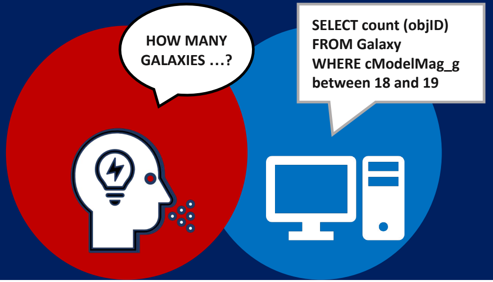
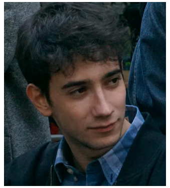
A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems

George Katsogiannis-Meimarakis (katso@athenarc.gr)

Georgia Koutrika (georgia@athenarc.gr)



Presenters



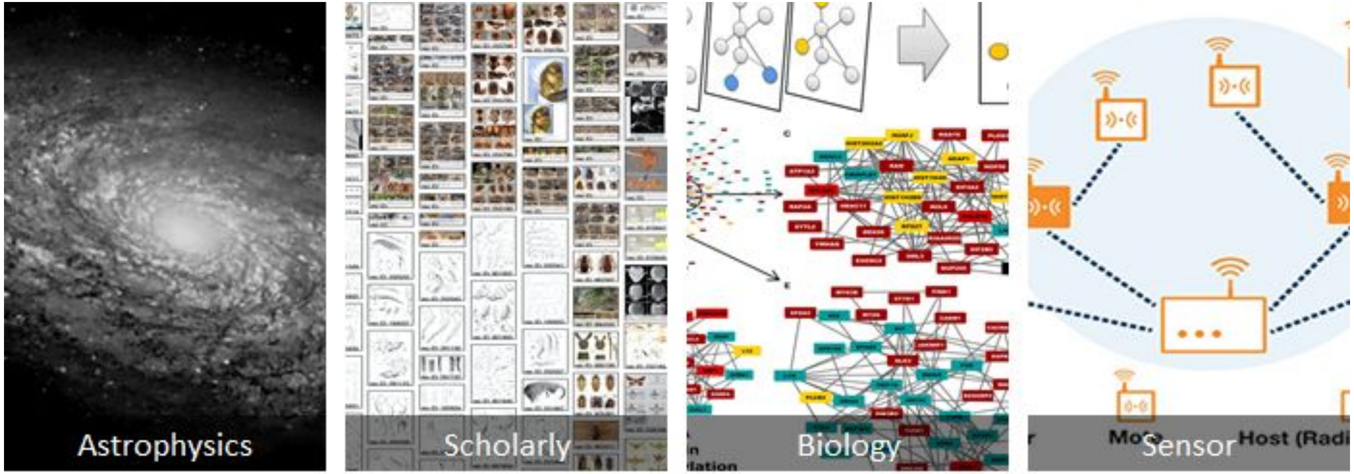
George Katsogiannis

- **Research Assistant** at Athena Research Center, Greece
  - Text-to-SQL
  - Data Exploration
  - 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

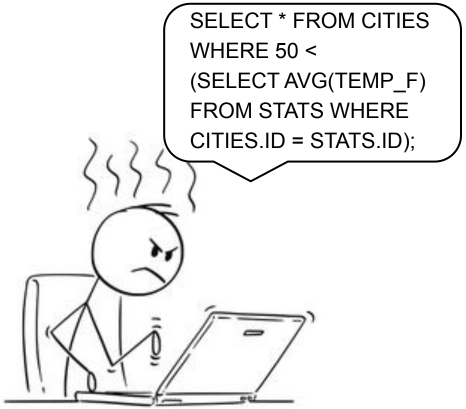
Why Text-to-SQL Systems?



- Many different data sets are generated by users, systems and sensors
- Data repositories can benefit many types of users looking for insights, patterns, information, etc
- Hence, the benefit of data exploration becomes increasingly more prominent.

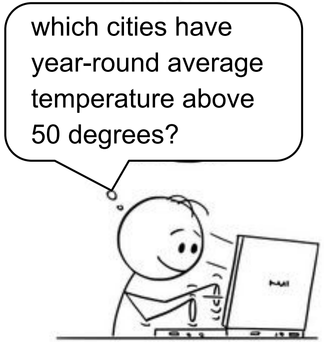
Why Text-to-SQL Systems?

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



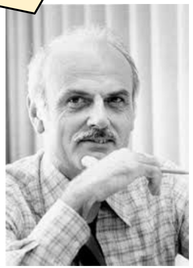
Why Text-to-SQL Systems?

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



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)



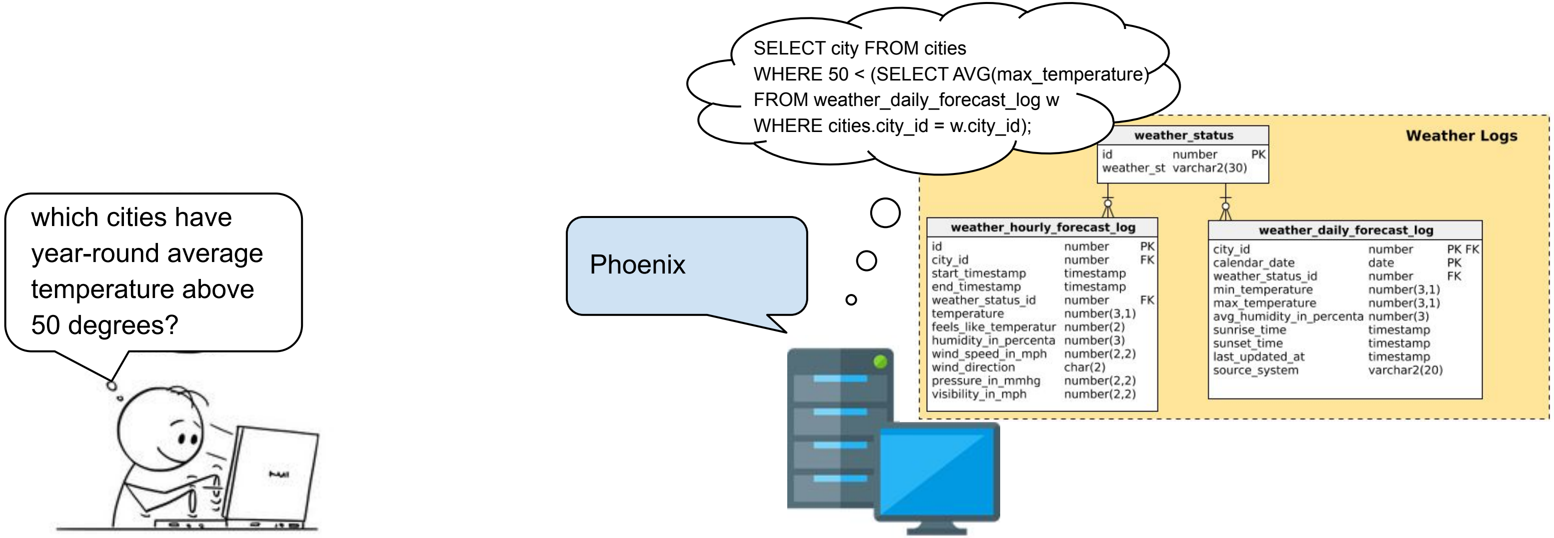
Tutorial Outline

1. The Text-to-SQL Problem - 20’
2. Text-to-SQL Landscape
3. Available Benchmarks
4. Natural Language Representation - 15’
    a. GloVe Embeddings
    b. Wordpiece Embeddings
    c. BERT
    d. Grappa
5. Text-to-SQL Deep Learning Taxonomy - 15’
    a. Schema Linking
    b. Input Encoding
    c. Decoder Output
6. Key Text-to-SQL Systems -25’
    a. Seq2SQL
    b. SQLNet
    c. HydraNet
    d. SQLova
    e. SDSQL
    f. BRIDGE
    g. IRNet
    h. ValueNet
    i. RAT-SQL
7. Challenges & Research Opportunities - 10’

The Text-to-SQL Problem

Text-to-SQL Landscape
Available Benchmarks
Natural Language Representation
Text-to-SQL Deep Learning Taxonomy
Key Text-to-SQL Systems
Challenges & Research Opportunities

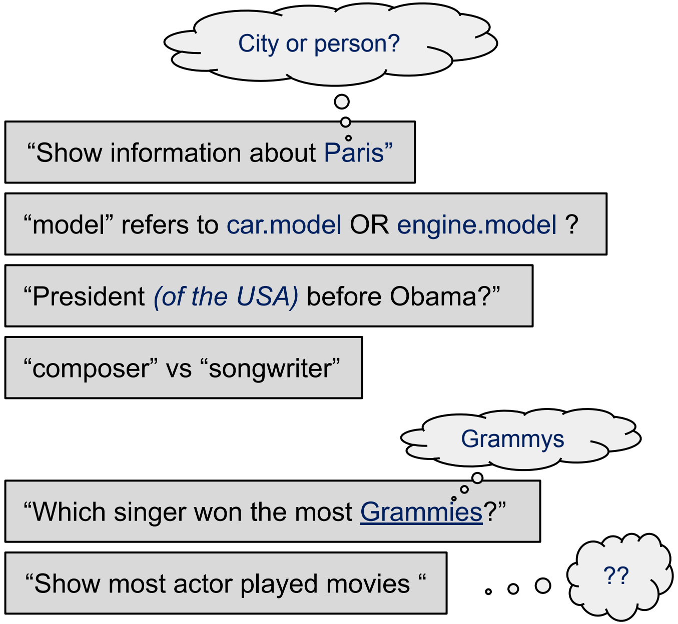
The Text-to-SQL Problem



Challenges

From the NL side

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



Challenges

From the SQL side

- **Complex Syntax:**
  - SQL is a structured language with a strict grammar and limited expressivity

    “Which countries have a GDP higher than the EU average?”

    Sounds simple but needs a complex nested query

- **Database Structure:**
  - The user’s data model may not match the data schema

    “Find directors who released a movie this year”

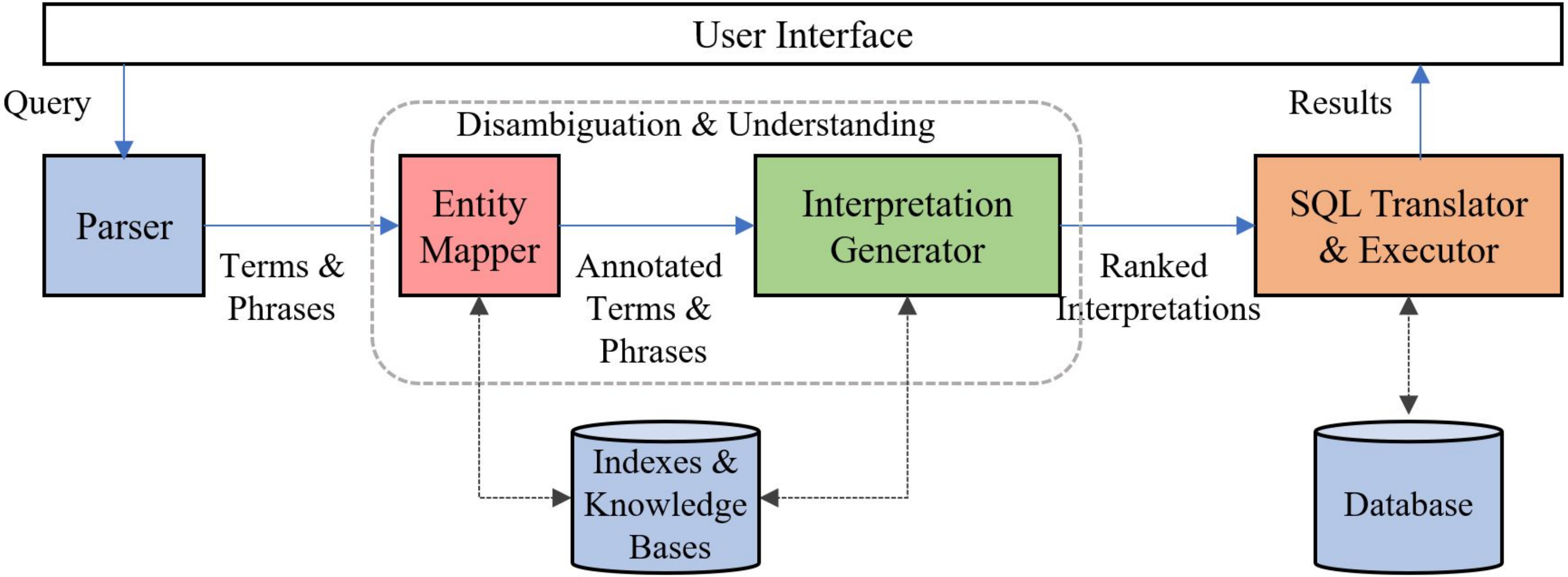
    Simple NLQ that might need 3,4 or 5 JOINs

The Text-to-SQL Problem

Text-to-SQL Landscape

Available Benchmarks
Natural Language Representation
Text-to-SQL Deep Learning Taxonomy
Key Text-to-SQL Systems
Challenges & Research Opportunities

System Workflow



[1] THOR(2021)

Generations of Text-to-SQL Systems

Keyword systems

a search engine-like functionality, where user queries contain just keywords, like “drama movies".

- **Discover** 🔗 [2]
  generates query interpretations as subgraphs (candidate networks) of the database schema graph.
- **DiscoverIR** 🔗 [3]
  information retrieval-style ranking heuristics to enhance the term disambiguation process.
- **Spark** 🔗 [4]
  improved ranking and fast execution methods

Generations of Text-to-SQL Systems

Enhanced Keyword systems

- queries with aggregate functions, GroupBy, comparison operators, and keywords that map to database metadata.
- syntactic constraints on their input to make sure they can parse the user query.
  e.g., "count movies actress "Priyanka Chopra"".

- **ExpressQ** [5]
  specific keywords trigger aggregate functions and GroupBy
- **SODA** [6]
  enriches the system knowledge (i.e. inverted indexes) with additional knowledge sources

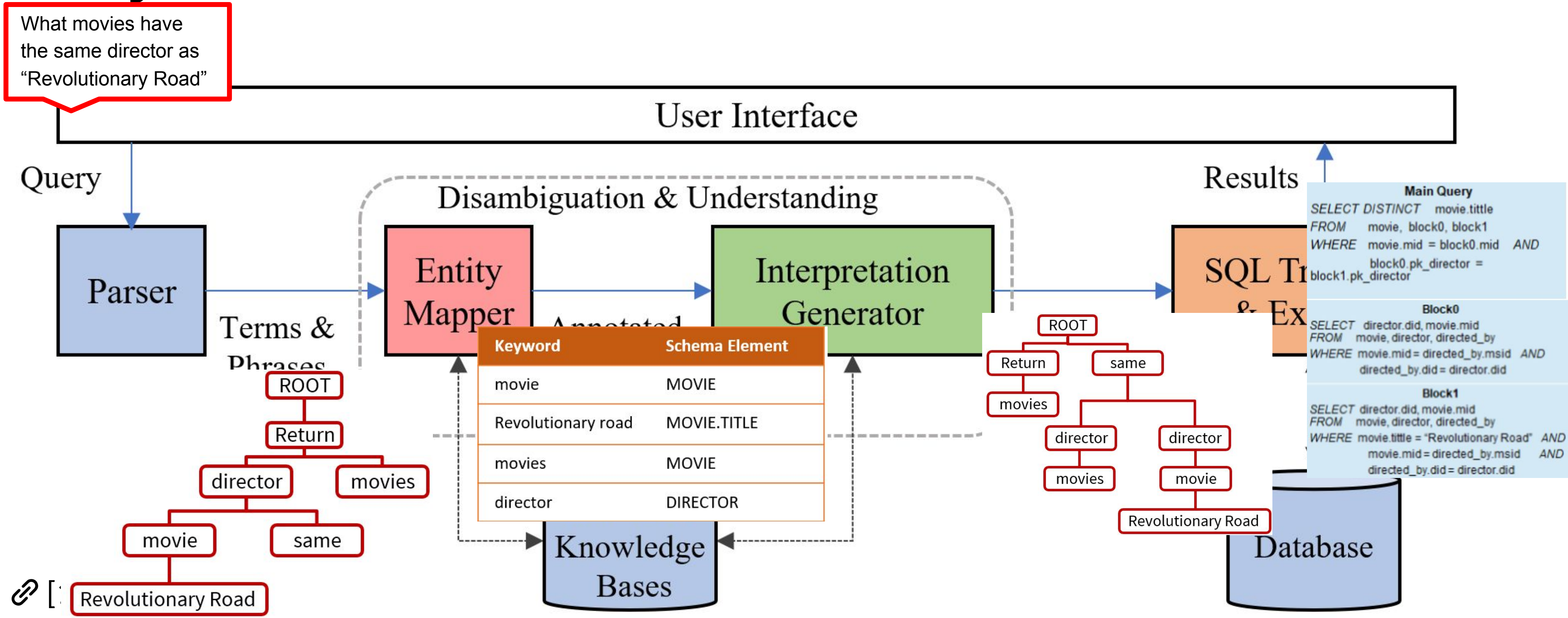
Generations of Text-to-SQL Systems

Natural language systems

- allow queries in natural language,
  “What is the number of movies of “Priyanka Chopra"".

- **NaLIR** 🔗 [7]
  syntactic parser to understand NL.
- **ATHENA** 🔗 [8]
  ontologies and ontology-to-data mappings

System Workflow



Keyword	Schema Element
movie	MOVIE
Revolutionary road	MOVIE.TITLE
movies	MOVIE
director	DIRECTOR

[

The dawn of Deep Learning Text-to-SQL

2016
- Language to Logical Form with Neural Attention

2017
- The Transformer
- Seq2SQL + WikiSQL
- SQLNet

2018
- TypeSQL
- Coarse-to-Fine
- Spider
- IncSQL
- BERT
- SyntaxSQLNet

2019
- SQLova
- IRNet
- X-SQL
- RAT-SQL

2020
- RYANSQL
- TaBERT
- HydraNet
- GraPPa
- BRIDGE
- SmBoP
- IE-SQL
- ValueNet

2021
- SDSQL
- What next?

- Datasets
- Word Representation

A timeline of NL2SQL systems using Deep Learning

Text-to-SQL as Neural Machine Translation

Neural machine translation (NMT) approaches
map the text-to-SQL problem to a **language translation problem**
and they train over a large body of <NL, SQL> pairs.

The Text-to-SQL Problem
Text-to-SQL Landscape

Available Benchmarks

Natural Language Representation
Text-to-SQL Deep Learning Taxonomy
Key Text-to-SQL Systems
Challenges & Research Opportunities

Evaluation of Text-to-SQL Systems

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)

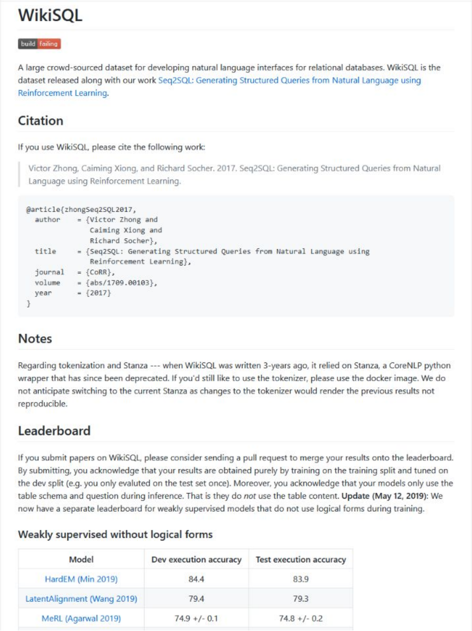
✗ **No standard, small query sets**

– Different test queries, often not available to reproduce the experiments.

✗ **Incomparable effectiveness evaluations**

– none, user study, manual evaluation, comparison to gold standard queries

Two new benchmarks



WikiSQL

build failing

A large crowd-sourced dataset for developing natural language interfaces for relational databases. WikiSQL is the dataset released along with our work Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning.

Citation

If you use WikiSQL, please cite the following work:

> Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning.

```
@article{zhongSeq2SQL2017,
  author    = {Victor Zhong and
               Caiming Xiong and
               Richard Socher},
  title     = {Seq2SQL: Generating Structured Queries from Natural Language using
               Reinforcement Learning},
  journal   = {CoRR},
  volume    = {abs/1709.00103},
  year      = {2017}
}
```

Notes

Regarding tokenization and Stanza --- when WikiSQL was written 3-years ago, it relied on Stanza, a CoreNLP python wrapper that has since been deprecated. If you'd still like to use the tokenizer, please use the docker image. We do not anticipate switching to the current Stanza as changes to the tokenizer would render the previous results not reproducible.

Leaderboard

If you submit papers on WikiSQL, please consider sending a pull request to merge your results onto the leaderboard. By submitting, you acknowledge that your results are obtained purely by training on the training split and tuned on the dev split (e.g. you only evaluted on the test set once). Moreover, you acknowledge that your models only use the table schema and question during inference. That is they do *not* use the table content. **Update (May 12, 2019):** We now have a separate leaderboard for weakly supervised models that do not use logical forms during training.

Weakly supervised without logical forms

Model	Dev execution accuracy	Test execution accuracy
HardEM (Min 2019)	84.4	83.9
LatentAlignment (Wang 2019)	79.4	79.3
MeRL (Agarwal 2019)	74.9 +/- 0.1	74.8 +/- 0.2

Spider 1.0

Yale Semantic Parsing and Text-to-SQL Challenge

What is Spider?

Spider is a large-scale *complex and cross-domain* semantic parsing and text-to-SQL dataset annotated by 11 Yale students. The goal of the Spider challenge is to develop natural language interfaces to cross-domain databases. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. In Spider 1.0, different complex SQL queries and databases appear in train and test sets. To do well on it, systems must *generalize well to not only new SQL queries but also new database schemas.*

Why we call it "Spider"? It is because our dataset is complex and cross-domain like a spider crawling across mutiple complex(with many foreign keys) nests(databases).

Spider Paper (EMNLP'18)

Spider Post

Related challenges: multi-turn ***SParC*** and conversational ***CoSQL*** text-to-SQL tasks.

SParC Challenge (ACL'19)

CoSQL Challenge (EMNLP'19)

News

- *03/11/2021* Please check out a nice work from Google Research (including new Spider splits) for studying compositional generalization in semantic parsing!
- *11/15/2020* We will use Test Suite Accuracy as our official evaluation metric for Spider, SParC, and CoSQL. Please find the evaluation code from here. Also, Notice that Test results after May 02, 2020 are reported on the new release (collected some annotation errors).
- *08/03/2020* Corrected "column_name" and "column_name_original" mismatches in 2 dbs ("scholar" and "formula_1") in tables.json, and reparsed SQL queries (this only affects some models (e.g. RATSQL) which use our parsed SQL as the SQL input). Please download the Spider dataset from this page again.
- *06/07/2020* We corrected some annotation errors and

Leaderboard - Execution with Values

Our current models do not predict any value in SQL conditions so that we do not provide execution accuracies. However, we encourage you to provide it in the future submissions. For value prediction, your model should be able to 1) copy from the question inputs, 2) retrieve from the database content (database content is available), or 3) generate numbers (e.g. 3 in "LIMIT 3"). *Notice:* Test results after May 02, 2020 are reported on the new release (collected some annotation errors).

Rank	Model	Test
1 Mar 10, 2021	SmBoP + GraPPa (DB content used) *Tel-Aviv University & Allen Institute for AI* (Rubin and Berant, NAACL'21) code	**71.1**
2 Nov 24, 2020	BRIDGE v2 + BERT(ensemble) (DB content used) *Salesforce Research* (Lin et al., EMNLP-Findings '20) code	68.3
2 Jan 16, 2021	COMBINE (DB content used) *Novelis.io Research* (Youssef et al.,'21)	68.2
3 Nov 24, 2020	BRIDGE v2 + BERT (DB content used) *Salesforce Research* (Lin et al., EMNLP-Findings '20) code	64.3
4 May 30, 2020	AuxNet + BART (DB content used) *Anonymous*	62.6
5 May 30, 2020	BRIDGE + BERT (DB content used) *Salesforce Research* (Lin et al., EMNLP-Findings '20) code	59.9
6 May 20, 2020	GAZP + BERT (DB content used) *University of Washington & Facebook AI Research* (Zhong et al., EMNLP '20)	53.5

Leaderboard - Exact Set Match without Values

For exact matching evaluation, instead of simply conducting string comparison between the predicted and gold SQL queries, we decompose each SQL into several clauses, and conduct set comparison in each SQL clause. Please refer to the paper and the Github page for more details. *Notice:* Test results after May 02, 2020 are reported on the new release (collected

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%

[9] Seq2SQL (2017)

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: Easy, Medium, Hard, Extra Hard
  - SQL elements such as JOIN, GROUP BY, UNION

- Better quality and complexity than WikiSQL

[10] Spider (2018)

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")
```

Natural Language Representation

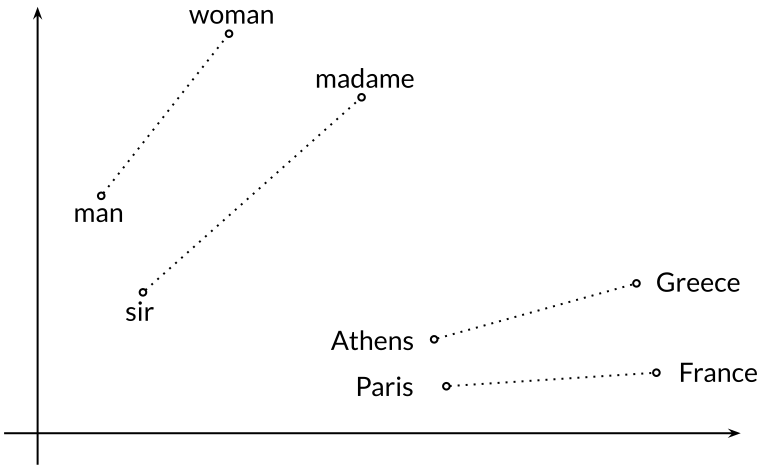
Natural Language Representation

How can we give natural language to a neural network?

- LSTM Neural Networks (1995) 🔗 [12]
- Word Embeddings
  - One-hot Embeddings
  - Word2Vec (2013) 🔗 [13]
  - GloVe (2014) 🔗 [14]
  - WordPiece Embeddings (2017) 🔗 [15]
- The Transformer (2017) 🔗 [16]
- The rise of language models
  - BERT (2018) 🔗 [17]
  - RoBERTa (2019) 🔗 [18]
  - TaBERT (2020) 🔗 [20]
  - GraPPa (2020) 🔗 [20]

GloVe Embeddings

- Create **meaningful vector representations**
- **Unsupervised learning** based on word **co-occurrence** in the training corpus
- Useful **linear substructures** for word relations
- Easy to find **semantical near neighbours**
- Pre-trained vectors created from large corpuses are **available for download**



NearestNeighbours(**frog**) = [frogs, toad, litoria, leptodactylidae, rana, lizard, eleutherodactylus]

[14] GloVe (2014)

The Wordpiece Model

- Approaches like GloVe, Word2Vec, etc. operate with a **fixed word vocabulary**
- The **vocabulary size** is limited by the system's memory
- Inevitably there will be words that are **out-of-vocabulary** (OOV)
- To avoid this, we can use embeddings based on **sub-word units**

[15] WordPiece (2017)

The algorithm:

- Uses a **training corpus** and a number of desired tokens (vocabulary size)
- The initial vocabulary contains all **unique characters**
- More tokens containing multiple characters are added to the vocabulary
- The goal is to **minimize** the number of tokens needed to **segment** the training corpus, subject to the vocabulary size

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

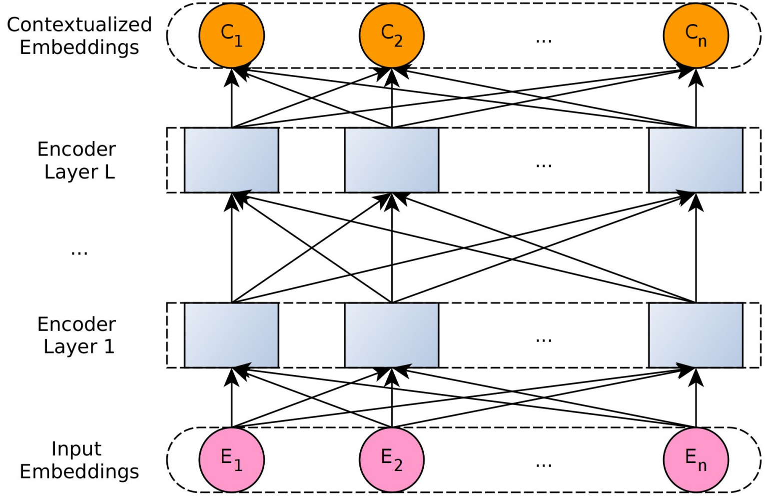
Using sub-words, we **eliminate** the possibility for out-of-vocabulary words, as long as all **characters** were also present during the creation of the embeddings

BERT

- 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
    - Semantic Similarity (STS-B: 86.5 %)
    - Linguistic Acceptability (CoLA: 60.5%)
    - Natural Language Inference (QNLI: 92.7%)

[17] BERT (2018)

BERT: Architecture



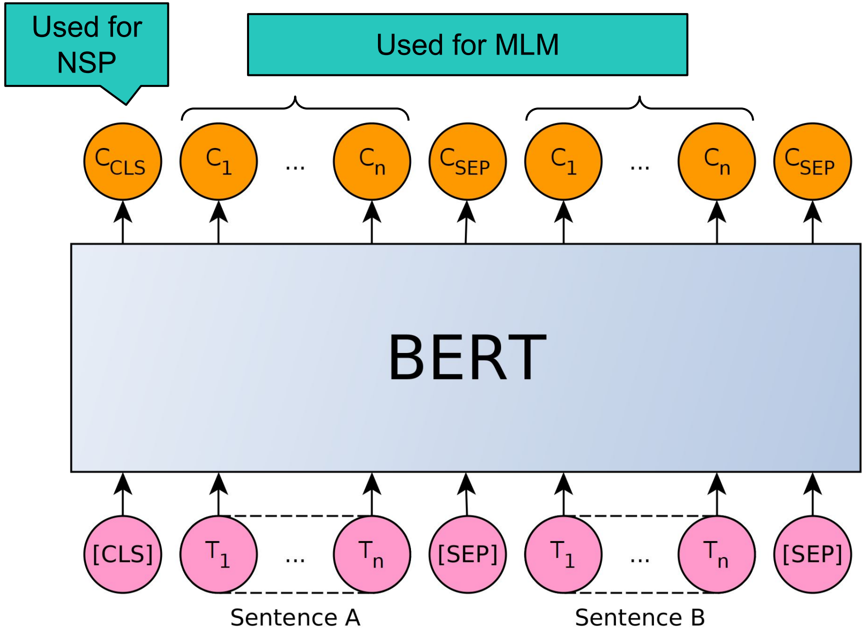
- **Output:** A sequence of tokens of equal length to the input
- Uses many stacks of bidirectional **Transformer** encoder layers
- **Input:** A sequence of token embeddings
    - Uses Wordpiece embeddings

BERT: Pre-training

- 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**)

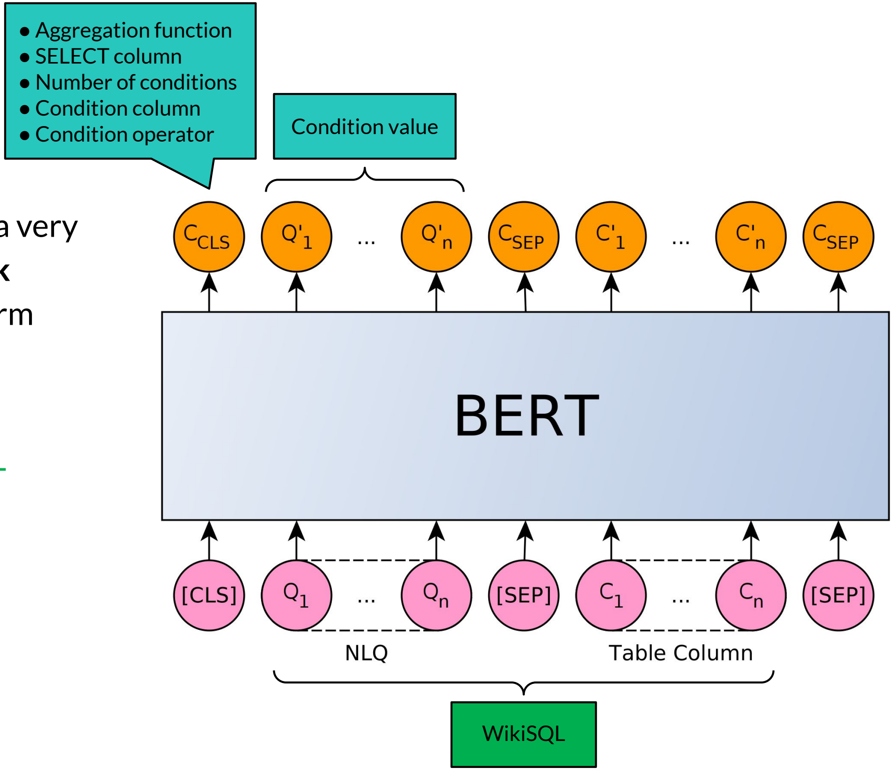
Input = **[CLS]** the man went to **[MASK]** store **[SEP]**
he bought a gallon **[MASK]** milk **[SEP]**

Labels = **MLM$_1$:** the, **MLM$_2$:** of, **NSP:** IsNext



BERT: 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



GraPPa

Similar to BERT, but larger and with better hyperparameters

- 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)

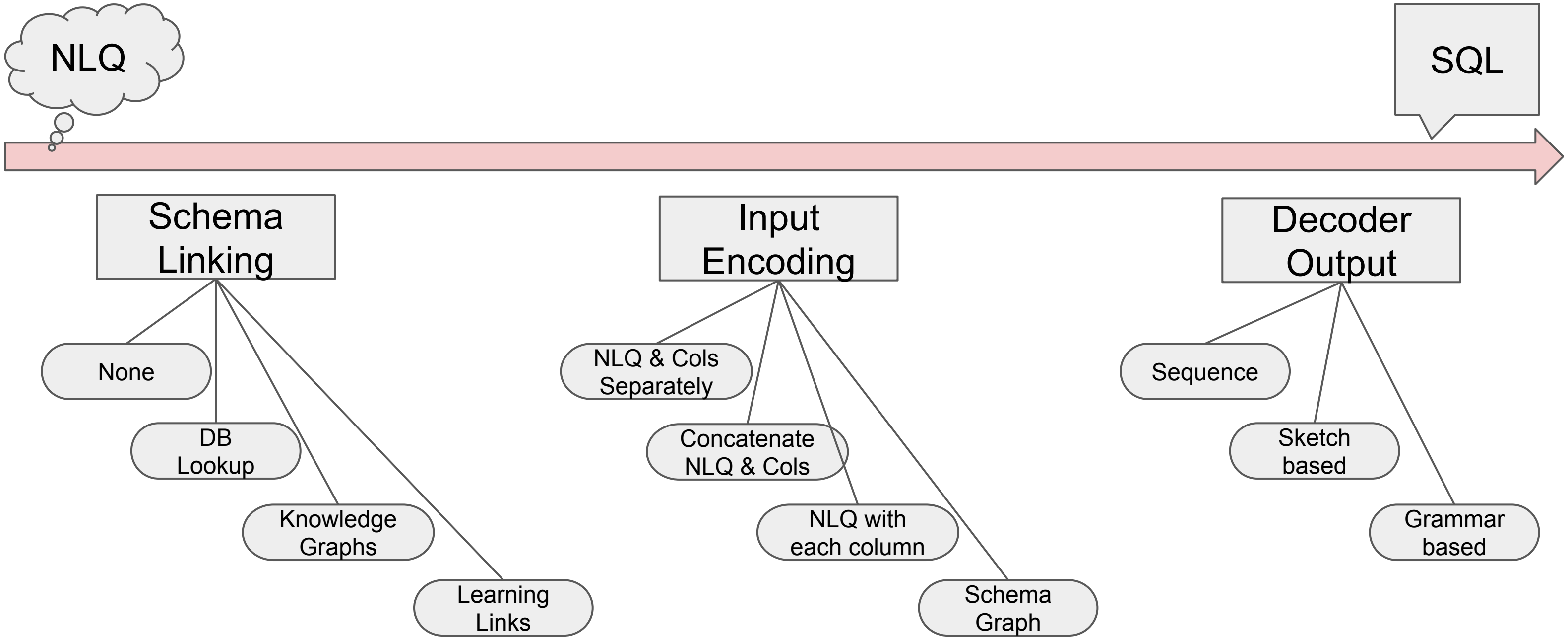
[20] GraPPa (2020)

The Text-to-SQL Problem
Text-to-SQL Landscape
Available Benchmarks
Natural Language Representation

Text-to-SQL Deep Learning Taxonomy

Key Text-to-SQL Systems
Challenges & Research Opportunities

A brief taxonomy



Schema Linking

- Can we discover schema links to help our network?
  - Table links
  - Column links
  - Value links
- Some links are **useful**, some are **not**
- Plethora of techniques
  - Database Lookup
  - n-grams for partial matching
  - Knowledge Graphs for value matching
  - Using classifiers
- Maybe no linking is better?

Table link: *department*

Value links: *head_ID, Department_ID*

NLQ:

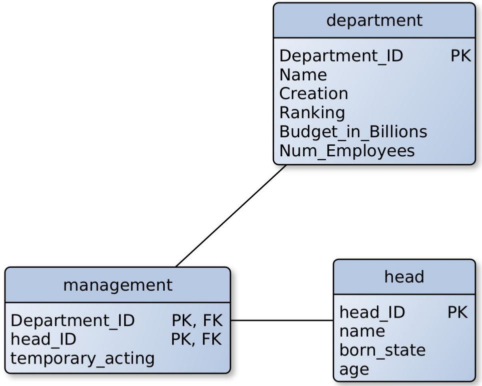
Table link: *head*

Value link: *age*

How many heads of the departments are older than 56 ?

SQL:

SELECT count(*)
FROM head
WHERE age > 56



Schema Linking Techniques

- Entities of the DB and their references in the NLQ might not be **exact matches**
  - **More than one tokens** can be used to refer to an entity

n-gram search
similarity metrics
heuristics

- Searching for **data value links** can be very **cumbersome**
  - The **size** of the data might be prohibiting
  - Data values might be inaccessible due to **privacy issues**

We can use **a reverse index** for faster searches

We can use **knowledge graphs** to get information about words from the NLQ

- Maybe a **neural network** can find schema links

Use a **classifier** for schema linking

NLQ:

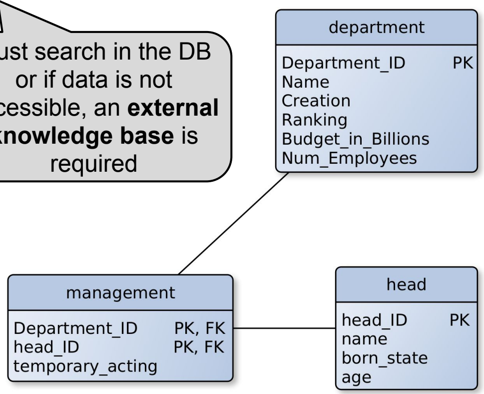
exact match

tri-gram search is needed to find this

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

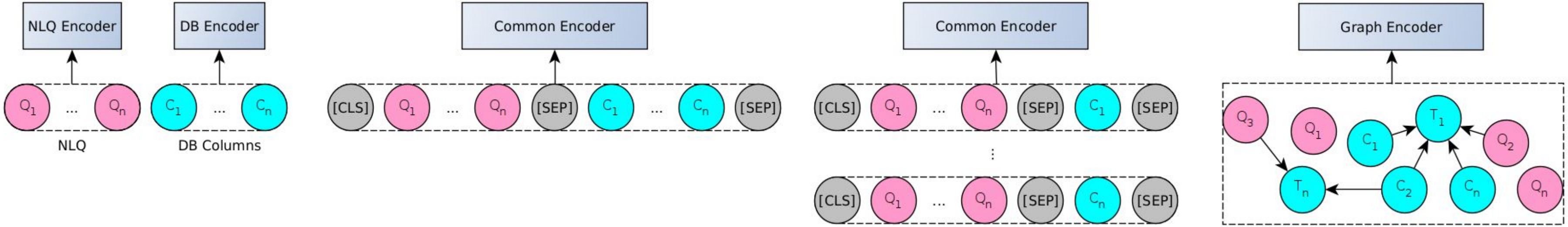
must be matched with **“head”**

must search in the DB or if data is not accessible, an **external knowledge base** is required



Input Encoding

How to structure the input for the neural network?



Encode NLQ and columns/tables separately

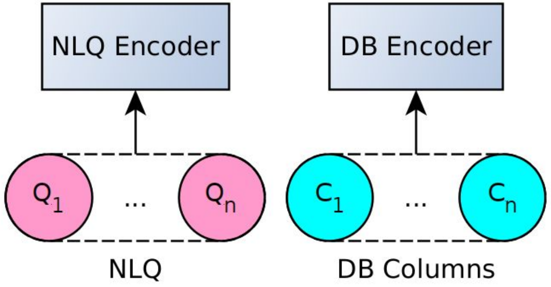
Concatenate NLQ and columns/tables

Encode NLQ with each column separately

Schema Graph encoding

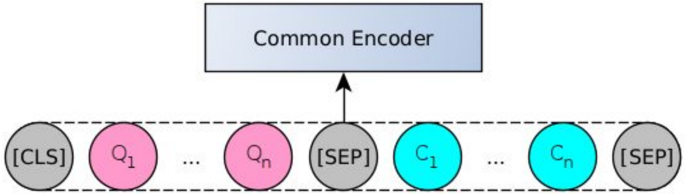
Input Encoding: Separate Encoding

- 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.)

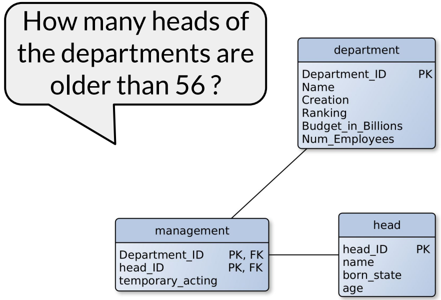


Concatenation of NLQ & DB

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

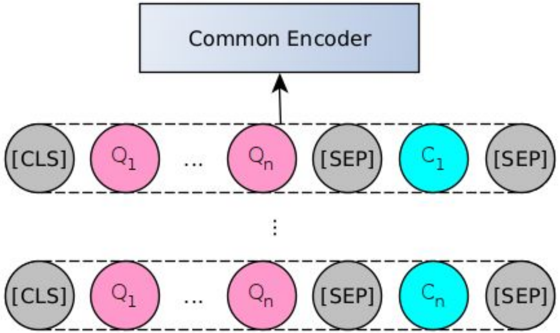


‘How’, ‘many’, ‘heads’, ‘of’, ‘the’, ‘departments’, ‘are’, ‘older’, ‘than’, ‘56, ‘?’, [SEP], ‘department’, [SEP], ‘name’, [SEP], ‘creation’, [SEP], ‘ranking’, [SEP], ‘budget_in_billions’, [SEP], ‘num_employes’, [SEP], ’management’, [SEP], ‘department_id’, [SEP], ‘head_id’, [SEP], ‘temporary_acting’,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



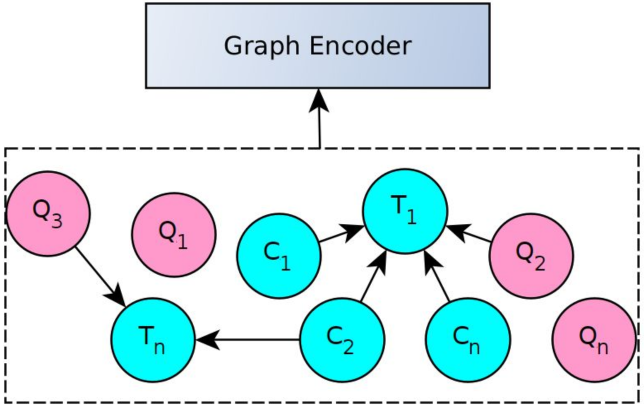
NLQ with Each Column Separately

- 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**



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



Decoder output

Three main categories of text-to-SQL systems based on **decoder output**

- Sequence-based
- Grammar-based
- Sketch-based

Sequence-based

[21] Language to Logical Form with Neural Attention (2016)

[9] Seq2SQL (2017)

- 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
- Rarely used in recent works

Sketch-based Slot-filling

- [22] SQLNet (2017)
- [23] SQLova (2019)
- [24] HydraNet (2020)

- We have a sketch of the query with **missing parts** that need to be filled
- Sketch used by SQLNet:

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

Grammar-based

- [25] IncSQL (2018)
- [26] IRNet (2019)
- [27] 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

A note on 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**

[11] Execution-Guided Decoding (2018)

The Text-to-SQL Problem
Text-to-SQL Landscape
Available Benchmarks
Natural Language Representation
Text-to-SQL Deep Learning Taxonomy

Key Text-to-SQL Systems

Challenges & Research Opportunities

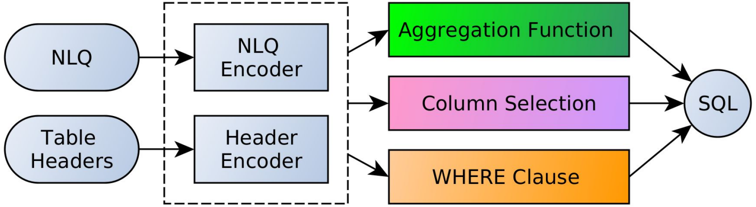
Text-to-SQL Systems

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

1. Seq2SQL
2. SQLNet
3. HydraNet
4. SQLova
5. SDSQL
6. BRIDGE
7. IRNet
8. ValueNet
9. RAT-SQL

Seq2SQL

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

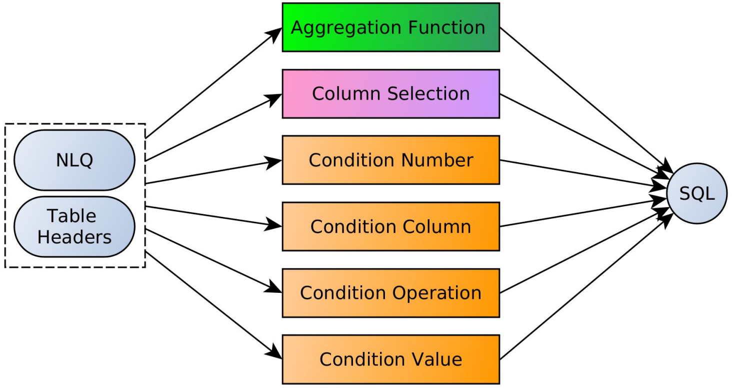


SELECT MAX (budget) WHERE year = 2021

NL Representation	Schema Linking
GloVe embeddings	None
Input Encoding	**Decoder Output**
Separately	Sequence

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>*

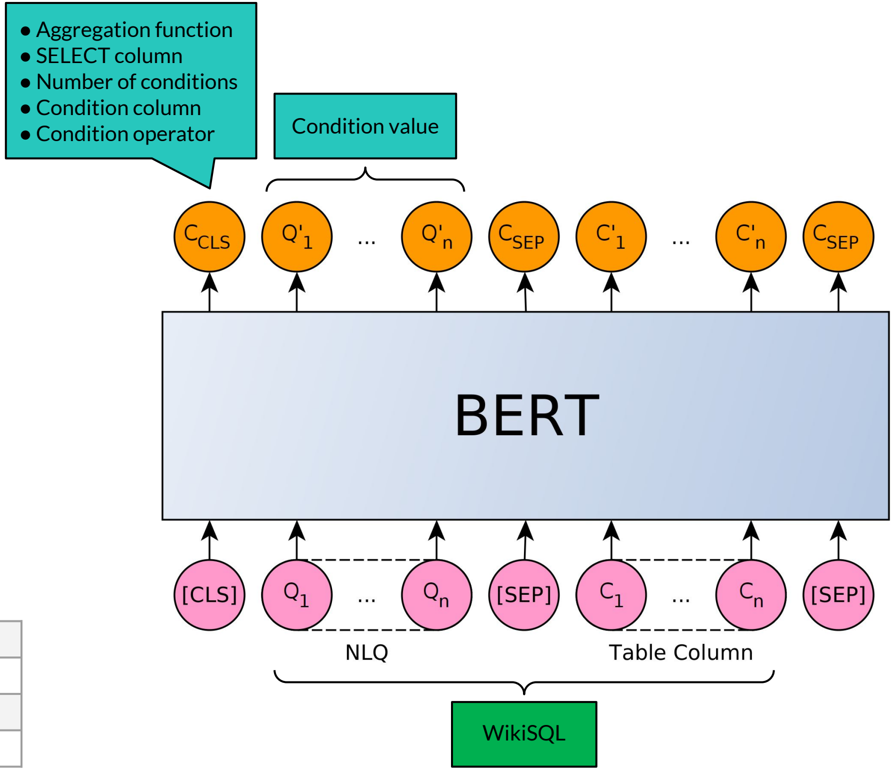
(**WHERE** *<COLUMN>* *<OP>* *<VALUE>*

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

NL Representation	Schema Linking
GloVe embeddings	None
Input Encoding	**Decoder Output**
Separately	Sketch-based

HydraNet

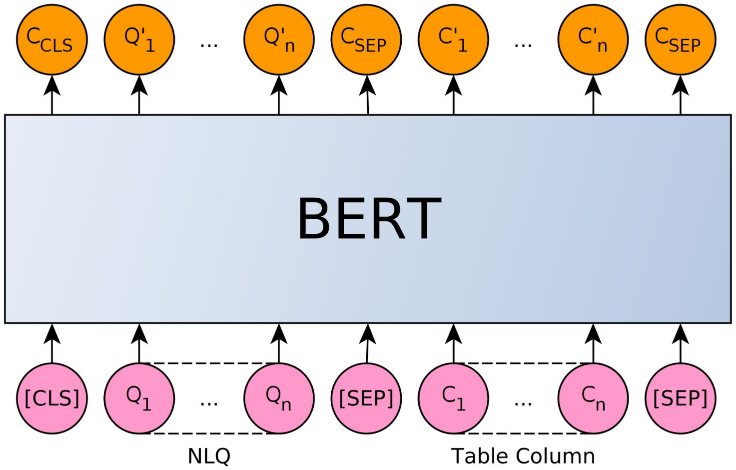
- Works with the same **sketch** as SQLNet
- Almost completely relies on **BERT**
  - Simple linear networks make predictions for the sketch's slots using BERT's output
- Each column is processed **separately**



NL Representation	Schema Linking
BERT	None
Input Encoding	**Decoder Output**
Each column separately	Sketch-based

HydraNet

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



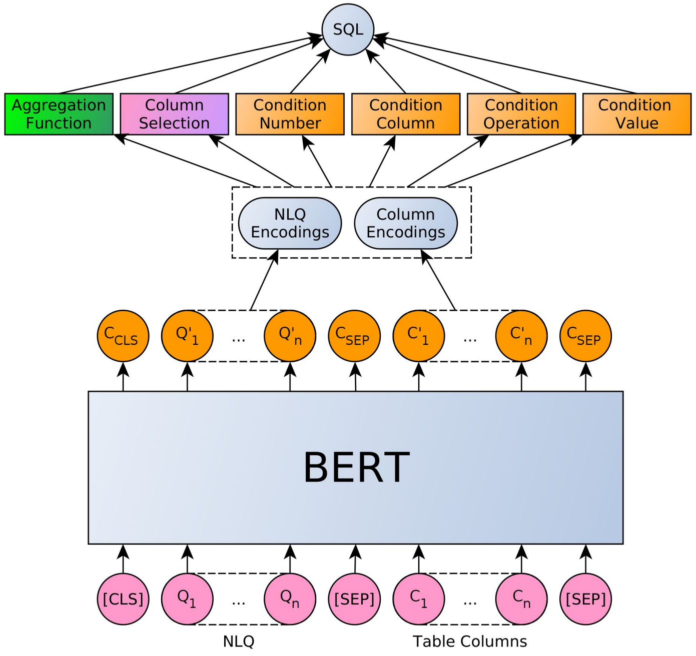
NL Representation	Schema Linking
BERT	None
Input Encoding	**Decoder Output**
Each column separately	Sketch-based

- **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

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

SQLova

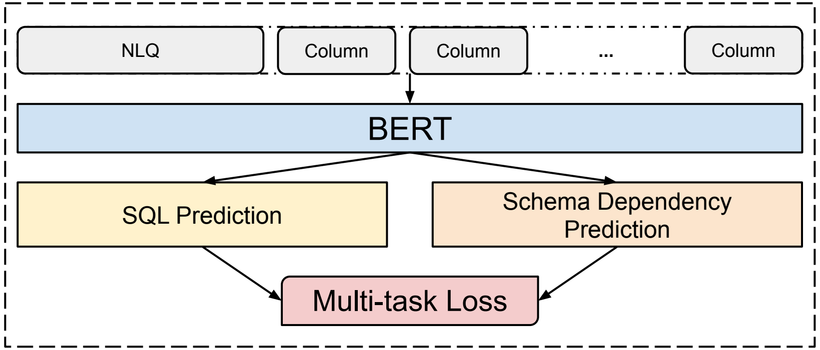
- Same **sketch** as SQLNet
- **Concatenates table columns to NLQ** for simultaneous encoding
- Uses a much **more complex network** after taking the BERT outputs
    - Almost identical to SQLNet
- Achieves **lower accuracy** on WikiSQL than HydraNet

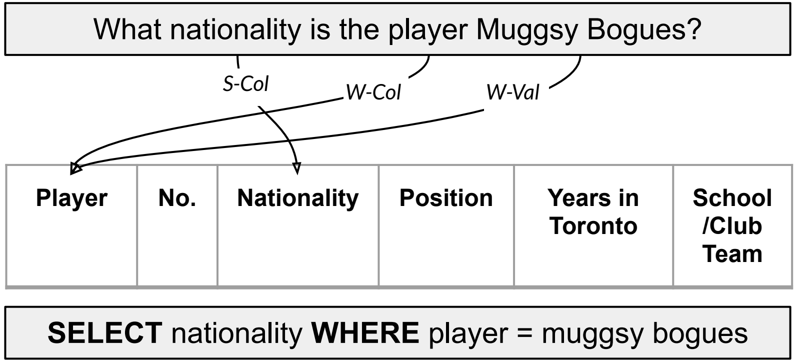


NL Representation	Schema Linking
BERT	None
Input Encoding	**Decoder Output**
Concatenate	Sketch-based

SDSQL

- Predicts SQL **similarly to SQLova**
- **Schema Dependency** learning along with SQL prediction
  - select-column (S-Col)
  - select-aggregation (S-Agg)
  - where-column (W-Col)
  - where-operator (W-Op)
  - where-value (W-Val)
- Automatically generate **dependency training data** based on expected SQL

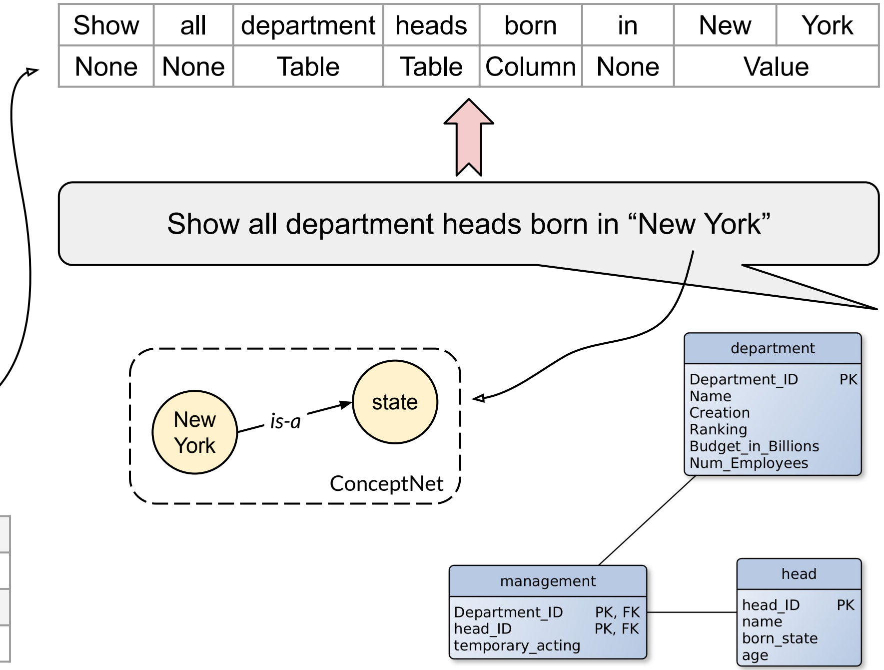




NL Representation	Schema Linking
BERT	Classifier
Input Encoding	**Decoder Output**
Concatenate	Sketch-based

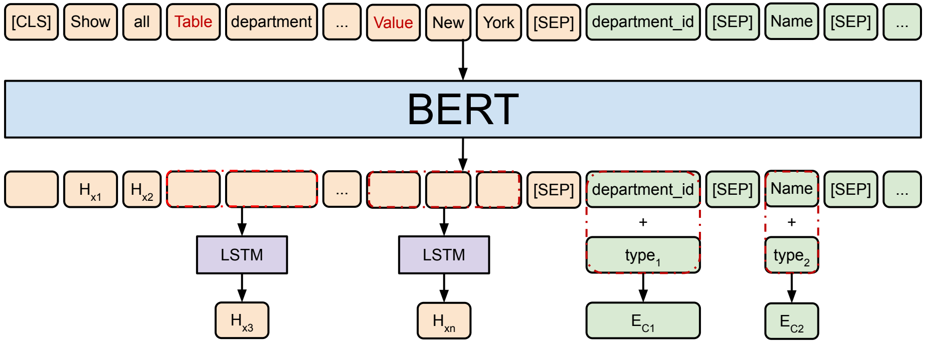
IRNet - Schema Linking

- 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



NL Representation	Schema Linking
GloVe/BERT	n-gram match, Knowledge graphs
Input Encoding	**Decoder Output**
Separately(GloVe)/Concatenate(BERT)	Grammar-based

IRNet - Encoding

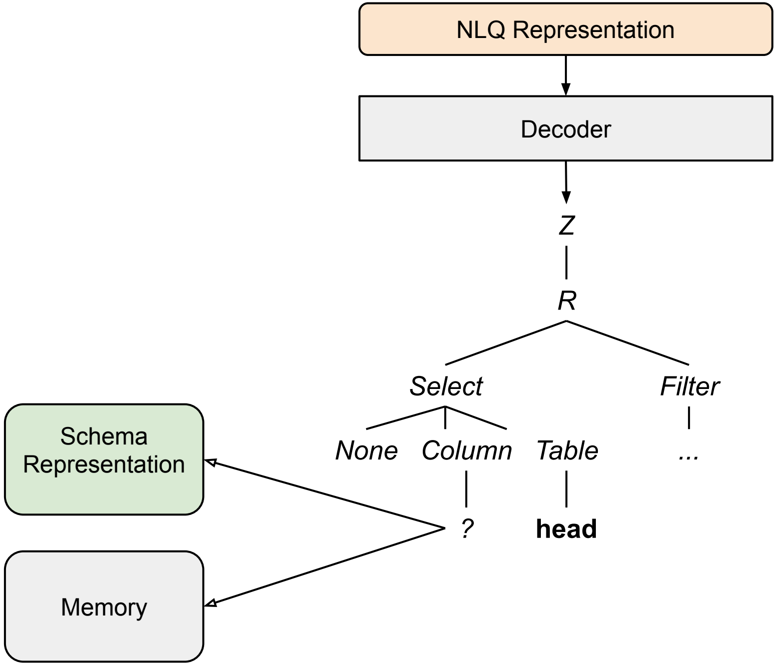


- Input can be encoded with **GloVe or BERT**
  - Accuracy with BERT is 8% higher
- **Schema link tokens** are appended to the matched NLQ spans
- Spans with multiple tokens are reduced to a **single token** using LSTM networks
- Column tokens are added to a **type embedding** (int, string, etc.)

NL Representation	**Schema Linking**
GloVe/BERT	n-gram match, Knowledge graphs
Input Encoding	**Decoder Output**
Separately(GloVe)/Concatenate(BERT)	Grammar-based

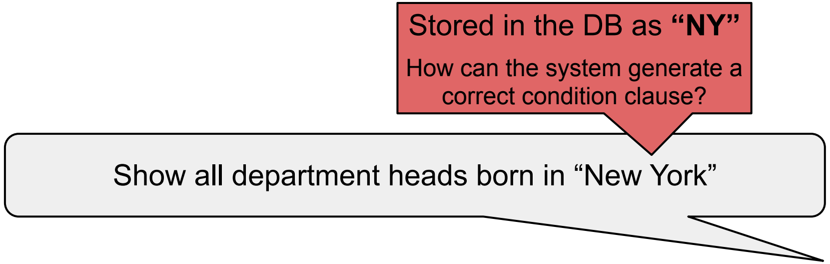
IRNet - Decoding

- 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 [28]
- When generating a **column or table name**, it can make a prediction from:
  - All **schema** elements
  - Elements already used in generated query (**memory**)



NL Representation	Schema Linking
GloVe/BERT	n-gram match, Knowledge graphs
Input Encoding	**Decoder Output**
Separately(GloVe)/Concatenate(BERT)	Grammar-based

ValueNet



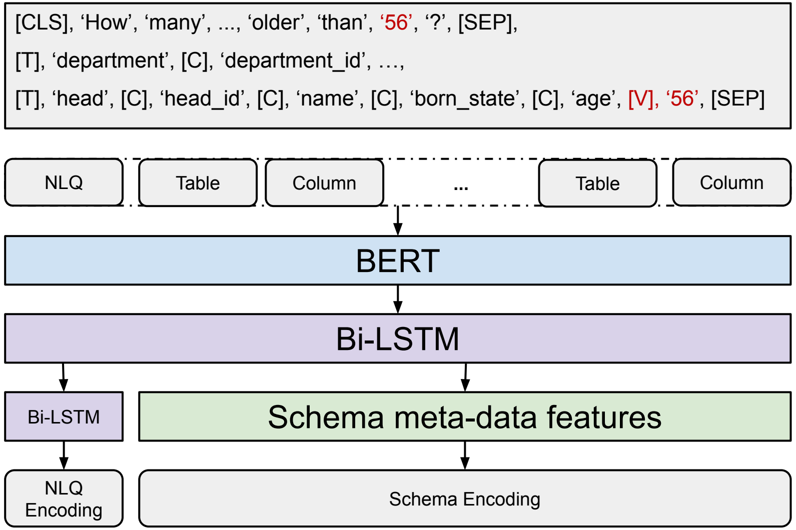
- Focuses on **better condition value** prediction
  - Most systems working on Spider do not predict condition values
  - We do not know the **set of options for values**
- Similar architecture to IRNet with some major improvements
  - Adds **value candidates** to the input
  - Predicts queries using an improved **SemQL 2.0** grammar

- Extended value candidate discovery
  - **Value extraction** using NER and heuristics
  - **Value candidate generation** using string manipulation (e.g. n-grams) and indices to search for similar values in the DB
  - **Value candidate validation** by looking up candidates in the DB
- Input Encoding: Concatenation of NLQ, table names, column names and discovered **value candidates**

NL Representation	**Schema Linking**
BERT	NER, heuristics, n-grams, indices
Input Encoding	**Decoder Output**
Concatenate	Grammar

BRIDGE - Encoder

- **Special tokens** [T], [C] and [V] are used to mark tables, columns, and linked values
- Schema linking is performed **only for values**, using **fuzzy string matching** against DB fields' **picklists**, for all tokens of the NLQ
- Encoded with BERT + LSTMs
- Tables and columns are also processed using **schema info** (type, foreign and primary keys)



NL Representation	Schema Linking
BERT	Fuzzy string matching with picklists
Input Encoding	**Decoder Output**
Concatenate	Sequence

BRIDGE - Decoder

- LSTM-based decoder
- At each step, the decoder performs one of the following actions:
  - Generate a token from a vocabulary
  - Generate a token from the NLQ
  - Generate a token from the schema
- All SQL queries are transformed to **execution order**
- **Schema-consistency guided decoding** using simple heuristics

SELECT count(*) **FROM** head **WHERE** age > 56

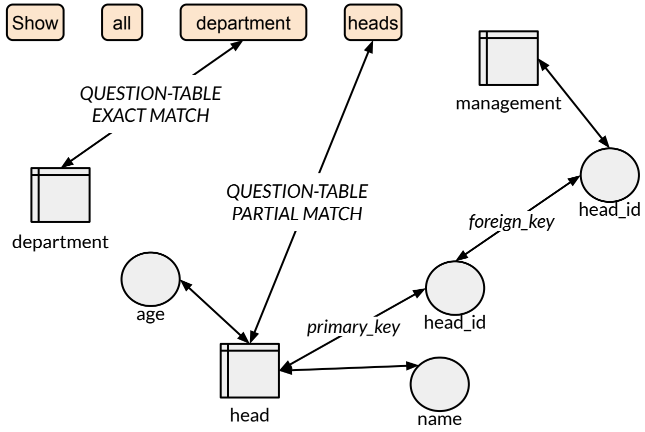
FROM head **WHERE** age > 56 **SELECT** count(*)

1. SQL syntax constraints
2. All schema attributes must be from tables appearing in the **FROM clause**

NL Representation	**Schema Linking**
BERT	String-matched values
Input Encoding	**Decoder Output**
Concatenate	Sequence

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

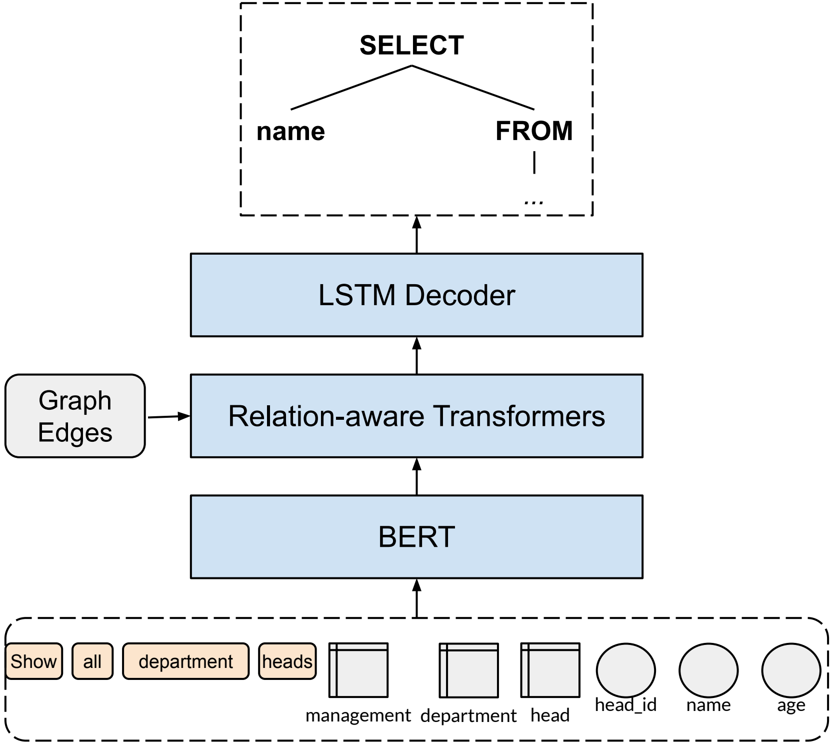


NL Representation	Schema Linking
GloVe/BERT	n-gram match, indices
Input Encoding	**Decoder Output**
Schema encoding	Grammar-based

[27] RAT-SQL (2020)

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



NL Representation	Schema Linking
GloVe/BERT	n-gram match, indices
Input Encoding	**Decoder Output**
Schema encoding	Grammar-based

Text-to-SQL System Overview

<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="5">Execution Accuracy on WikiSQL Test Set</td></tr>
<tr><td>SQLNet</td><td rowspan="4">Sketch-based</td><td>68.0 %</td></tr>
<tr><td>HydraNet</td><td rowspan="7">BERT</td><td>For each column</td><td>92.2 %
(using EG decoding)</td></tr>
<tr><td>SQLova</td><td rowspan="4">Concatenate</td><td>89.6 %
(using EG decoding)</td></tr>
<tr><td>SDSQL</td><td>Classifier</td><td>92.7 %
(using EG decoding)</td></tr>
<tr><td>IRNet</td><td>n-grams, KG</td><td rowspan="3">Grammar-based</td><td>60.1* %</td><td rowspan="4">Exact Set Match without Values on Spider Test Set</td></tr>
<tr><td>ValueNet</td><td>NER, heuristics, n-grams, indices</td><td>NA</td></tr>
<tr><td>RAT-SQL</td><td>n-grams, indices</td><td>Graph encoding</td><td>70.5* %</td></tr>
<tr><td>BRIDGE</td><td>Picklist string matching</td><td>Concatenate</td><td>Sequence</td><td>67.5* %</td></tr>
</table>

*Scores achieved using different language models and improvements

The Text-to-SQL Problem
Text-to-SQL Landscape
Available Benchmarks
Natural Language Representation
Text-to-SQL Deep Learning Taxonomy
Key Text-to-SQL Systems

Challenges & Research Opportunities

Challenges

Benchmarks?

Focus on effectiveness based on the number of queries translated

They do not:

✗ measure query expressivity

✗ measure time

✗ allow for more than one correct answers

To build better text-to-SQL systems as well as combine the best of existing approaches, we need to understand the capabilities of existing systems in depth.

THOR Query Benchmark [1]

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

SQL Challenges

NL Challenges

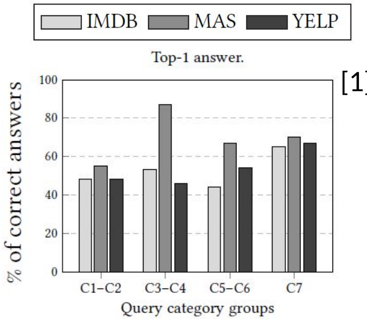
Category		Keyword	Natural Language
C1	**No joins & no metadata**	*"Brad Pitt"*	*Find about "Brad Pitt"*
C2	**Joins & no metadata**	*"Brad Pitt" "Fight Club"*	*Did "Brad Pitt" act in "Fight Club"?*
C3	**No joins & metadata**	*movie "Star Wars" prod_year*	*Find the production year of the movie "Star Wars"*
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 m*
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*

Challenges

Universal Solutions?

Different data sets present different intricate characteristics

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



[1]

Try out a DL system on SDSS (Sloan Digital Sky Survey)

Can we build systems that work well for different datasets?

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

Answer Validation?

How can the user confirm that the results match the intention of the query?

Natural language explanations (or SQL-to-NL)

Challenges

Fact Checking? [32,33,34]

Can we check a NL fact against a database?
Can we repair the claim with the correct information?

More Challenges

- dealing with context
- text-to-SPARQL
- text-to-vis

Building Natural Language Interfaces to Databases has come a long way

… and has a long way to go

Thank you for your attention :)

George Katsogiannis-Meimarakis
Georgia Koutrika

ATHENA Research & Innovation
Information Technologies

References (1/3)

[1] O. Gkini, T. Belmpas, G. Koutrika, Y. Ioannidis. An In-Depth Benchmarking of Text-to-SQL Systems. ACM SIGMOD 2021.
[2] Vagelis Hristidis and Yannis Papakonstantinou. 2002. Discover: Keyword Search in Relational Databases. In VLDB. 670–681.
[3] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. 2003. Efficient IR-style Keyword Search over Relational Databases. In VLDB. 850–861.
[4] Yi Luo, Xuemin Lin,WeiWang, and Xiaofang Zhou. 2007. Spark: Top-k Keyword Query in Relational Databases. In ACM SIGMOD. 115–126
[5] Zhong Zeng, Mong Li Lee, and Tok Wang Ling. 2016. Answering Keyword Queries involving Aggregates and GROUPBY on Relational Databases. EDBT (2016), 161–172.
[6] Lukas Blunschi, Claudio Jossen, Donald Kossmann, Magdalini Mori, and Kurt Stockinger. 2012. SODA: Generating SQL for Business Users. PVLDB 5, 10 (2012), 932–943.
[7] Fei Li and H. V. Jagadish. 2014. Constructing an Interactive Natural Language Interface for Relational Databases. PVLDB 8, 1 (Sept. 2014), 73–84.
[8] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, and Fatma Özcan. 2016. ATHENA: An Ontology-Driven System for Natural Language Querying over Relational Data Stores. VLDB.
[9] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. CoRR, September 2017
[10] 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.

References (2/3)

[11] 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.
[12] S. Hochreiter and J. Schmidhuber . 1997. Long Short-term Memory. Neural computation. 9. 1735-80.
[13] T. Mikolov, K. Chen, G. Corrado and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space.
[14] J. Pennington, R. Socher and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. EMNLP 2014.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20]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 (3/3)

[21] 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).
[22] X. Xu, C. Liu and D. Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning.
[23] W. Hwang, J. Yim, S. Park and M. Seo. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization.
[24] Q. Lyu, K. Chakrabarti, S. Hathi, S. Kundu, J. Zhang and Z. Chen. 2020. Hybrid Ranking Network for Text-to-SQL.
[25] 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.
[26] 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.
[27] 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.
[28] 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).
[29] 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.
[30] B. Hui, X. Shi, R.Geng, B. Li, Y. Li, J. Sun and Xiaodan Zhu. Improving Text-to-SQL with Schema Dependency Learning. 2021
[31] U. Brunner and K. Stockinger. ValueNet: A Neural Text-to-SQL Architecture Incorporating Values. 2020
[32] Mohammed Saeed and Paolo Papotti. Fact-checking Statistical Claims with Relational Datasets. IEEE Data Engineering, 2021
[33] S.Jo, I. Trummer, W. Yu, X. Wang, C. Yu, D. Liu, and N. Mehta. Verifying text summaries of relational data sets. SIGMOD ’19
[34] G. Karagiannis, M. Saeed, P. Papotti, and I. Trummer. Scrutinizer: Fact checking statistical claims. Proc. VLDB Endow., 2020