	Restaurants	39	1
2014	Academic	179	1
2017	IMDb	111	1
	Yelp	68	1
	Scholar	396	1
	WikiSQL	**80,654**	**24,241**
2018	Advising	281	1
	Spider	**10,181**	**200**
2020	MIMICSQL	10,000	1
2021	Spider-Syn	8,034	160
	Spider-DK	535	?
	KaggleDBQA	272	8

Two new large benchmarks revolutionise text-to-SQL research, opening the door to machine learning

WikiSQL

- Large crowd-sourced dataset for developing NL interfaces for relational databases
  - 80K NL/SQL pairs over 25K tables
- NL questions on tables gathered from Wikipedia
  - Not entire databases!
  - The SQL queries that can be performed are quite simple
- Contains many mistakes
  - Research suggests that the upper bound has been reached
  - Human accuracy estimated at 88%

WikiSQL: Example

NLQ:

What nationality is the player Muggsy Bogues?

SQL:

```
SELECT nationality
WHERE player = muggsy bogues
```

Player	No.	Nationality	Position	Years in Toronto	School /Club Team
Leandro Barbosa	20	Brazil	Guard	2010-2012	Tilibra
Muggsy Bogues	14	USA	Guard	1999-2001	Wake Forest
Jerryd Bayless	5	USA	Guard	2010-2012	Arizona
...	...	...	...	...	...

Table: Toronto Raptors all-time roster

WikiSQL: (Bad) Example

NLQ:

Name the most late 1943 with late 194 in slovenia

SQL:

```
SELECT max(late 1943)
WHERE ! late 1941 = slovenia
```

A table copied incorrectly from Wikipedia resulted to the generation of a SQL query that does not make much sense and a NLQ that is even more incoherent!

Wikipedia (original table)

	Late 1941	Late 1942	Sept. 1943	Late 1943	Late 1944
Bosnia and Herzegovina	20,000	60,000	89,000	108,000	100,000
Croatia	7,000	48,000	78,000	122,000	150,000
Serbia (Kosovo)	5,000	6,000	6,000	7,000	20,000
Macedonia	1,000	2,000	10,000	7,000	66,000
Montenegro	22,000	6,000	10,000	24,000	30,000
Serbia (proper)	23,000	8,000	13,000	22,000	204,000
Slovenia[82][83][84]	2,000	4000	6000	34,000	38,000
Serbia (Vojvodina)	1,000	1,000	3,000	5,000	40,000
Total	**81,000**	**135,000**	**215,000**	**329,000**	**648,000**

WikiSQL (badly copied)

! Late 1941	Late 1942	Sept. 1943	Late 1943	Late 1944	1978 Veteran membership
Croatia	7000	48000	78000	122000	150000
Slovenia	2000	4000	6000	34000	38000
Serbia	23000	8000	13000	22000	204000
...	...	...	...	...	...

Table: Yugoslav Partisans: Composition

Spider

- Large-scale complex and cross-domain semantic parsing and text-to-SQL dataset
  - 10,181 questions
  - 5,693 complex SQL queries
  - 200 databases from 138 different domains
- Annotated by 11 Yale students
- Queries of varying complexity
  - Categories of difficulty: Easy → Medium → Hard → Extra Hard
  - SQL elements such as JOIN, GROUP BY, UNION, INTERSECT, nested queries
- Better quality and complexity than WikiSQL

Spider: Example

Easy

What is the number of cars with more than 4 cylinders?

```
SELECT COUNT(*)
FROM cars_data
WHERE cylinders > 4
```

Medium

For each stadium, how many concerts are there?

```
SELECT T2.name, COUNT(*)
FROM concert AS T1 JOIN stadium AS T2
ON T1.stadium_id = T2.stadium_id
GROUP BY T1.stadium_id
```

Hard

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```

Extra Hard

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)
FROM country
WHERE name NOT IN
    (SELECT T1.name
     FROM country AS T1 JOIN
     country_language AS T2
     ON T1.code = T2.country_code
     WHERE T2.language = "English"
       AND T2.is_official = "T")
```

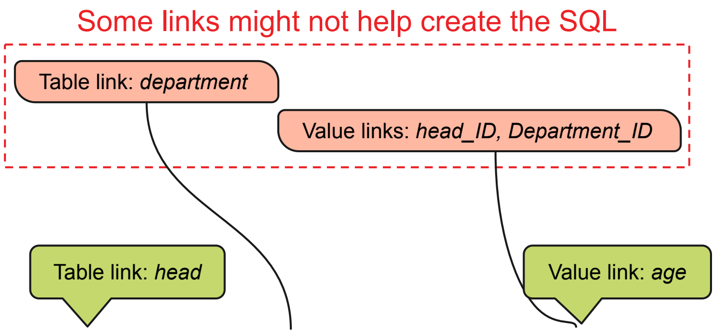
A Taxonomy of Text-to-SQL Deep Learning Systems

Taxonomy Overview of a Deep Learning Text-to-SQL system

Schema Linking

Finding connections between the NLQ and the DB

- Consider a human writing a SQL query based on a NL specification
- Important to find how elements of the NL appear in the DB
- Three main types of schema links:
  - Table links
  - Column links
  - Value links

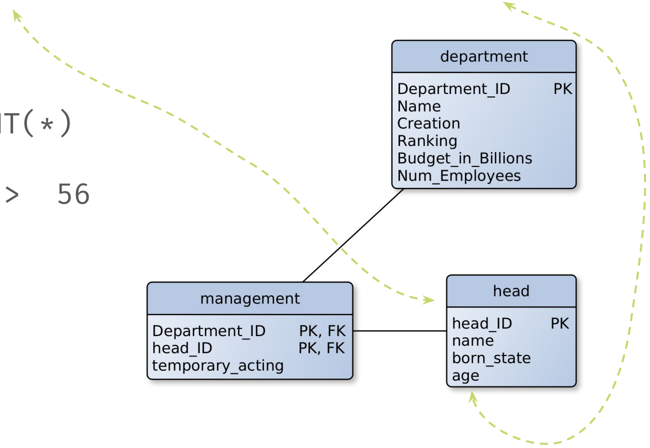


NLQ:

How many heads of the departments are older than 56 ?

SQL:

```
SELECT COUNT(*)
FROM head
WHERE age > 56
```



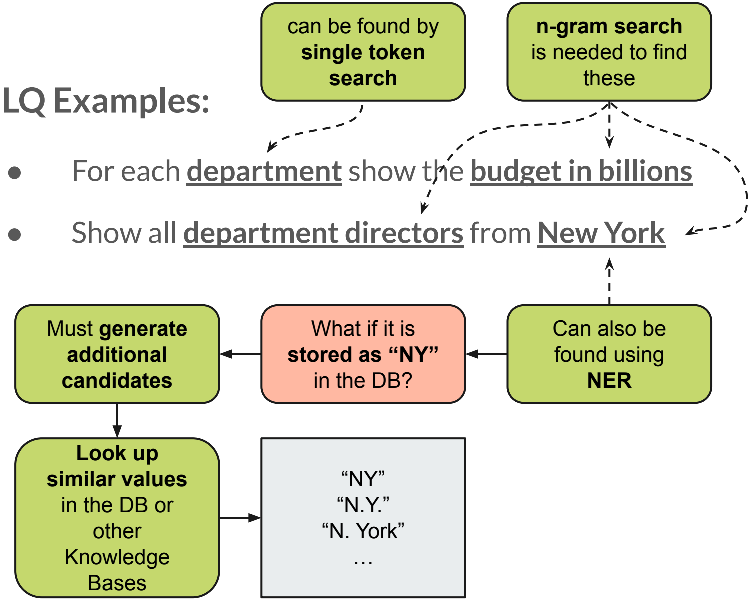
Schema Linking: Query Candidates

The three questions of schema linking:

- Which parts of the **NLQ** to consider?
    - Single Tokens
    - Multi-word candidates (n-grams)
    - Named Entities
    - Generate Additional Candidates
- Which parts of the **DB** to consider?
- How to decide on a **match**?

NLQ Examples:

- For each **department** show the **budget in billions**
- Show all **department directors** from **New York**

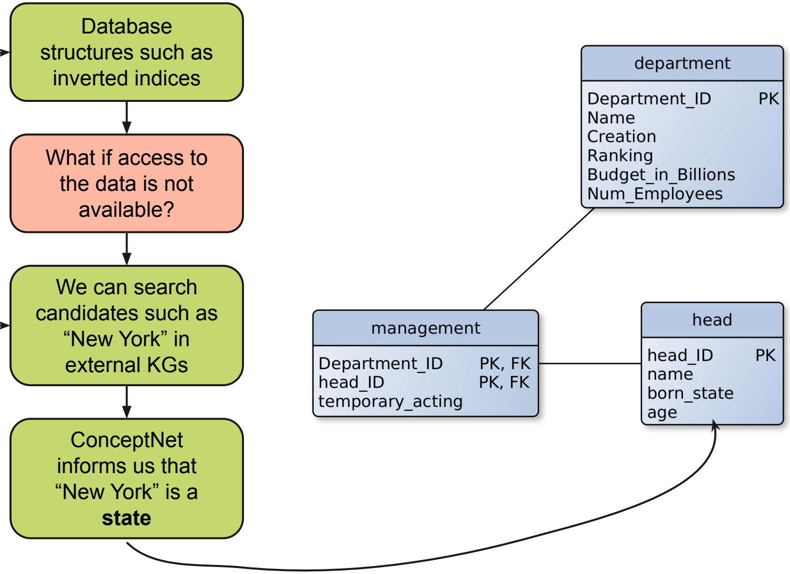


Schema Linking: Database Candidates

The three questions of schema linking:

- Which parts of the **NLQ** to consider?
- Which parts of the **DB** to consider?
  - Table and Column Names
  - Values via Lookup
  - Values via Knowledge Graphs
- How to decide on a **match**?

Need an efficient method due to large size of data

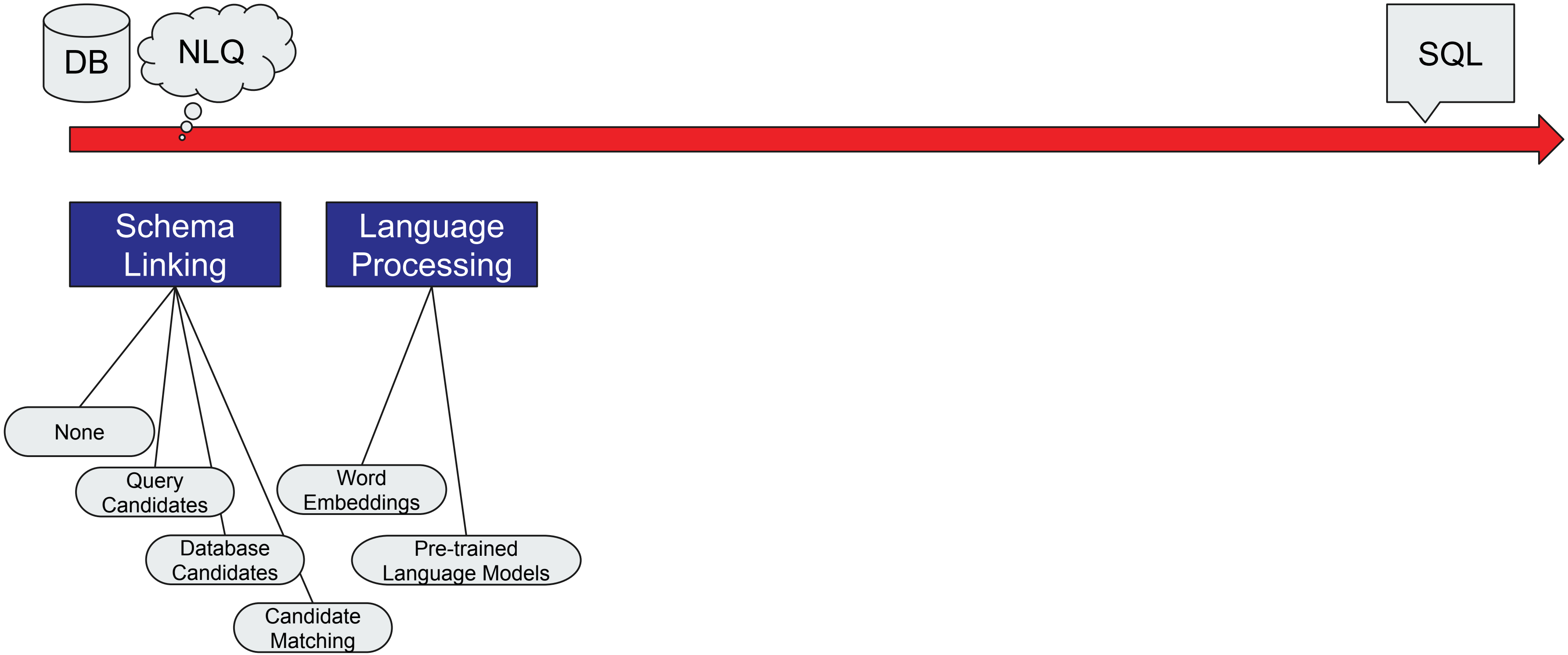


Schema Linking: Candidate Matching

The three questions of schema linking:

- Which parts of the **NLQ** to consider?
- Which parts of the **DB** to consider?
- How to decide on a **match**?
  - Exact and partial match
  - Fuzzy/Approximate String Matching
  - Learned Embeddings
  - Classifiers

Query Candidate	DB Candidate	Match Method
“department”	“department”	Exact Match
“budget”	“budget in billions”	Partial Match
“dept.”	department	Fuzzy Match
“department director”	“head”	Learned Embeddings
		Classifiers



Taxonomy Overview of a Deep Learning Text-to-SQL system

Natural Language Processing

How can we give natural language to a neural network?

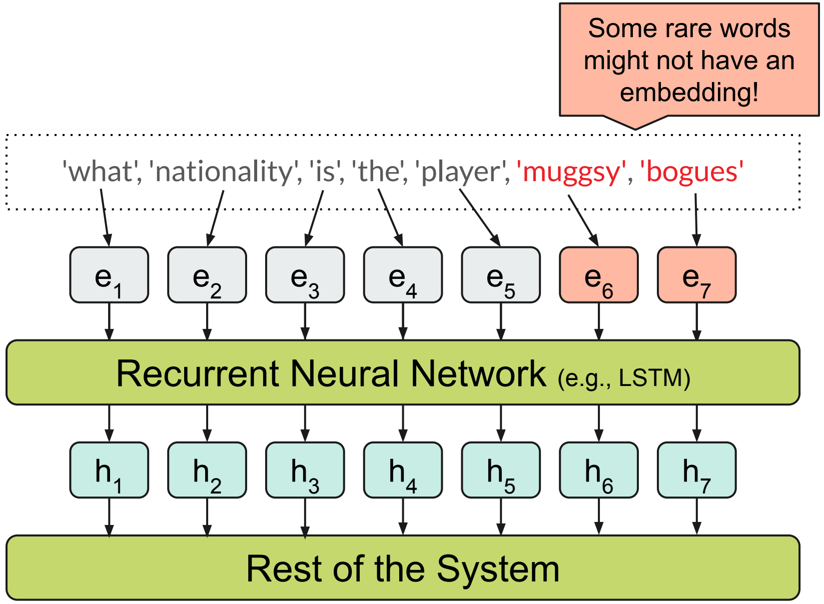
- LSTM Neural Networks (1995) 🔗 [5]
- Word Embeddings
  - One-hot Embeddings
  - Word2Vec (2013) 🔗 [6]
  - GloVe (2014) 🔗 [7]
- The Transformer (2017) 🔗 [9]
- The rise of language models
  - BERT (2018) 🔗 [10]
  - RoBERTa (2019) 🔗 [11]
  - TaBERT (2020) 🔗 [12]
  - GraPPa (2020) 🔗 [13]
  - BART (2020) 🔗 [28]
  - T5 (2020) 🔗 [29]

encoder-only

encoder-decoder

Using Word Embeddings

- Each word of the input is assigned to a pre-trained word embedding vector
    - **Out of vocabulary** problem
- The embedding sequence is then processed by a RNN to create a hidden representation
- Major drawbacks of RNNs:
    - Large **processing costs** for long sequences
    - Hard to make associations of words that are not near each other



Using Transformer-based PLMs: BERT

- A very large pre-trained neural network
  - BERT Base: 110M parameters
  - BERT Large: 340M parameters
- Can be applied to a wide variety of NL tasks
  - The pre-trained model is fine-tuned with additional **task-specific layers**
  - Provided very good results (usually state-of-the-art) in many NL tasks
- Based on Transformer neural networks
  - Each element of the sequence is processed simultaneously, decreasing computation costs
  - All outputs are based on all other elements of the sequence, using attention
- Uses WordPiece embeddings to eliminate the out-of-vocabulary problem

GloVe vs Wordpiece

NLQ: What nationality is the player Muggsy Bogues?

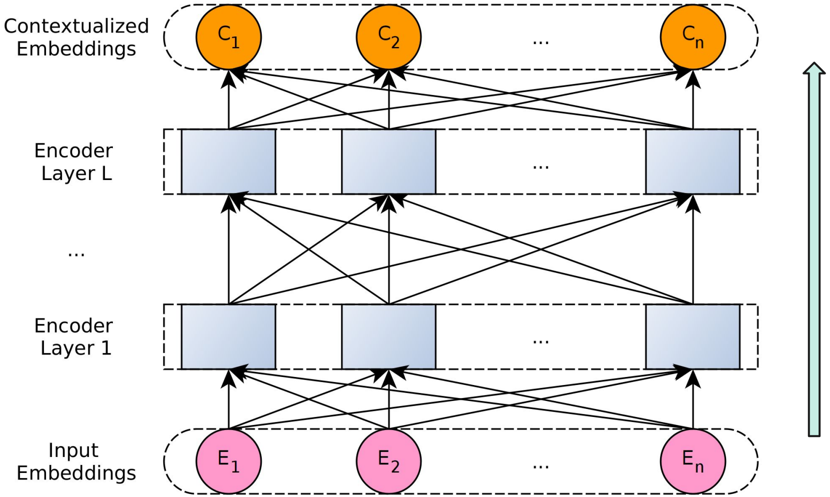
- GloVe:
    - 'what', 'nationality', 'is', 'the', 'player', 'muggsy', 'bogues', '?'
- Wordpiece:
    - 'what', 'nationality', 'is', 'the', 'player', 'mug', '##gs', '##y', 'bog', '##ues', '?'

Unknown rare words

Known sub-words

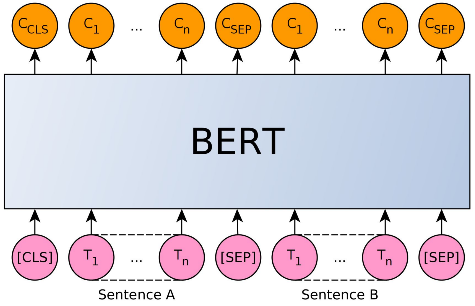
Using sub-words, we **eliminate** the possibility for out-of-vocabulary words, as long as all **characters** were also present during the creation of the embeddings

BERT: Architecture



Notice the **encoder-only** architecture, which produces a **contextualized embedding** output

- **Output**: A sequence of tokens of equal length to the input
- Uses many stacks of **bidirectional Transformer** encoder layers
- **Input**: A sequence of token embeddings
  - Uses Wordpiece embeddings



BERT: Pre-training & Fine-tuning

Pre-training:

- Training corpus of 3.3B words
  - BooksCorpus (800M words)
  - English Wikipedia (2.5B words)
- The model is **simultaneously** pre-trained on two tasks
  - Masked Language Modeling (**MLM**)
  - Next Sentence Prediction (**NSP**)

Fine-tuning:

- An application of **Transfer Learning**
  - We have a model (BERT) trained on a very large corpus and a more **general task**
  - We add some extra layers and perform additional training on **our task**
- We must make two decisions
  - How to give our task's **input** to BERT
  - How to use BERT's **output** to make predictions for our task

Task-specific PLMs: GraPPa

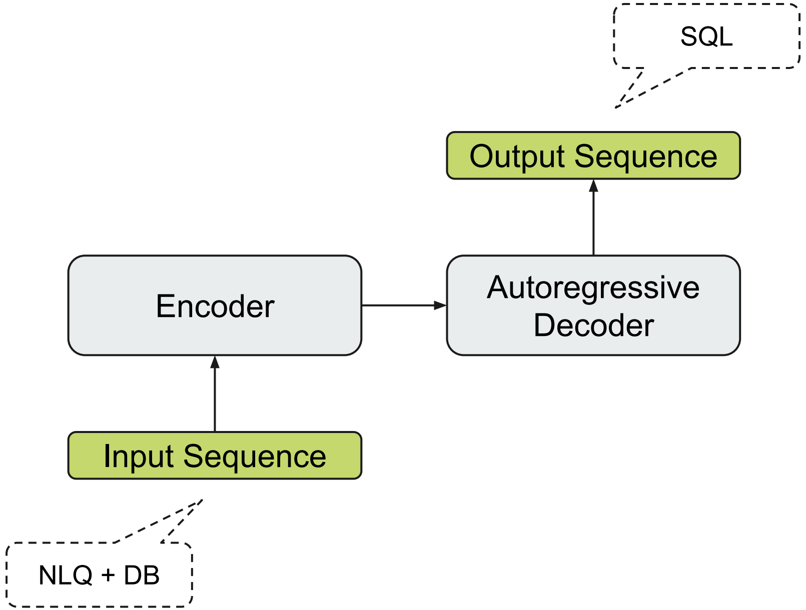
- Initialized by RoBERTa-Large
- Synthetic pre-training **data** is created from tabular datasets like:
  - Spider
  - WikiSQL
  - WikiTableQuestions
- Experiments show **better performance in text-to-SQL** when using GraPPa instead of RoBERTa

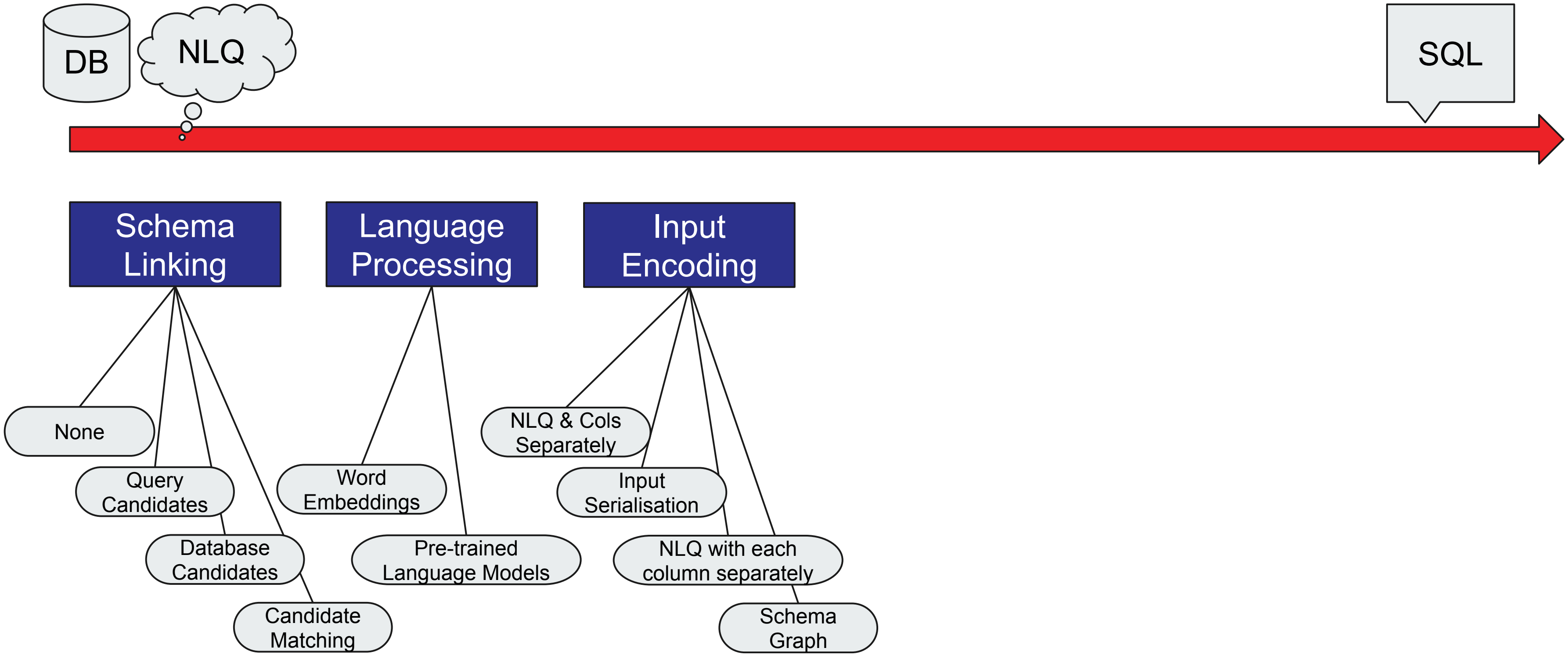
Pre-training **tasks**:

- Masked Language Modelling (MLM)
  - Input: NLQ/Table Description + Columns
  - The network must **predict the masked words** both in the NLQ and columns
- SQL Semantic Prediction (SSP)
  - Input: NLQ + Columns
  - The network must predict for each column, **if it appears in the SQL and its role** (e.g. SELECT, GROUP BY)

Encoder-Decoder PLMs

- Another category of very powerful transformer-based pre-trained models
- Operate on a **sequence-to-sequence** (text-to-text) framework
- Limited design choices, but very good results (e.g., T5-3B + PICARD)

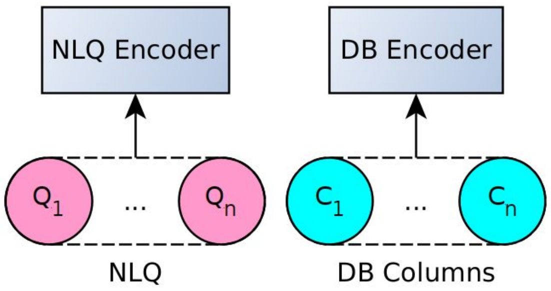




Taxonomy Overview of a Deep Learning Text-to-SQL system

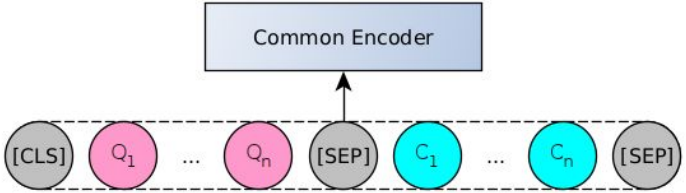
Input Encoding: Separate Encoding

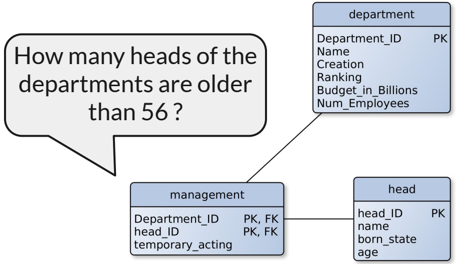
- Used by the first text-to-SQL systems (Seq2SQL, SQLNet) for WikiSQL
- The main reason is the **different format** of the NLQ and table columns
  - **NLQ:** Sequence of words
  - **Column names:** Sequence of sequences of words
- The two different inputs **must be combined** (attention, concatenation, sum, etc.)



Input Encoding: Serialisation

- Widely used by newer systems incorporating language models
- No need to combine different inputs
- The database schema is flattened into a sequence of words

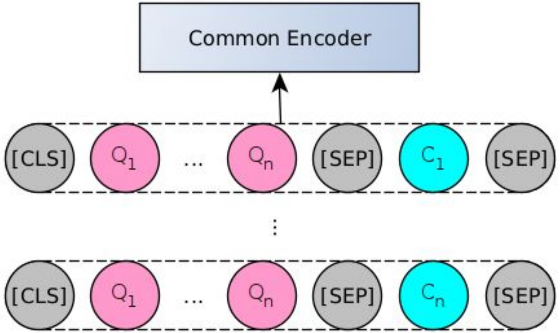




'How', 'many', 'heads', 'of', 'the', 'departments', 'are', 'older', 'than', '56, '?', [SEP], 'department', [SEP], 'name', [SEP], 'creation', [SEP], 'ranking', [SEP], 'budget_in_billions', [SEP], 'num_employes', [SEP], 'management', [SEP], 'department_id', [SEP], 'head_id', [SEP], 'temporary_acting', [SEP], 'head', [SEP], 'head_id', [SEP], 'name', [SEP], 'born_state', [SEP], 'age', [SEP]

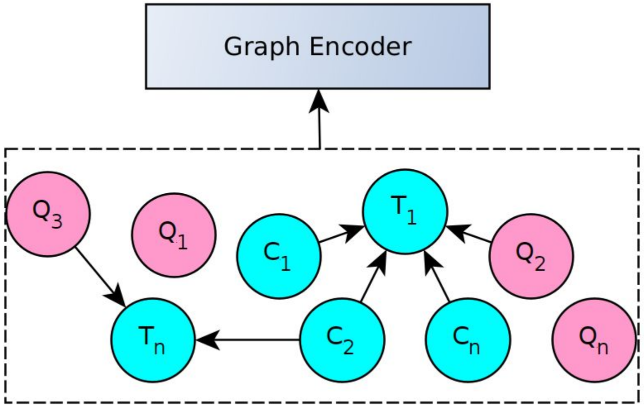
Input Encoding: NLQ with Each Column Separately

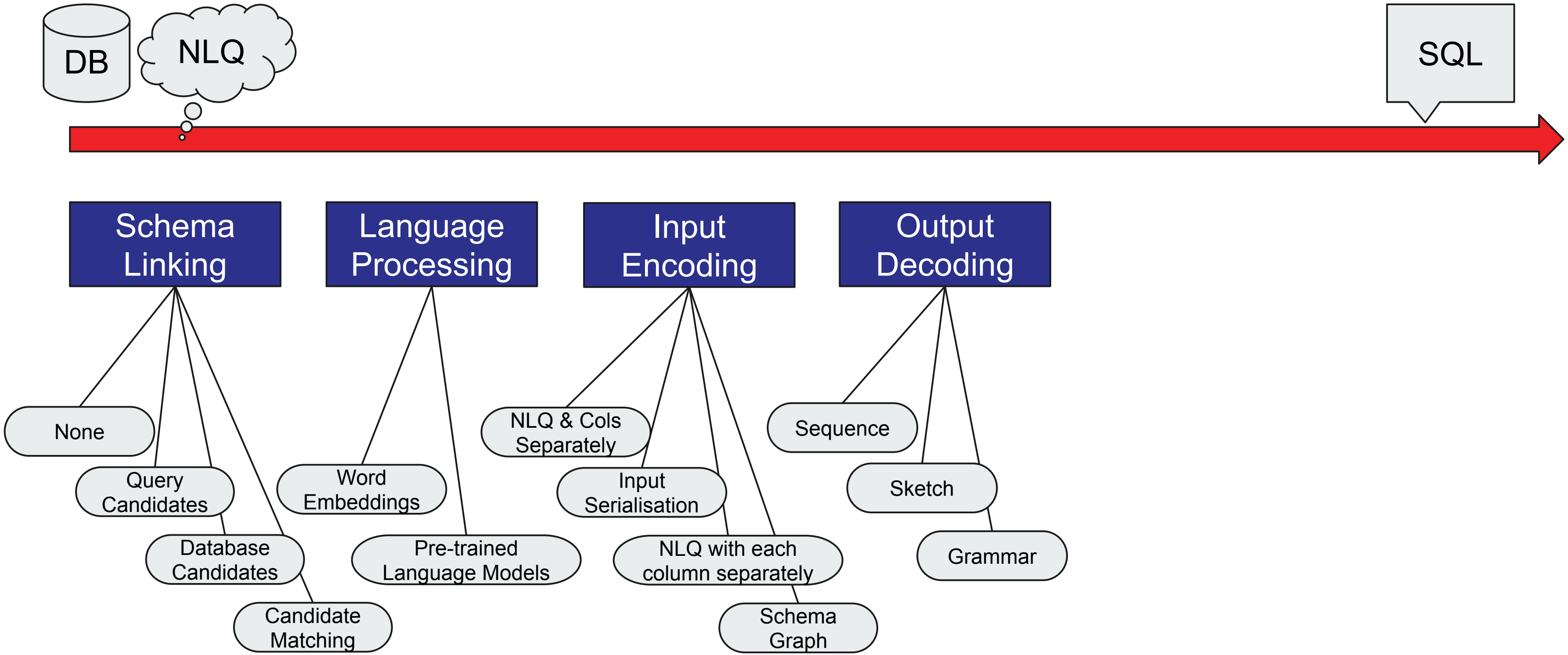
- A unique approach proposed by **HydraNet** (more later on)
- The NLQ is **processed** with each column **separately**
- **Predictions** are made for each column **separately**
- Works very well on **WikiSQL**
- No similar approach for **Spider**



Input Encoding: Graph Encoding

- Using graphs allows the preservation of all the **schema relations**
  - Which columns belong to which table
  - Which columns are keys
  - Which tables are connected by foreign keys
- The **words of the NLQ** can be added to the graph based on schema links and similarity
- Much more **complex** neural design





Taxonomy Overview of a Deep Learning Text-to-SQL system

Output Decoding: Sequence-based

🔗 [14] Language to Logical Form with Neural Attention (2016)
🔗 [2] Seq2SQL (2017)
🔗 [22] BRIDGE (2020)
🔗 [30] T5-3B + PICARD (2021)

- We consider two sequences:
  - NLQ (input sequence)
  - SQL query (output sequence)
- Text-to-SQL becomes a sequence-to-sequence transformation problem
  - The network learns to generate a sequence of tokens, which is the SQL query

👍 Simplifies the text-to-SQL problem

👎 More possibilities for errors
  - Nothing prevents syntactical errors when predicting
  - Usually avoided until recently
  - Recent works show promising techniques that help avoid such errors

Output Decoding: Sketch-based

[15] SQLNet (2017)
[16] SQLova (2019)
[17] HydraNet (2020)

- We have a sketch of the query with missing parts that need to be filled
- Sketch used by systems designed for WikiSQL

```
SELECT <AGG> <COLUMN>
(
     WHERE <COLUMN> <OP> <VALUE>
     ( AND <COLUMN> <OP> <VALUE> ) *
) ?
```

Further simplifies the task of producing a SQL query into smaller sub-tasks

Hard to extend for complex queries

Output Decoding: Grammar-based

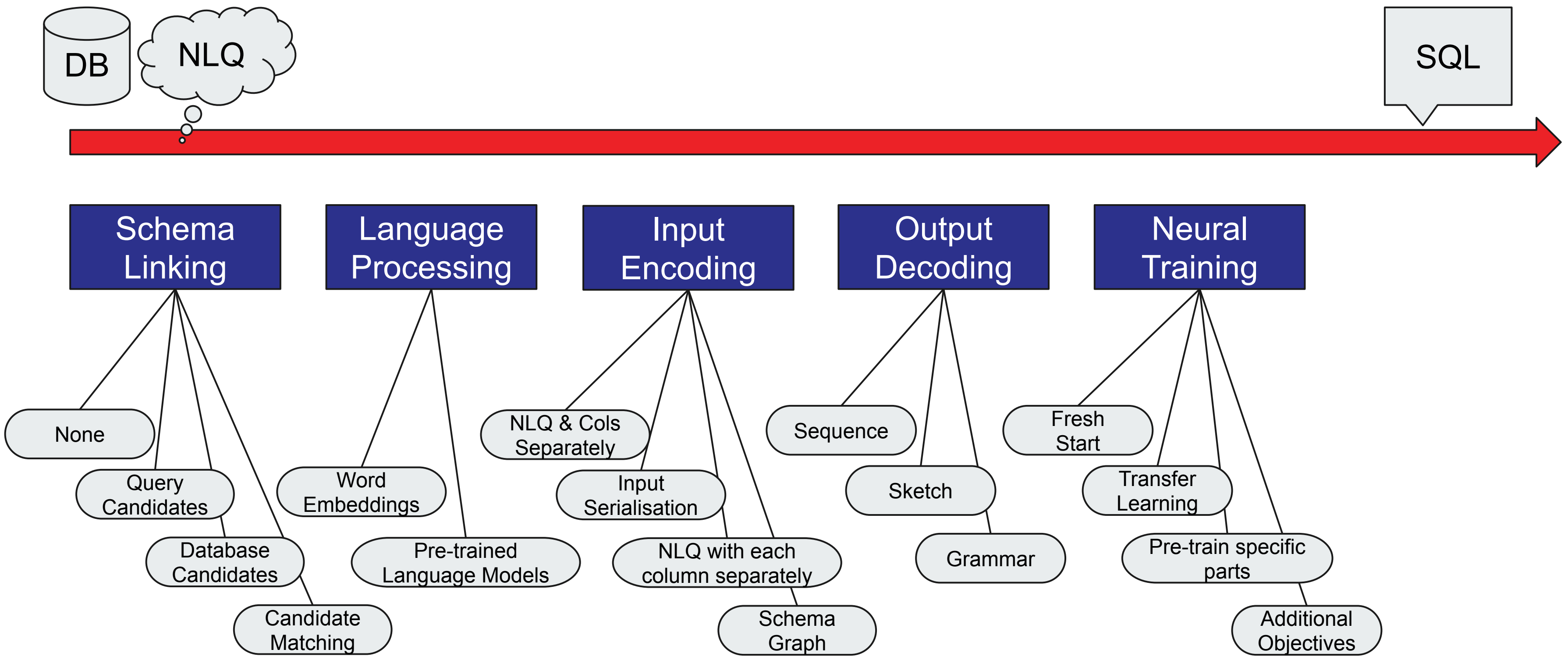
🔗 [18] IncSQL (2018)
🔗 [19] IRNet (2019)
🔗 [20] RAT-SQL (2020)

- Generate a sequence of rules instead of simple tokens
- Apply the rules sequentially to get a SQL query

👍 Easier to avoid errors

👍 Can cover more complex SQL queries

👎 Needs more complex design



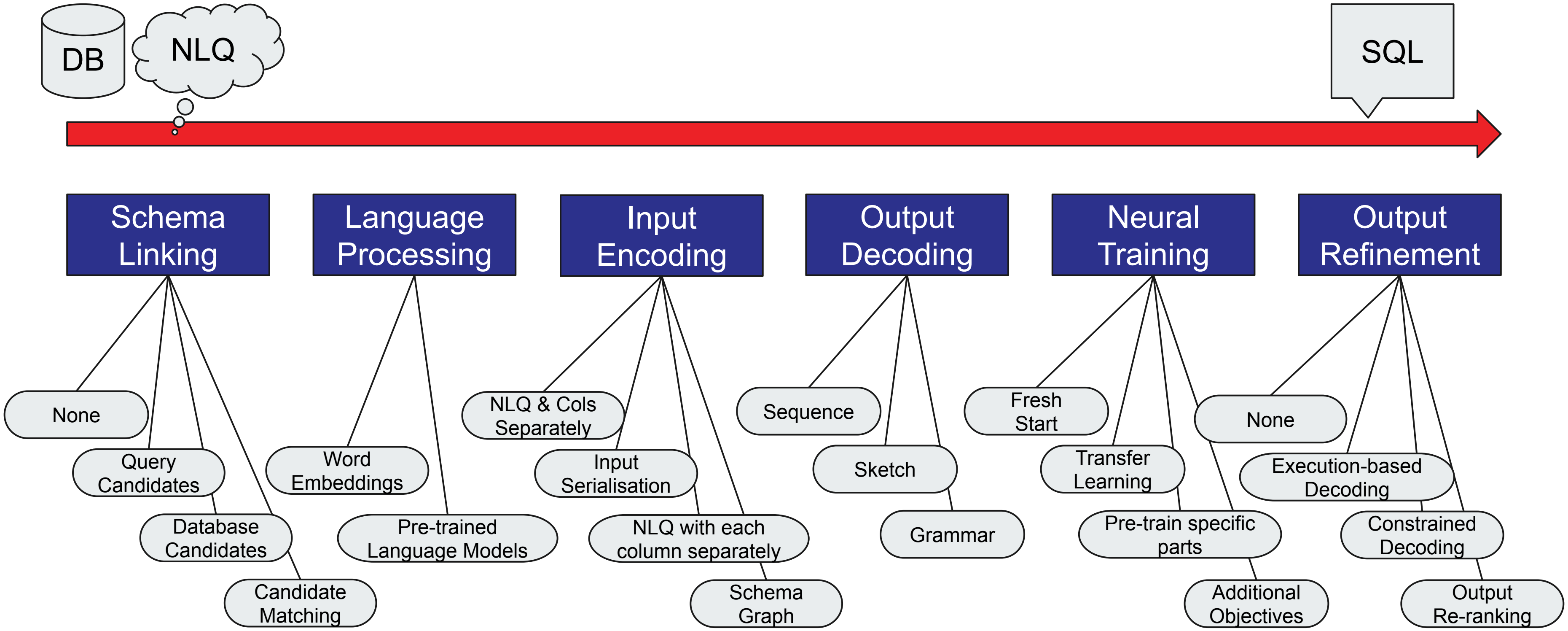
Taxonomy Overview of a Deep Learning Text-to-SQL system

Neural Training

Erosion: Delete parts of the DB schema and train the model to produce the correct SQL with the eroded schema

Shuffling: Randomly change the order of attributes and conditions in both the NLQ and SQL and train the network to re-order them correctly

1. **Fresh Start:** Train the network from scratch
    - The most common approach for neural networks
2. **Transfer Learning:** First pre-train on a generic task, then fine-tune for text-to-SQL
    - The Computer Vision and NLP domains have proven its power
    - Has seen widespread use with the introduction of Transformer-based PLMs
3. **Additional Objectives:** Train for additional sub-tasks simultaneously with text-to-SQL
    - Training for additional tasks, related to the main problem, can boost performance
4. **Pre-train Specific Parts:** Maybe some components of the network can benefit by independent pre-training
    - GP proposes to pre-train the decoder, in order to better learn the output's grammar



Taxonomy Overview of a Deep Learning Text-to-SQL system

Output Refinement: Execution-Guided Decoding

- Sketch-based approaches greatly **reduce** the possibility of errors
- There are still a few possibilities
    - **Aggregation function mismatch** (e.g. AVG on string type)
    - **Condition type mismatch** (e.g. comparing a float type column with a string type value)

- Execution guided decoding helps the system **avoid** making such choices at **prediction time**
- By executing **partially complete** predicted SQL queries, the system can reject choices that create **execution errors** or **yield empty results**

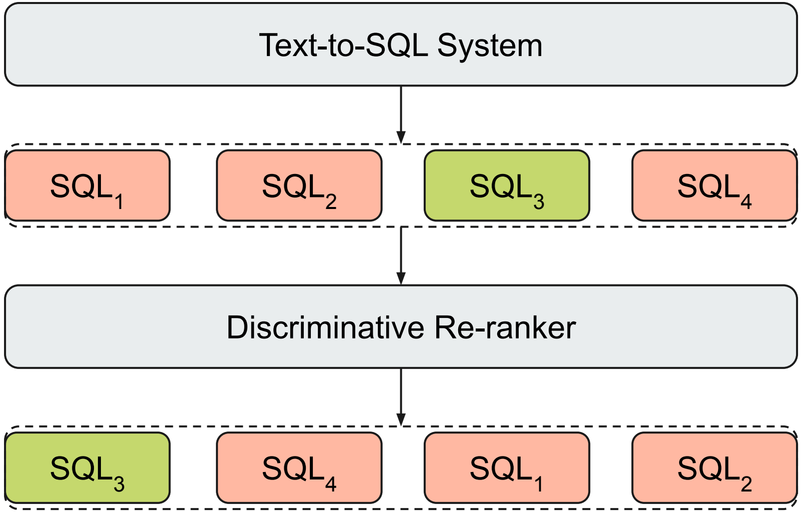
Output Refinement: Constrained Decoding

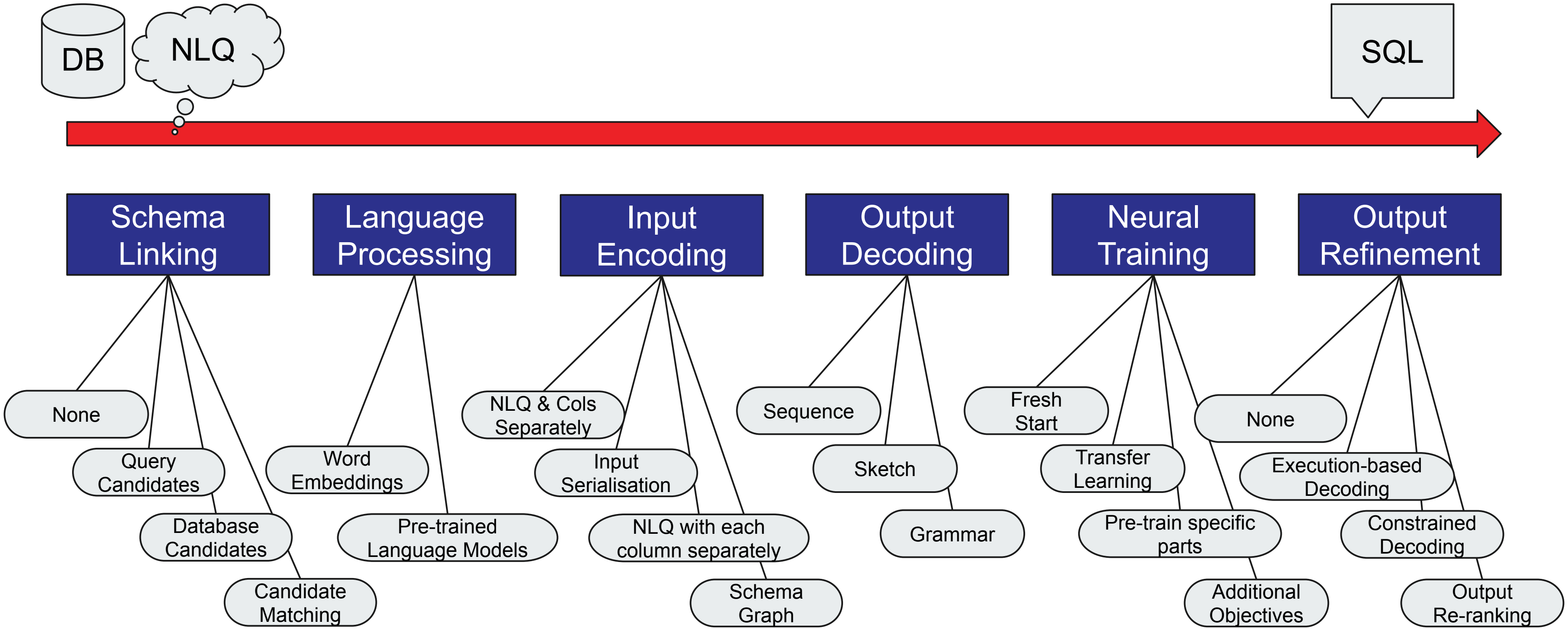
- Models with sequence-based decoders are becoming all the more **powerful** (e.g., T5)
- However, their main drawback is their proneness to **syntactic and grammatical errors**
- Constrained decoding works to **prevent** sequence-based models from producing **erroneous queries**

- PICARD proposes a novel method for incrementally parsing and constraining auto-regressive decoders
  - For each token prediction, PICARD examines the top-*k* most probable tokens
  - If any of the *k* tokens would result in a **grammatical error**, it is discarded
  - If any of the *k* tokens contain an **attribute that is not present in the DB**, it is discarded

Output Refinement: Discriminative Re-ranking

- The nature of neural networks allows us to **extract multiple predictions** for the same NLQ
- Maybe the highest-ranked by the network is not always the correct
- Global-GNN proposes an additional network to **re-rank the *k* highest-ranked** predictions





Taxonomy Overview of a Deep Learning Text-to-SQL system

Key Text-to-SQL Systems

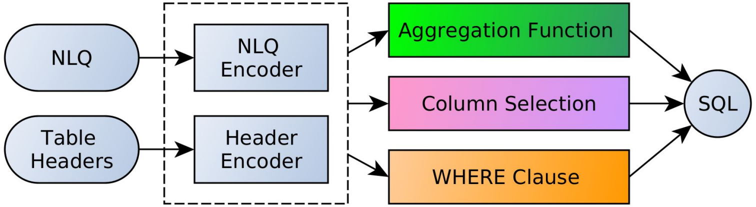
Text-to-SQL Systems

Taking a closer look on key text-to-SQL systems

1. Seq2SQL
2. SQLNet
3. HydraNet
4. SQLova
5. IRNet
6. RAT-SQL
7. T5-3B + PICARD

Seq2SQL

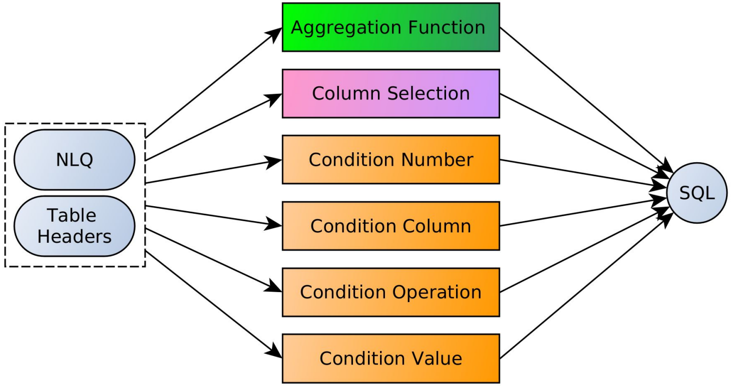
- GloVe Embeddings
- Common LSTM encoders **for all networks**
- Separate networks predict **different parts** of the SQL query
- Trained using **reinforcement learning**



SELECT MAX (budget) WHERE year = 2021

Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
None	Word Embeddings	Separately	Sequence-based	Fresh Start	None

SQLNet



- Completely **sketch-based**
- Each component has its own pair of LSTM encoders
- Introduces **Column Attention**
  - A neural module in each network that tries to emphasize words in the NLQ that might be connected to the table's headers
- **Without** Reinforcement Learning

SELECT *<AGG>* *<COLUMN>*

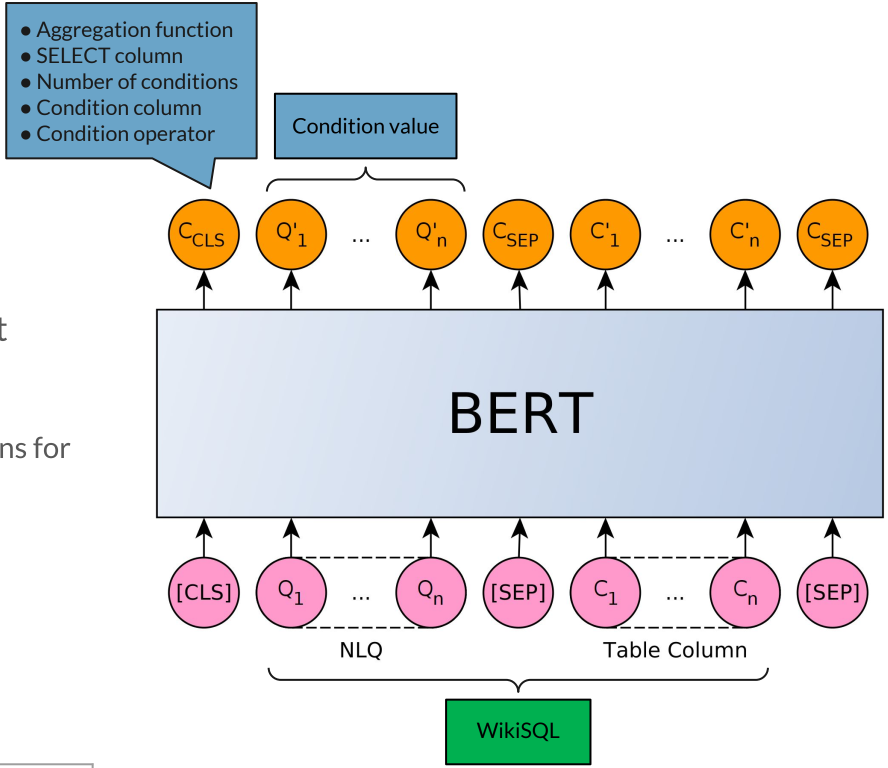
(**WHERE** *<COLUMN>* *<OP>* *<VALUE>*

(**AND** *<COLUMN>* *<OP>* *<VALUE>*) *) ?

Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
None	Word Embeddings	Separately	Sketch-based	Fresh Start	None

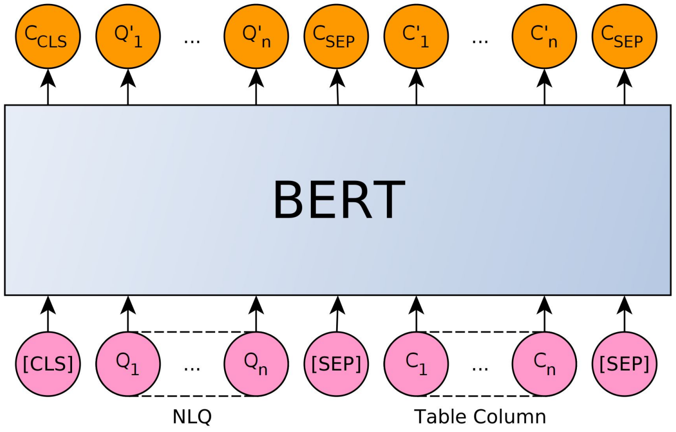
HydraNet

- Works with the same **sketch** as SQLNet
- Almost completely relies on **BERT**
  - Simple linear networks make predictions for the sketch's slots using BERT's output
- Each column is processed **separately**



Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
None	Encoder-only PLM	Each column separately	Sketch-based	Transfer Learning	None

HydraNet



Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
None	Encoder-only PLM	Each column separately	Sketch-based	Transfer Learning	None

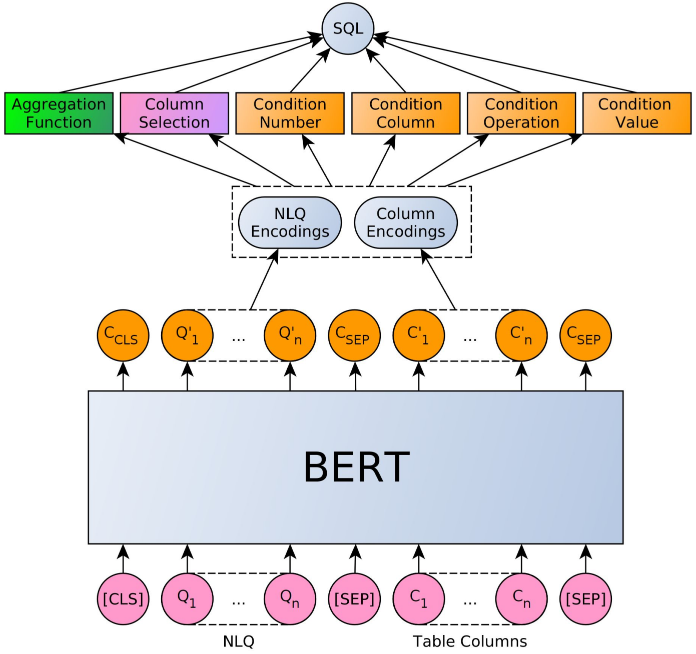
$$P(y_j = \text{start}|c_i, Q) = \text{softmax}(W_{start} \cdot Q'_j)$$

$$P(c_i \in S_Q|Q) = \text{sigmoid}(W_{sc} \cdot C_{CLS})$$

- **For each column** of the table, construct the input for BERT containing the *column_type*, *table_name* and *column_name*
- Classification tasks:
  - Predict if column *i* is in the **SELECT clause**
  - Predict an **aggregation function** for column *i*
  - Predict if column *i* is in the **WHERE clause**
  - Predict a **WHERE clause operator** for column *i*
- Predict the **condition value** for column *i*:
  - For each NLQ token *j* predict if: (a) it is the **start** of the value, (b) if it is the **end** of the value

SQLova

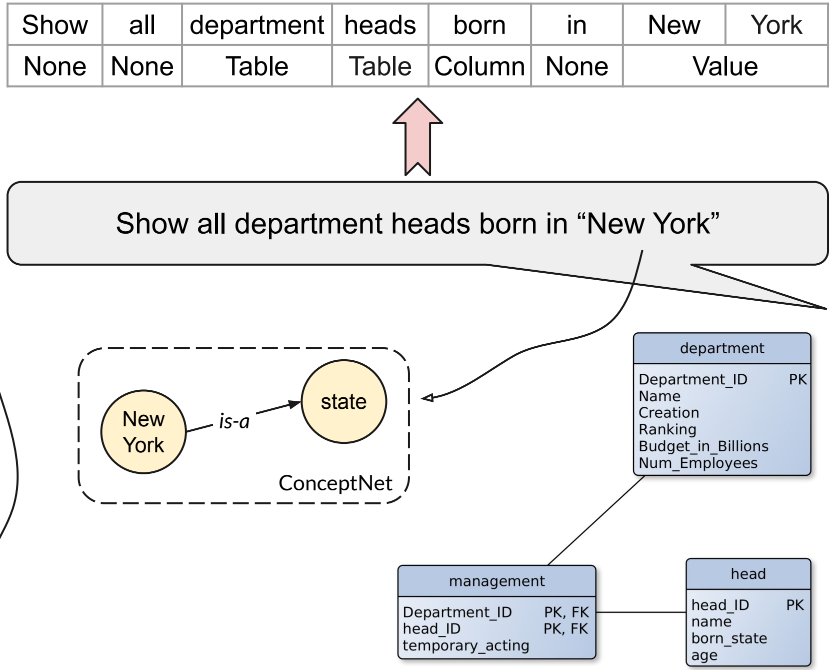
- Same **sketch** as SQLNet
- **Concatenates table columns to NLQ** for simultaneous encoding
- Uses a much **more complex network** after taking the BERT outputs
  - Almost identical to SQLNet
- Achieves **lower accuracy** on WikiSQL than HydraNet



Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
None	Encoder-only PLM	Serialise	Sketch-based	Transfer Learning	None

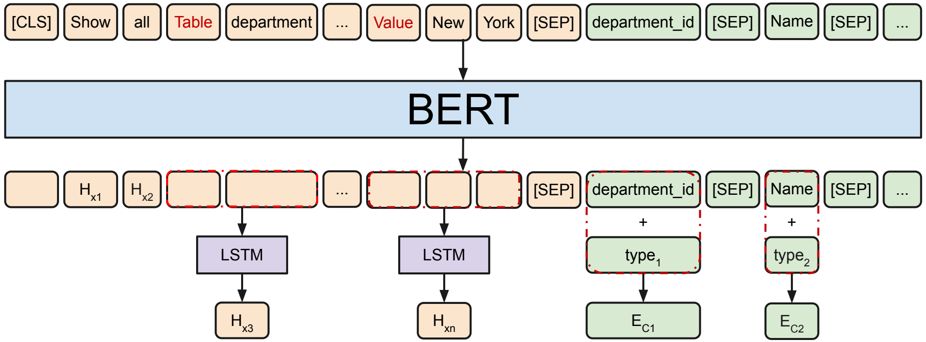
IRNet - Schema Linking

- Considers all n-grams of length 1-6 in the NLQ
- If a n-gram matches a column or a table it is marked as a **complete match** or **partial match** accordingly
- If a n-gram is **inside quotes** it is marked as a **value link**
    - Assumes that DB **values are not accessible**
    - Value links are **searched on ConceptNet** to find the linked column/table
- The NLQ is **split into spans** based on the **types** of discovered links



Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
n-gram match, Knowledge graphs	Encoder-only PLM	Separately (GloVe) or Serialise (BERT)	Grammar-based	Transfer Learning	None

IRNet - Encoding

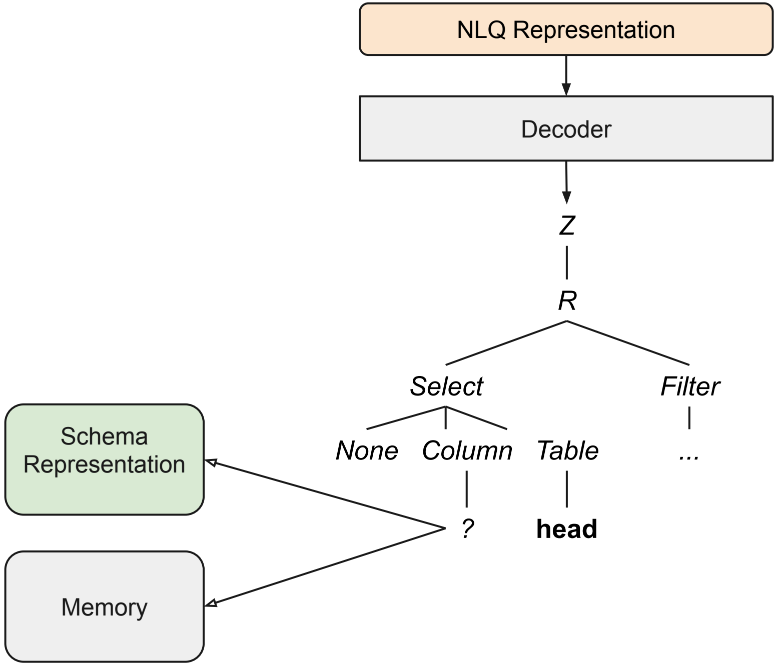


Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
n-gram match, Knowledge graphs	Encoder-only PLM	Separately (GloVe) or Serialise (BERT)	Grammar-based	Transfer Learning	None

- Input can be encoded with **GloVe or BERT**
  - Accuracy with BERT is 8% higher
- **Schema link tokens** are appended to the matched NLQ spans
- Spans with multiple tokens are reduced to a **single token** using LSTM networks
- Column tokens are added to a **type embedding** (int, string, etc.)

IRNet - Decoding

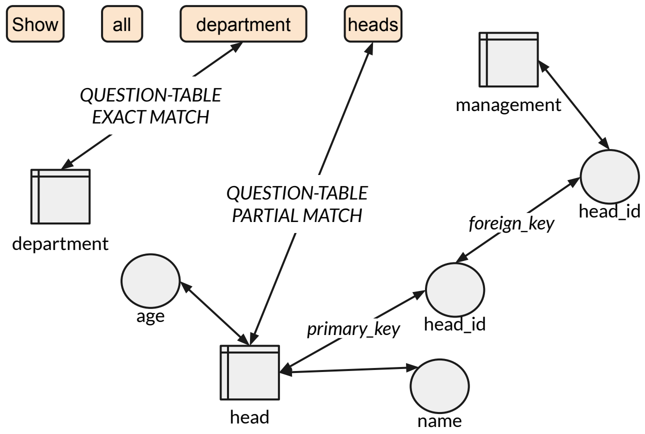
- Generates **SemQL** instead of SQL
- Generate a SemQL query **as an Abstract Syntax Tree** (AST)
    - Uses a LSTM decoder that predicts rules for building the SemQL AST [2
- When generating a **column or table name**, it can make a prediction from:
    - All **schema** elements
    - Elements already used in generated query (**memory**)



Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
n-gram match, Knowledge graphs	Encoder-only PLM	Separately (GloVe) or Serialise (BERT)	Grammar-based	Transfer Learning	None

RAT-SQL - Encoder

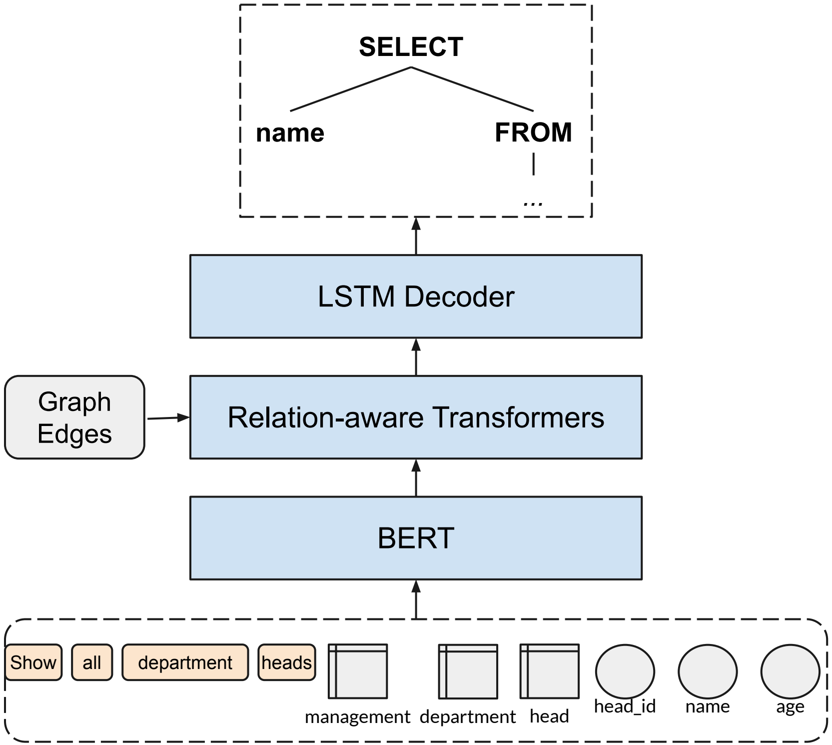
- **Question-contextualized schema graph**
- **Schema** nodes and **NLQ** word nodes
- Edges are **relations** between them from:
  - **Schema relations**
  - **Name-based Linking** (exact or partial n-gram match)
  - **Value-based Linking** (through DB indices or textual search)
- Encoding with GloVe & LSTM or BERT



Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
n-gram match, indices	Encoder-only PLM	Graph encoding	Grammar-based	Transfer Learning	None

RAT-SQL - Decoder

- Specially modified Transformers, for **relation-aware self-attention**, biases the network towards known relations (edges)
- SQL generation as an AST, by predicting a sequence of **decoder actions**
  - Uses a similar **LSTM decoder** to IRNet

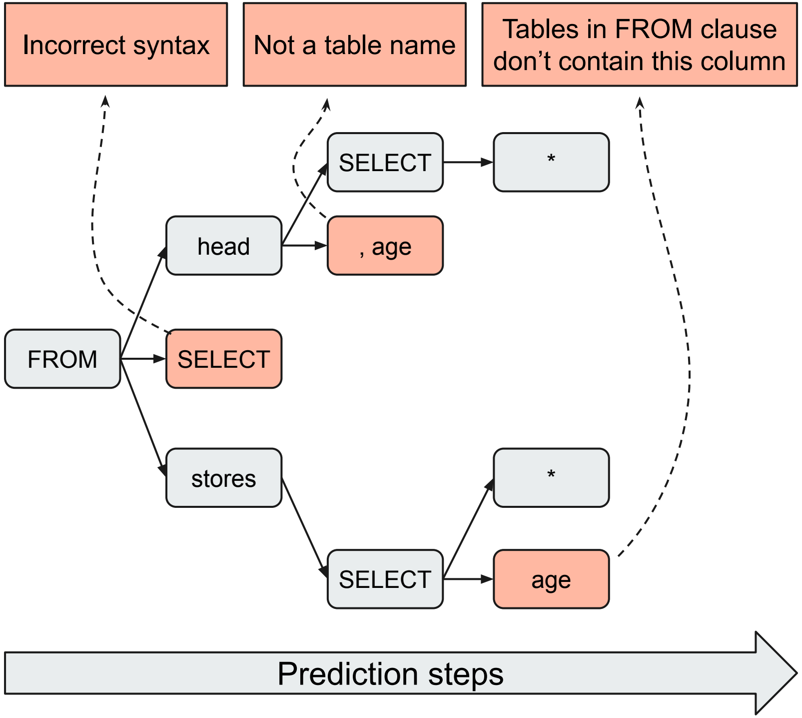


Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
n-gram match, indices	Encoder-only PLM	Graph encoding	Grammar-based	Transfer Learning	None

PICARD

- PICARD is a **constraining technique** for autoregressive decoders of language models
  - Checks for spelling, syntax and grammar errors
  - Checks for availability of used attributes
  - Checks the use of correct aliases
- Tackles the **drawbacks of sequence-based decoders**
- Manages to reach the **top of the Spider leaderboard** in combination with **T5-3B**

Schema Linking	NL Representation	Input Encoding	Output Decoding	Neural Training	Output Refinement
None	Enc-Dec PLM	Serialisation	Sequence-based	Transfer Learning	Constrained Decoding



<table>
<tr><th>System</th><th>NL Representation</th><th>Schema Linking</th><th>Input Encoding</th><th>Decoder Output</th><th>Accuracy</th><th></th></tr>
<tr><td>Seq2SQL</td><td rowspan="2">GloVe</td><td rowspan="4">None</td><td rowspan="2">Separate</td><td>Sequence</td><td>59.4 %</td><td rowspan="4">Execution Accuracy on WikiSQL Test Set</td></tr>
<tr><td>SQLNet</td><td rowspan="3">Sketch-based</td><td>68.0 %</td></tr>
<tr><td>HydraNet</td><td rowspan="4">Encoder-only PLM</td><td>For each column</td><td>92.2 %
(using EG decoding)</td></tr>
<tr><td>SQLova</td><td rowspan="2">Serialise</td><td>89.6 %
(using EG decoding)</td></tr>
<tr><td>IRNet</td><td>n-grams, KG</td><td rowspan="2">Grammar-based</td><td>60.1* %</td><td rowspan="3">Exact Set Match without Values on Spider Test Set</td></tr>
<tr><td>RAT-SQL</td><td>n-grams, indices</td><td>Graph encoding</td><td>70.5* %</td></tr>
<tr><td>T5-3B+PICARD</td><td>Encoder-Decoder PLM</td><td>None</td><td>Serialise</td><td>Sequence</td><td>71.9%</td></tr>
</table>

Text-to-SQL System Overview

**Scores achieved using different language models and improvements*

Challenges and Research Opportunities

Challenges: Benchmarks and Existing Systems

Focus on effectiveness based on the queries translated

They do not:

✗ do not measure **query expressivity** (from a NL or SQL standpoint)

✗ do not care about **execution time** or **model sizes**

✗ do not allow for more than one **correct answers**

Spider Dataset

Model	Dev					Test				
	Easy	Medium	Hard	Extra Hard	All	Easy	Medium	Hard	Extra Hard	All
IRNet					**53.2**	70.1	49.2	39.5	19.1	**46.7**
IRNet (BERT)					**61.9**	77.2	58.7	48.1	25.3	**54.7**
RAT-SQL	80.4	63.9	55.7	40.6	**62.7**	74.8	60.7	53.6	31.5	**57.2**
RAT-SQL + BERT	86.4	73.6	62.1	42.9	**69.7**	83.0	71.3	58.3	38.4	**65.6**
RAT-SQL + BART Encoder					**67.6**	82.6	71.1	58.1	37.0	**65.1**
RAT-SQL + GAP Model					**71.8**	87.2	75.1	63.7	41.2	**69.7**
RAT-SQL + GRAPPA					**73.4**					**69.6**
RAT-SQL + GRAPPA + GP					**72.8**					**69.8**
T5-3B					**71.5**					**70.1**
T5-3B + PICARD					**75.5**					**71.9**
LGESQL + ELECTRA					**75.1**					**72**

THOR Query Benchmark

- 216 keyword-based and 241 natural language queries
- Divided into 17 categories
- Spanning 3 datasets of varying sizes and complexities: IMDB, MAS, YELP

	Category	Keyword	Natural Language
C1	No joins & no metadata	"Brad Pitt"	Find about "Brad Pitt"
C2	Joins & no metadata	"Brad Pitt" "Fight Club"	Did "Brad Pitt" act in "Fight Club"?
C3	No joins & metadata	movie "Star Wars" prod_year	Find the production year of the movie "Star Wars"
C4	Joins & metadata	actor "Brad Pitt" movie	Find the movies of actor "Brad Pitt"
C5	Aggregates	COUNT actor movie "Star Wars"	Find the number of actors of the movie "Star Wars"
C6	GroupBy	COUNT movie GROUPBY prod_year	Find the number of movies per production year
C7	Numeric constraints	movie prod_year=2010	Which movies were produced in 2010
C8	Logical Operations	movie prod_year=2010 or prod_year=2014	Find the movies produced in 2010 or 2014
C9	Nested	MAX COUNT movie GROUPBY prod_year	What is the maximum number of movies produced in
C10	Metadata synonyms	film (= movie)	Return all films (= movie)
C11	Value synonyms	woman (= female) actor	Find all women (= female) actors
C12	Metadata misspellings	actor "Brad Pitt" movei	Find the moveis of actor "Brad Pitt"
C13	Value misspellings	actor "Bred Pett" movie	Find the movies of actor "Bred Pett"
C14	Metadata stemming	actor names	Return all actor names
C15	Value stemming	females	Return all females
C16	Negation	movie not (COUNT actor > 10)	Find the movies that do not have more than 10 actors
C17	Inference logic	top movie	Return the top movie

SQL Challenges

NL Challenges

Challenges: Query Expressivity

Few systems tackle most SQL challenges (to an extent), but NL challenges are even harder

	Category	Keyword	Natural Language
C1	No joins & no metadata	*"Brad Pitt"*	*Find about "Brad Pitt"*
C2	Joins & no metadata	*"Brad Pitt" "Fight Club"*	*Did "Brad Pitt" act in "Fight Club"?*
C3	No joins & metadata	*movie "Star Wars" prod_year*	*Find the production year of the movie "Star Wars"*
C4	Joins & metadata	*actor "Brad Pitt" movie*	*Find the movies of actor "Brad Pitt"*
C5	Aggregates	*COUNT actor movie "Star Wars"*	*Find the number of actors of the movie "Star Wars"*
C6	GroupBy	*COUNT movie GROUPBY prod_year*	*Find the number of movies per production year*
C7	Numeric constraints	*movie prod_year=2010*	*Which movies were produced in 2010*
C8	Logical Operations	*movie prod_year=2010 or prod_year=2014*	*Find the movies produced in 2010 or 2014*
C9	Nested	*MAX COUNT movie GROUPBY prod_year*	*What is the maximum number of movies produced in a year*
C10	Metadata synonyms	*film (= movie)*	*Return all films (= movie)*
C11	Value synonyms	*woman (= female) actor*	*Find all women (= female) actors*
C12	Metadata misspellings	*actor "Brad Pitt" movei*	*Find the moveis of actor "Brad Pitt"*
C13	Value misspellings	*actor "Bred Pett" movie*	*Find the movies of actor "Bred Pett"*
C14	Metadata stemming	*actor names*	*Return all actor names*
C15	Value stemming	*females*	*Return all females*
C16	Negation	*movie not (COUNT actor > 10)*	*Find the movies that do not have more than 10 actors*
C17	Inference logic	*top movie*	*Return the top movie*

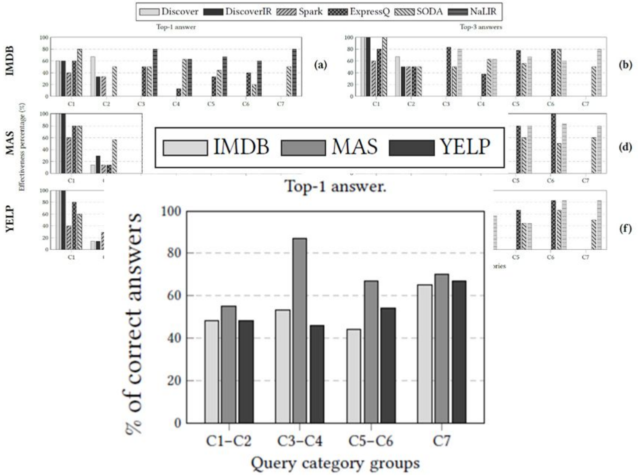
Can we build systems that can answer any type of NL question?

Challenges: Universal solutions

No universal solutions exist

Different data sets present different intricate characteristics

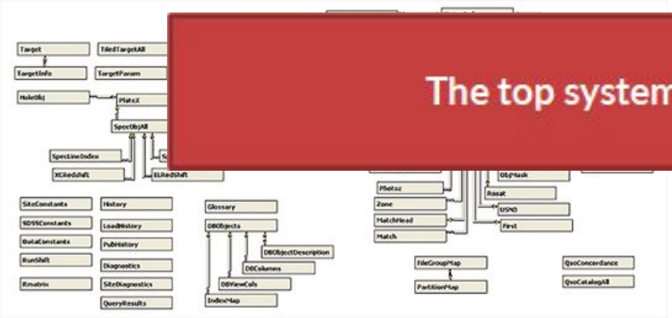
✗ Domain-specific or application-specific solutions: ontologies, knowledge bases



Can we build systems that work well for different datasets?

Challenges: Real-life datasets

- Research & Innovation Policy Making: CORDIS
- Astrophysics: SDSS
- Cancer Biomarker Research



The top systems for SPIDER do not work!

Challenges: Real-life datasets

1. Unknown (and often cryptic) schemas

   e.g., u, g, specobj, photoobj

2. Scaling to very large schemas

   Photo_obj table alone has over 500 attributes

3. Complex systems

4. No training data

Challenges: Deep Learning all the way?

Database-based approaches generate semantically correct SQL queries, NMT approaches promise to be able to generalize to different types of queries and data

✗ Not there yet → low query expressivity

Can we combine the best of both worlds?

- techniques?
- systems?

Challenges: One answer or more?

Deep learning approaches generate one translation for a user query

✗ what if there are more than one way to answer a query

> Show me Italian restaurants

1. "business categorized as restaurant and as Italian”
2. "business categorized as restaurant that serves Italian”

We need to balance diversity and disambiguation

Challenges: The next steps for Data Democratisation

- Even if we solve the text-to-SQL problem, **is our job done?**
- How can the user **validate the predicted SQL** so that it matches the intention of their query?
  - Natural Language explanations of SQL (SQL-to-text)
- What if the user **does not understand the DB** well enough to ask a NLQ?
  - Query recommendation systems
  - Intelligent exploration systems
- What if the user **does not understand the returned data**?
  - Data visualisation
  - Query result explanations

Text-to-SQL systems are just one of the pieces in the data democratisation puzzle

References (1/5)

[1] O. Gkini, T. Belmpas, G. Koutrika, Y. Ioannidis. An In-Depth Benchmarking of Text-to-SQL Systems. ACM SIGMOD 2021.

[2] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. CoRR, September 2017

[3] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. EMNLP 2018.

[4] C. Wang, K. Tatwawadi, M. Brockschmidt, P. Huang, Y. Mao, O. Polozov and R. Singh Robust. 2018. Text-to-SQL Generation with Execution-Guided Decoding.

[5] S. Hochreiter and J. Schmidhuber . 1997. Long Short-term Memory. Neural computation. 9. 1735-80.

[6] T. Mikolov, K. Chen, G. Corrado and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space.

[7] J. Pennington, R. Socher and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. EMNLP 2014.

References (2/5)

[8] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes and J. Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. 2017. Attention Is All You Need. NIPS 2017.

[10] D. Jacob, C. Ming-Wei, L. Kenton and T. Kristina. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 4171–4186.

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

[12] P. Yin, G. Neubig, W. Yih and S. Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.

[13] T. Yu, C. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. Radev, R. Socher and C. Xiong. 2020. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing.

References (3/5)

[14] L .Dong and M. Lapata. 2016. Language to Logical Form with Neural Attention. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

[15] X. Xu, C. Liu and D. Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning.

[16] W. Hwang, J. Yim, S. Park and M. Seo. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization.

[17] Q. Lyu, K. Chakrabarti, S. Hathi, S. Kundu, J. Zhang and Z. Chen. 2020. Hybrid Ranking Network for Text-to-SQL.

[18] T. Shi, K. Tatwawadi, K. Chakrabarti, Y. Mao, O. Polozov, and W. Chen. 2018. IncSQL: Training Incremental Text-to-SQL Parsers with Non-Deterministic Oracles.

[19] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J. Lou, T. Liu, and D. Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.

[20] B. Wang, R. Shin, X. Liu, O. Polozov, M. Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.

References (4/5)

[21] P. Yin and G. Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

[22] X. V. Lin, R. Socher and C. Xiong. Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. Findings of the Association for Computational Linguistics: EMNLP 2020.

[23] B. Hui, X. Shi, R.Geng, B. Li, Y. Li, J. Sun and Xiaodan Zhu. Improving Text-to-SQL with Schema Dependency Learning. 2021

[24] U. Brunner and K. Stockinger. ValueNet: A Neural Text-to-SQL Architecture Incorporating Values. 2020

[25] Mohammed Saeed and Paolo Papotti. Fact-checking Statistical Claims with Relational Datasets. IEEE Data Engineering, 2021

[26] S.Jo, I. Trummer, W. Yu, X. Wang, C. Yu, D. Liu, and N. Mehta. Verifying text summaries of relational data sets. SIGMOD '19

[27] G. Karagiannis, M. Saeed, P. Papotti, and I. Trummer. Scrutinizer: Fact checking statistical claims. Proc. VLDB Endow., 2020

References (5/5)

[28] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics

[29] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research

[30] T. Scholak, N. Schucher, D. Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing

[31] K. Xuan, Y. Wang, Y. Wang, Z. Wen, Y. Dong. 2021. SeaD: End-to-end Text-to-SQL Generation with Schema-aware Denoising. Arxiv pre-print

[32] L. Zhao, H. Cao, Y. Zhao. 2021. GP: Context-free Grammar Pre-training for Text-to-SQL Parsers. CoRR

[33] B. Bogin, M. Gardner, J. Berant. 2019. Global Reasoning over Database Structures for Text-to-SQL Parsing. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing

Thank you! Questions?