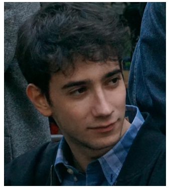
Deep Learning Approaches for Text-to-SQL Systems

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

ATHENA Research & Innovation Information Technologies

Presenters



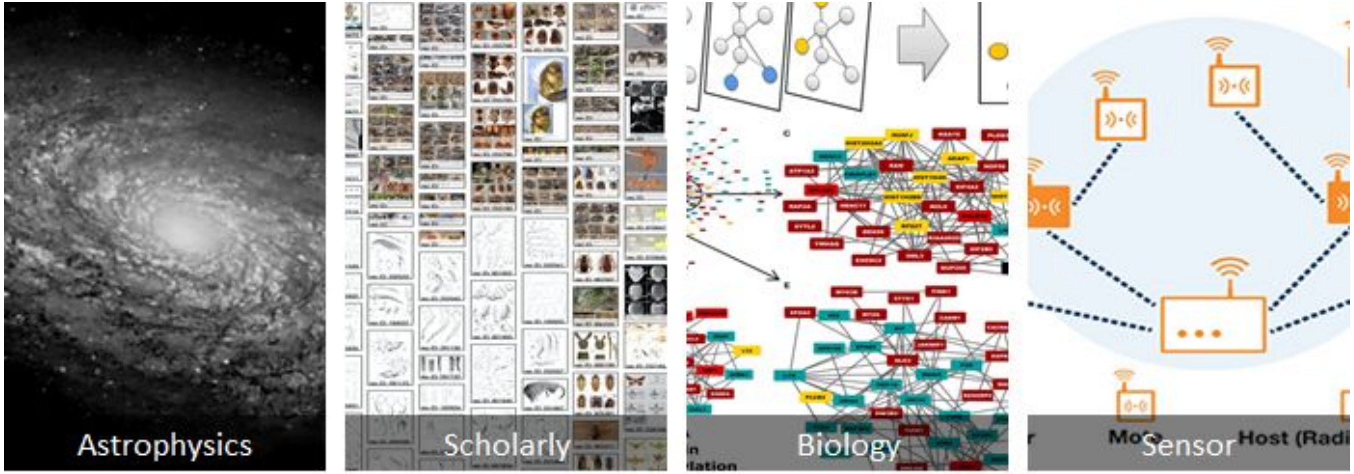
George Katsogiannis

- **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

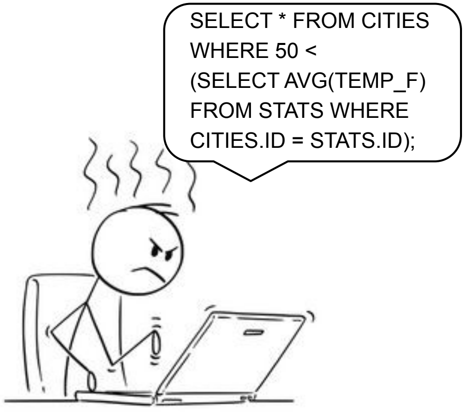
Why Text-to-SQL Systems?



- 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?

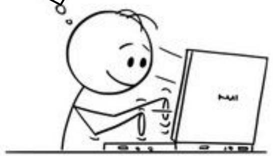
- **Data volume** and **complexity** make it difficult to query data.
- Database query interfaces are notoriously **user-UNFRIENDLY.**



Why Text-to-SQL Systems?

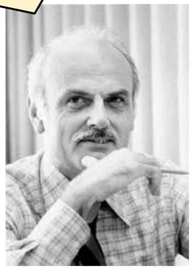
Expressing queries in natural language can open up data access to everyone

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)



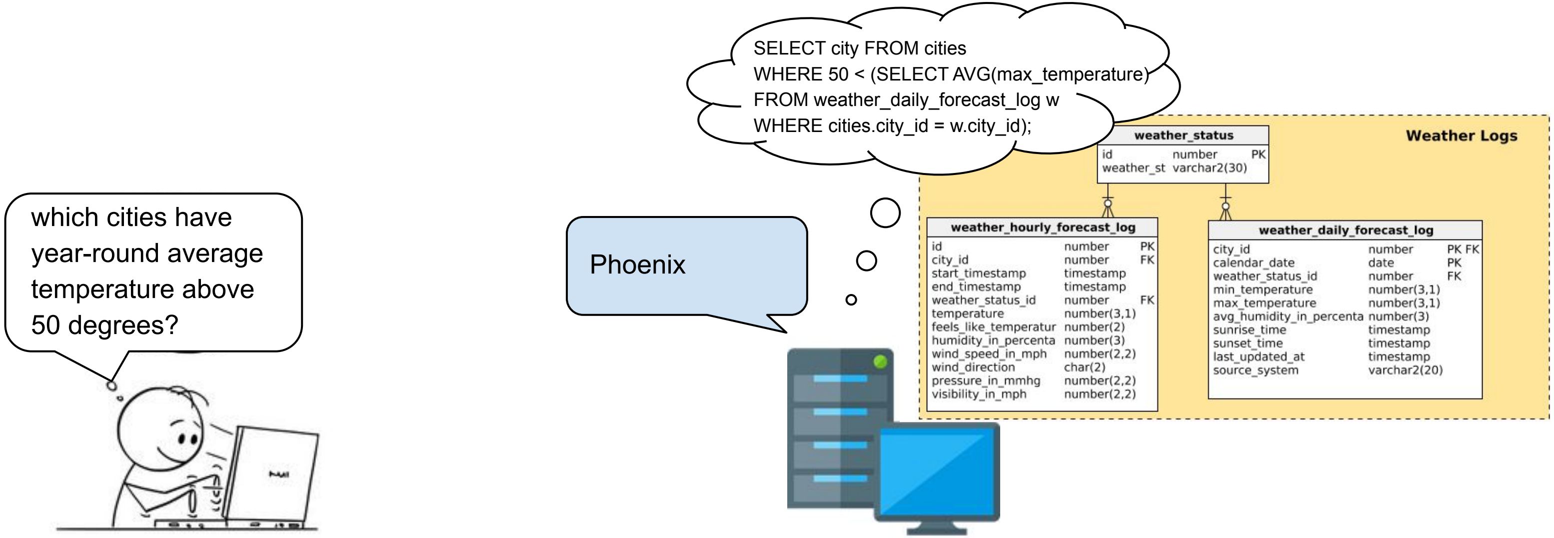
Tutorial Outline

1. The Text-to-SQL Problem
2. Text-to-SQL Landscape
3. Available Benchmarks
4. Natural Language Representation
5. Text-to-SQL Deep Learning Approaches
6. Key Text-to-SQL Systems
7. Challenges & Research Opportunities

The Text-to-SQL Problem

Text-to-SQL Landscape
Available Benchmarks
Natural Language Representation
Text-to-SQL Deep Learning Approaches
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

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

    “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”

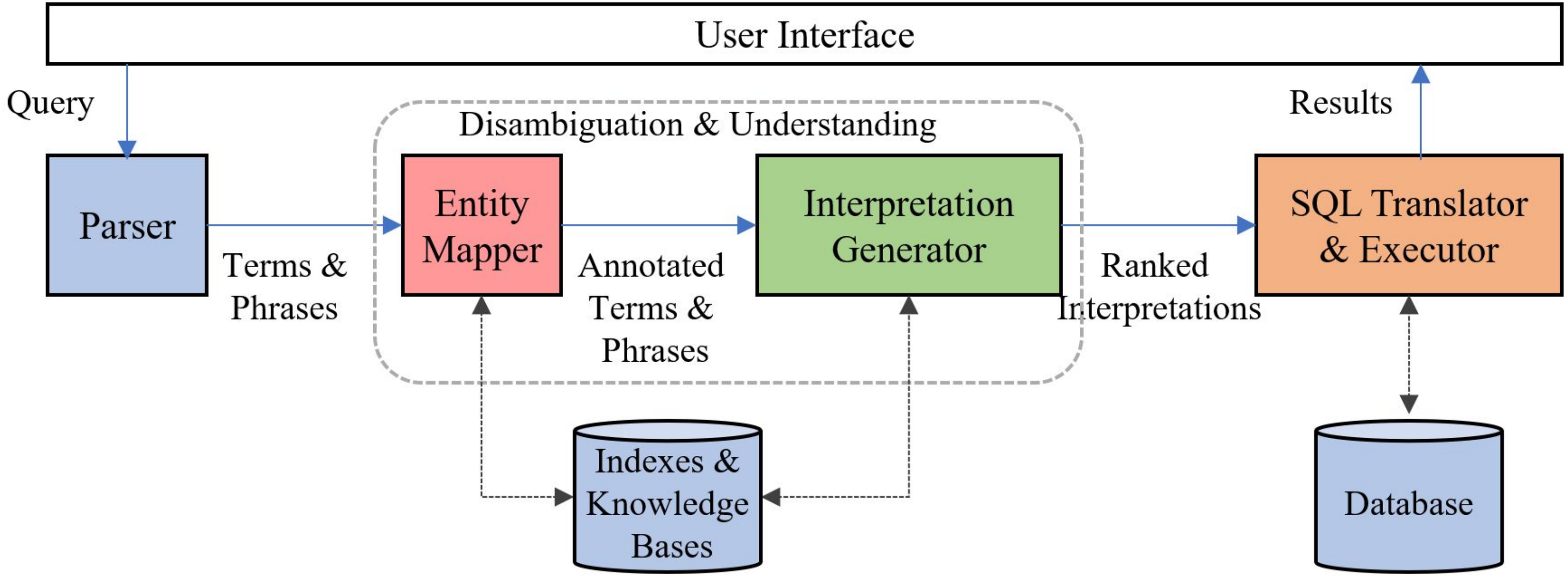
    Simple NLQ that might need 3,4 or 5 JOINs

The Text-to-SQL Problem

Text-to-SQL Landscape

Available Benchmarks
Natural Language Representation
Text-to-SQL Deep Learning Approaches
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]

- **SODA** 🔗 [6]
  enriches the system knowledge (i.e. inverted indexes) with additional knowledge sources

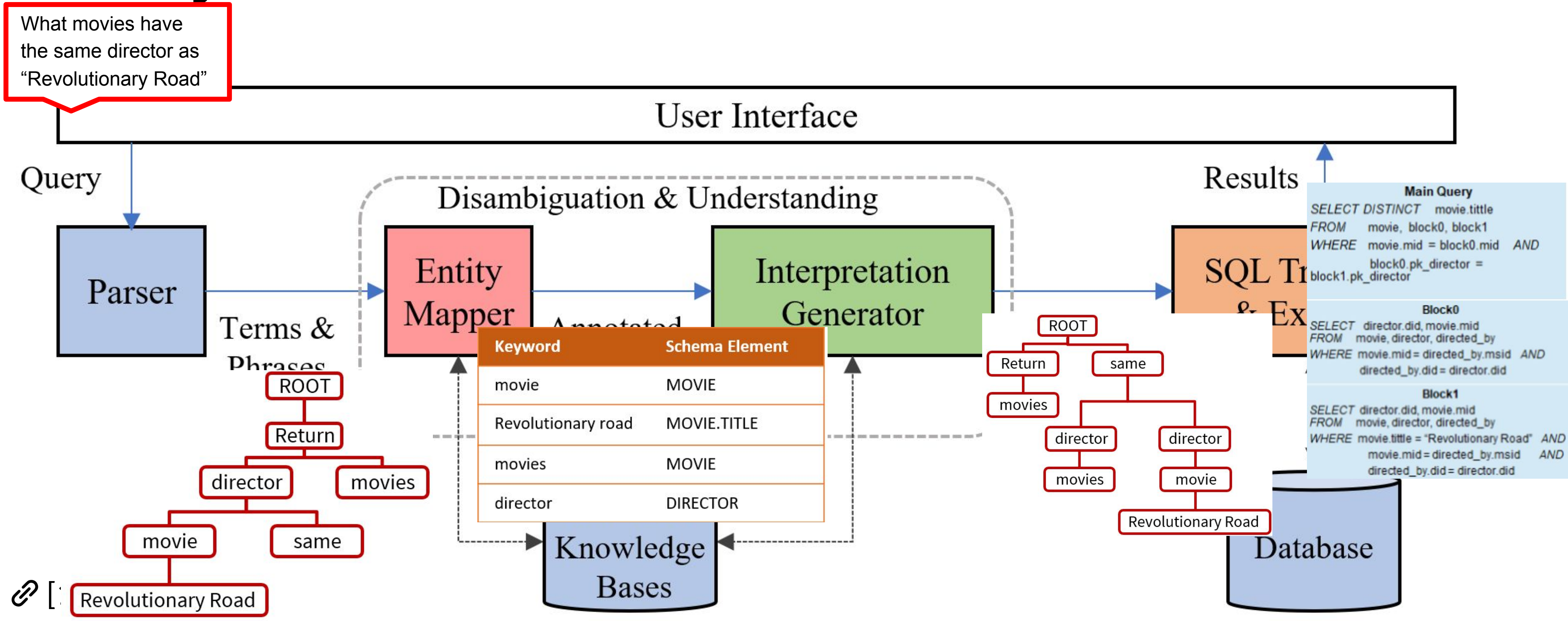
Generations of Text-to-SQL Systems

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



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

2021
- 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 Approaches
Key Text-to-SQL Systems
Challenges & Research Opportunities

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

	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**

Wikipedia (original table)

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

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!

Spider

- Large-scale complex and cross-domain text-to-SQL dataset
  - 10,181 questions and 5,693 SQL queries on 200 DBs 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
- Less queries and tables than WikiSQL but better quality and complexity

[10] Spider (2018)

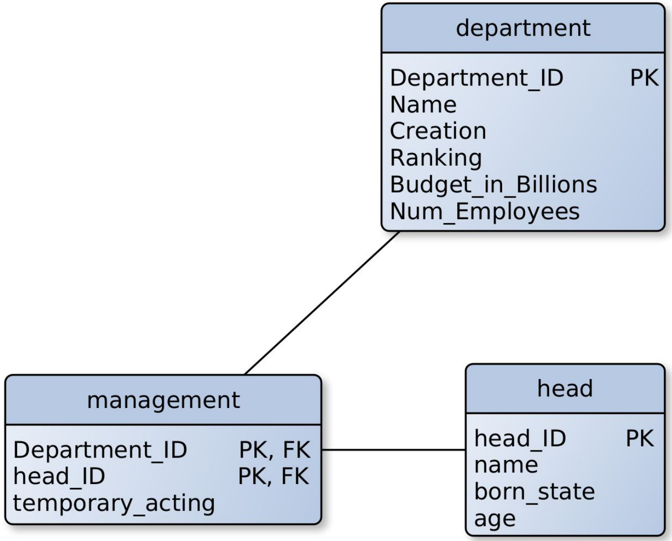
Spider: Example

NLQ:

How many heads of the departments are older than 56 ?

SQL:

SELECT count(*)
FROM head
WHERE age > 56



Database: Department Management

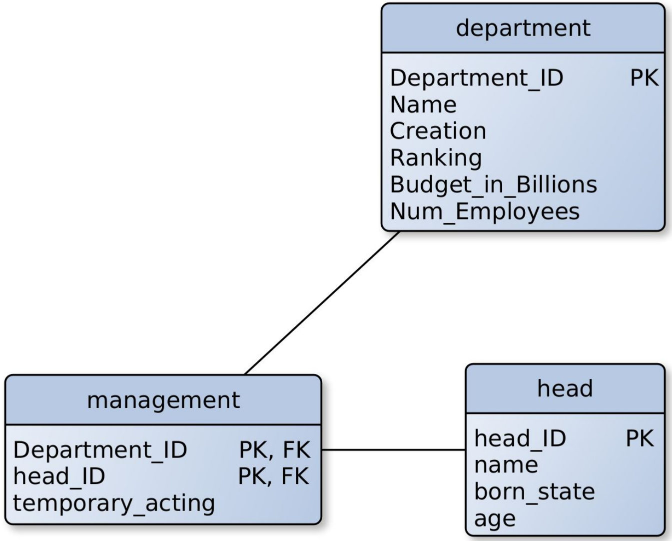
Spider: Example

NLQ:

Which department has more than 1 head at a time?
List the id, name and the number of heads.

SQL:

SELECT T1.department_id , T1.name , count(*)
FROM management AS T2
JOIN department AS T1
ON T1.department_id = T2.department_id
GROUP BY T1.department_id
HAVING count(*) > 1



Database: Department Management

Natural Language Representation

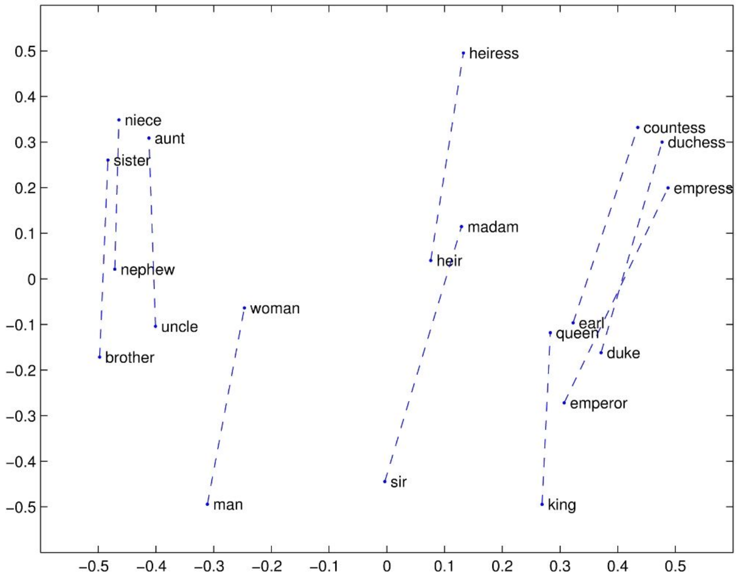
Natural Language Representation

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]

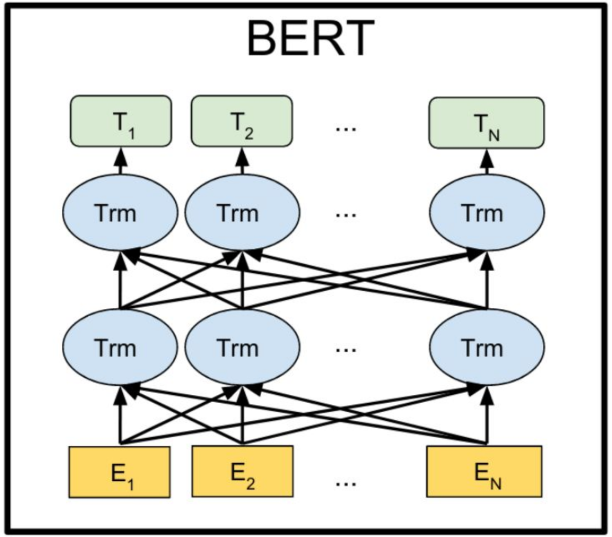
[14] GloVe (2014)

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



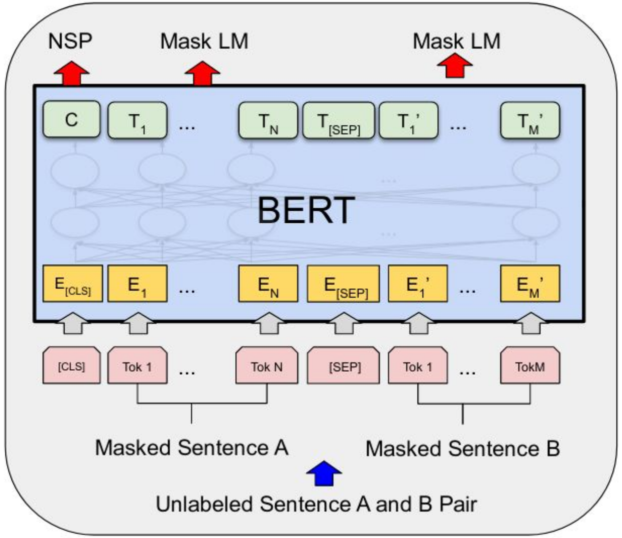
- **Output:** A sequence of tokens of equal length to the input
- Uses many **Transformer** layers
- **Input:** A sequence of token embeddings
    - Uses Wordpiece embeddings

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**)

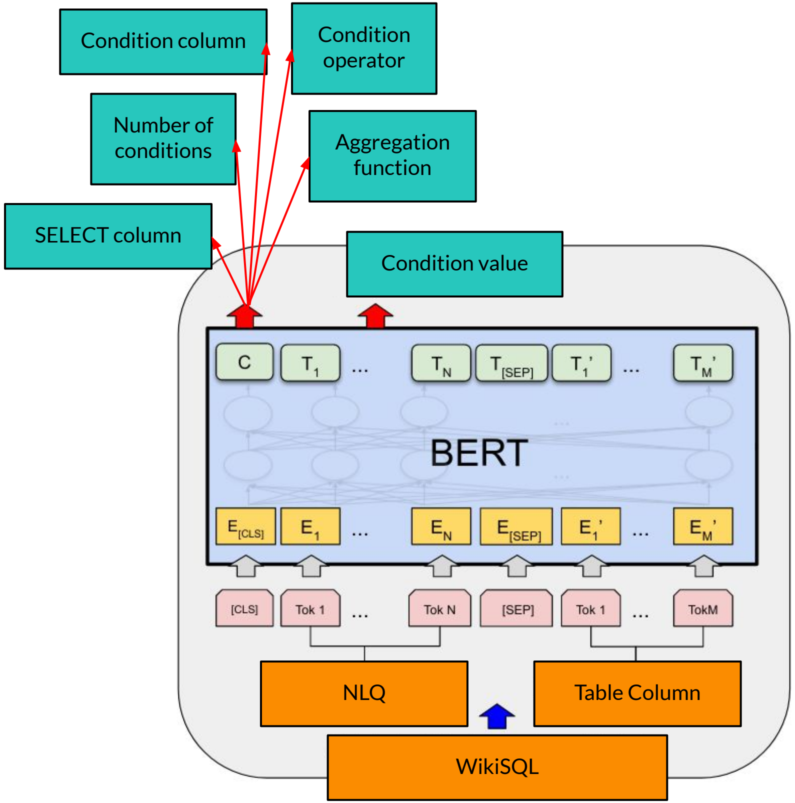
Input = **[CLS]** the man went to **[MASK]** store **[SEP]**
he bought a gallon **[MASK]** milk **[SEP]**

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



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

Text-to-SQL Deep Learning Approaches

Key Text-to-SQL Systems
Challenges & Research Opportunities

Text-to-SQL Approaches

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

- Sequence-to-Sequence
- Grammar-based
- Sketch-based Slot Filling

Sequence-to-Sequence

[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

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

Key Text-to-SQL Systems

Challenges & Research Opportunities

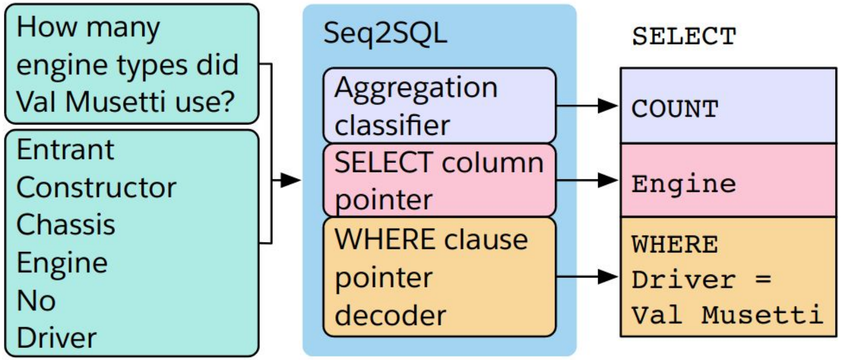
Text-to-SQL Systems

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

1. Seq2SQL
2. SQLNet
3. HydraNet
4. SQLova
5. IRNet
6. RAT-SQL

Seq2SQL

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

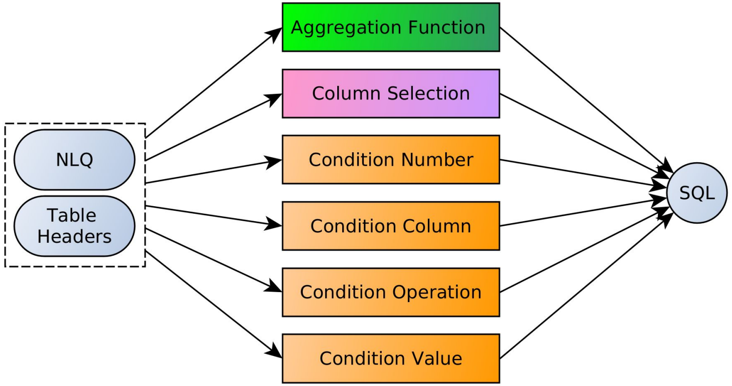


[9] Seq2SQL (2017)

SQLNet

- Completely **sketch-based**
- Each component has **its own** LSTM encoder
- 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

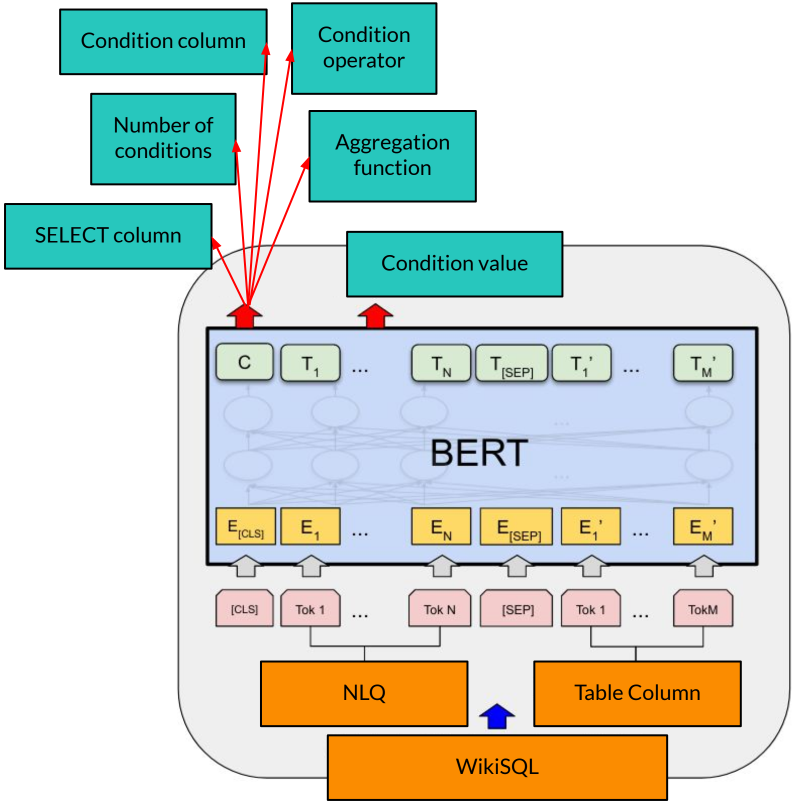
SELECT <*AGG*> <*COLUMN*>
(**WHERE** <*COLUMN*> <*OP*> <*VALUE*>
(**AND** <*COLUMN*> <*OP*> <*VALUE*>) *) ?



[22] SQLNet (2017)

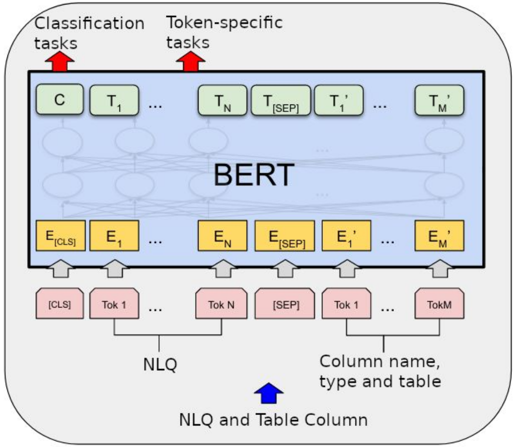
HydraNet

- 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**
    - This is in contrast to the common approach of processing all the table info at once



[24] HydraNet (2020)

HydraNet



1. **INPUT:** For each column of the table, construct the input: ([CLS], NLQ, [SEP], column_type, table_name, column_name, [SEP])
2. Give input to BERT
3. Classification tasks:

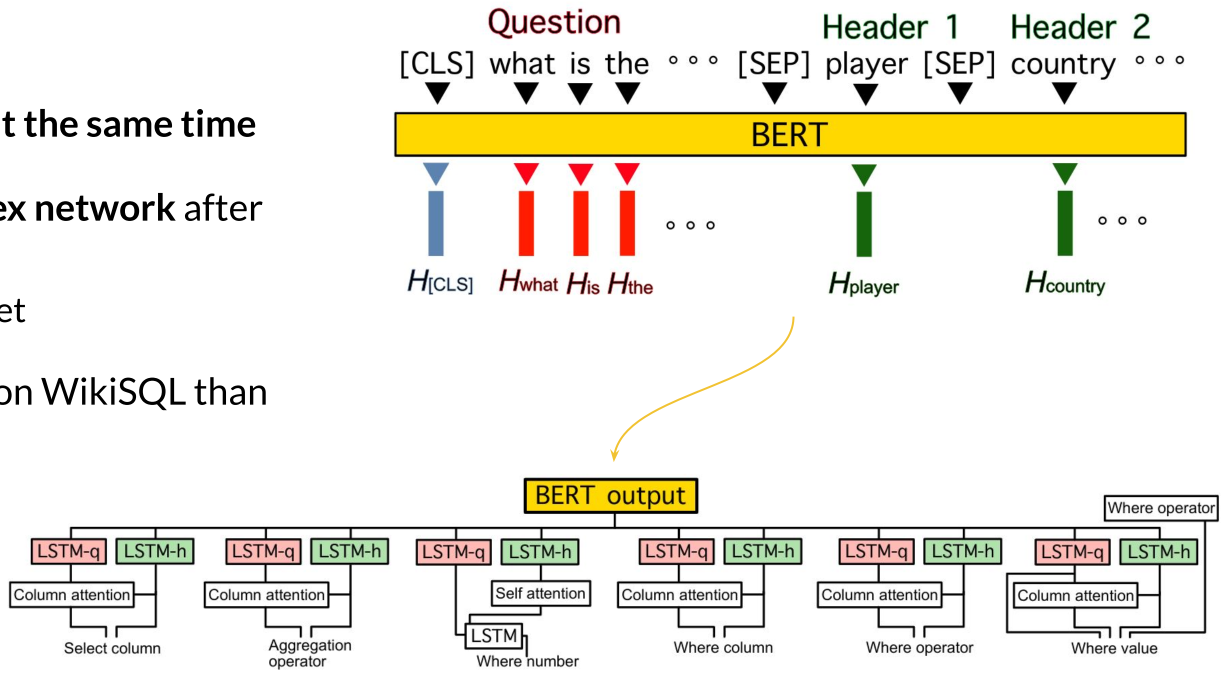
$$P(c_i \in S_q|q) = \text{sigmoid}(W_{sc} \cdot h[CLS])$$

   - ➢ 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 an **operator** in WHERE clause for column *i*
4. 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

$$P(y_j = \text{start}|c_i,q) = \text{softmax}(W_{start} \cdot h^q_j)$$

SQLova

- Same **sketch** as SQLNet
- Gives **all column names at the same time**
- Uses a much more **complex network** after taking the BERT outputs
  - Very similar to SQLNet
- Achieves **lower** accuracy on WikiSQL than HydraNet



[23] SQLova (2019)

A note on Execution-Guided Decoding

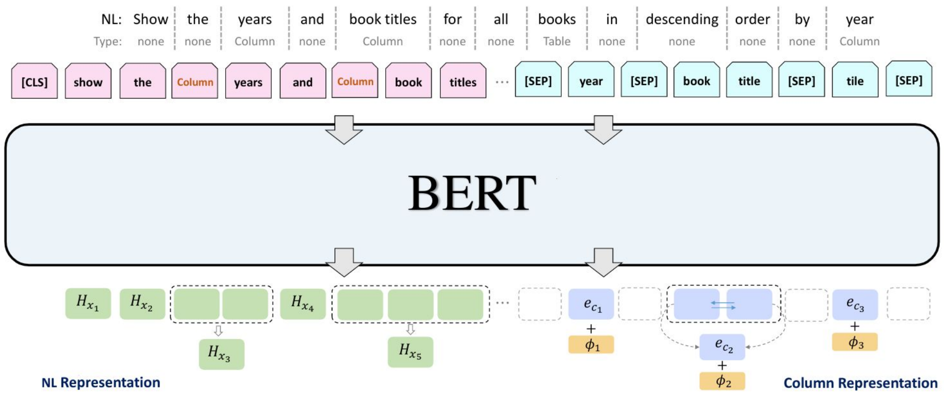
- Sketch-based approaches greatly **reduce** the possibility of errors
- There are still a few possibilities
    - **Aggregation function mismatch** (e.g. AVG on string type)
    - **Condition type mismatch** (e.g. comparing a float type column with a string type value)

- 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)

IRNet - Encoding

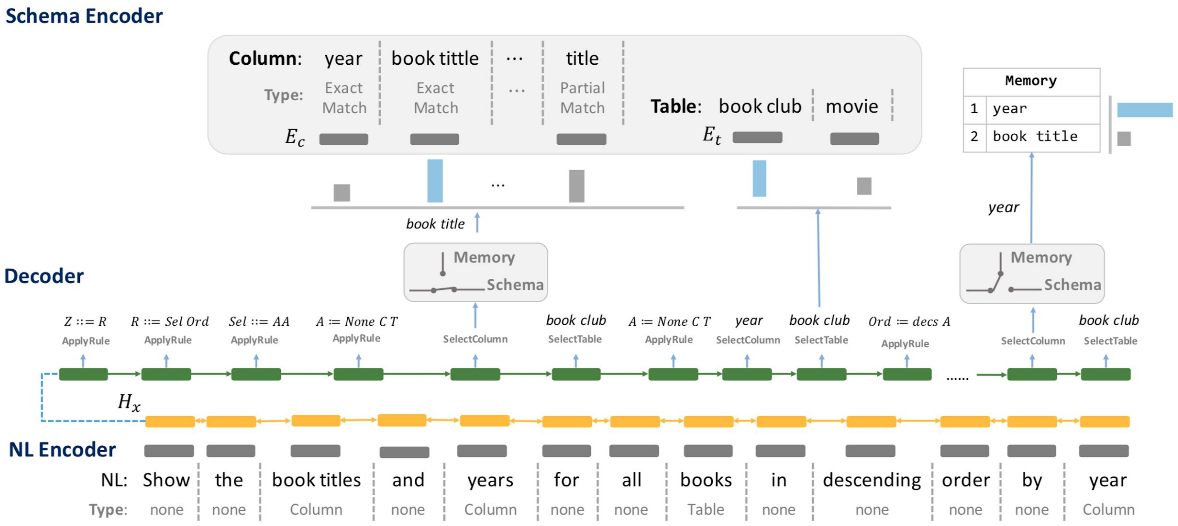
- Performs **Schema Linking**
  - Adds tokens that indicate matches to either a **table**, **column** or **value** of the database
- NL, column and table **encoding**
  - Simple Word Embeddings or BERT
- Additional token processing to create a **single token** for each entity



[26] IRNet (2019)

IRNet - Decoding

- Generates **SemQL** instead of SQL
- Generate a SemQL query **as an Abstract Syntax Tree**
  - [28] A Syntactic Neural Model for General-Purpose Code Generation (2017)
- When generating a **column or table name**, it can make a prediction from:
  - All **schema** columns
  - Columns already used in generated query (**memory**)



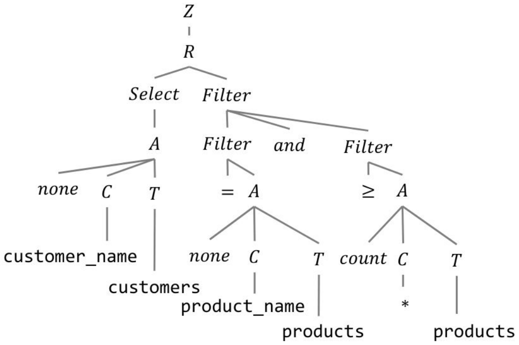
IRNet - SemQL

NL: List the names of the customers who have once bought product "food".

SQL:

```sql
SELECT T1.customer_name FROM customers AS T1 JOIN
orders AS T2 JOIN order_items AS T3 JOIN products AS
T4 WHERE T4.product_name = "food" GROUP BY
T1.customer_id HAVING count(*) >= 1
```

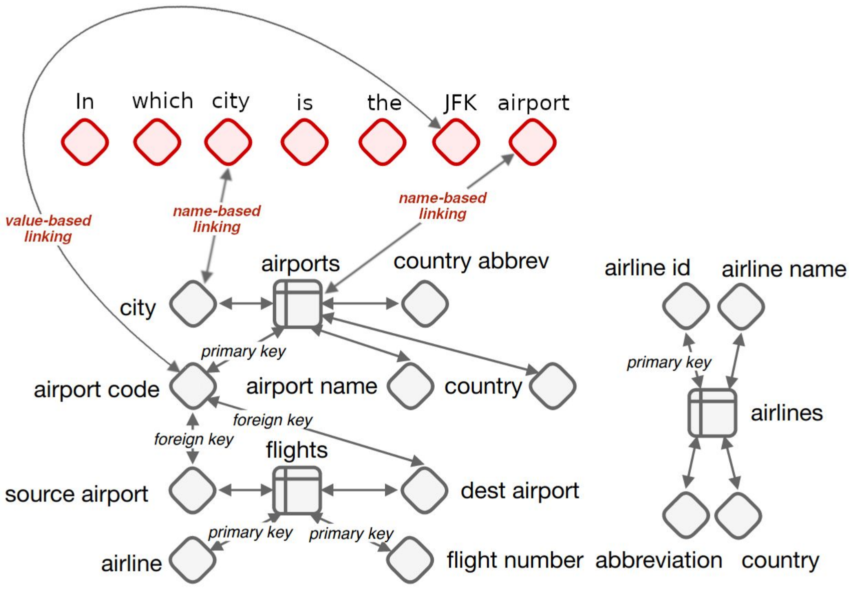
SemQL:



$$
\begin{aligned}
Z &::= intersect\ R\ R \mid union\ R\ R \mid except\ R\ R \mid R \\
R &::= Select \mid Select\ Filter \mid Select\ Order \\
&\quad \mid Select\ Superlative \mid Select\ Order\ Filter \\
&\quad \mid Select\ Superlative\ Filter \\
Select &::= A \mid A\ A \mid A\ A\ A \mid A\ A\ A\ A \mid A\ A \cdots A \\
Order &::= asc\ A \mid desc\ A \\
Suerlative &::= most\ A \mid least\ A \\
Filter &::= and\ Filter\ Filter \mid or\ Filter\ Filter \\
&\quad \mid > A \mid > A\ R \mid < A \mid < A\ R \\
&\quad \mid \geq A \mid \geq A\ R \mid = A \mid = A\ R \\
&\quad \mid \neq A \mid \neq A\ R \mid between\ A \\
&\quad \mid like\ A \mid not\ like\ A \mid in\ A\ R \mid not\ in\ A\ R \\
A &::= \max C\ T \mid \min C\ T \mid count\ C\ T \\
&\quad \mid sum\ C\ T \mid avg\ C\ T \mid none\ C\ T \\
C &::= column \\
T &::= table
\end{aligned}
$$

RAT-SQL

- **Question-contextualized schema graph**
  - **Schema** nodes and **NLQ** word nodes
  - Edges are **relations** between them from:
    - Schema relations,
    - Name-based Linking and
    - Value-based Linking
- Encoding with GloVe & LSTM or BERT

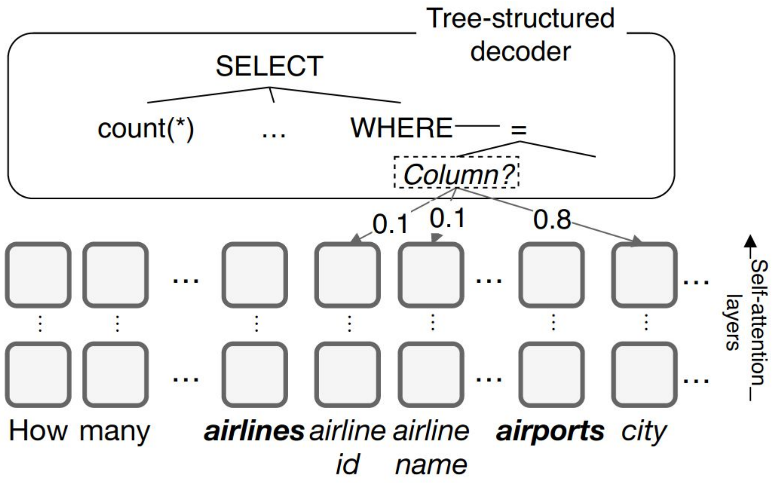


Question-contextualized Schema Graph: Grey nodes represent schema nodes and red nodes represent NLQ nodes.

[27] Rat-SQL (2020)

RAT-SQL (cont.)

- Specially modified Transformers, for **relation-aware self-attention**, biases the network towards known relations
- SQL generation as an AST, by predicting a sequence of **decoder actions**
  - [28] A Syntactic Neural Model for General-Purpose Code Generation (2017)
  - Encoded representations are used to fill column and table names in the AST



Key Text-to-SQL Systems Overview

Comparing design choices that each system has to answer

- How is the input encoded?
- What kind of output is produced?
- How to handle schema linking?
- How is Natural Language represented?

Key Text-to-SQL Systems Overview

1. How is the input encoded?
   - Does the system get all the **needed information** to solve the problem?
   - Is it given in a **meaningful** way?

2. What kind of output is produced?
   - How to achieve high expressivity and generate **complex SQL queries**?
   - How to avoid generating **syntactically or semantically** incorrect queries?

3. How to handle schema linking?
   - Can the network do it **by itself**?
   - Is there room for **improvement** for the available schema linking methods?

4. How is Natural Language represented?
   - NL is one of the main **sources of complexity** in the text-to-SQL task
   - Improving NL representation has a **direct effect on performance**

Key Text-to-SQL Systems Overview

<table>
<tr><th></th><th></th><th>Input encoding</th><th>Decoder Output</th><th>Schema Linking</th><th>NL Representation</th></tr>
<tr><td>First neural approach for text-to-SQL</td><td>Seq2SQL</td><td rowspan="2">Separate encoding of NLQ and schema</td><td>Sequence</td><td rowspan="4">No, the network will figure it out</td><td rowspan="2">Word Embeddings</td></tr>
<tr><td>First completely sketch-based</td><td>SQLNet</td><td rowspan="3">Sketch-based</td></tr>
<tr><td>“Natural” use of BERT</td><td>HydraNet ★</td><td>NLQ with each column separately</td><td rowspan="4">Language models - Transfer Learning</td></tr>
<tr><td>Combined earlier approaches with BERT</td><td>SQLova</td><td rowspan="2">Concatenation of NLQ and schema</td></tr>
<tr><td>Decoding as SemQL AST</td><td>IRNet</td><td rowspan="2">Grammar-based</td><td rowspan="2">Yes, outside the neural network</td></tr>
<tr><td>Representing input as a graph</td><td>RAT-SQL ★★</td><td>Graph encoding</td></tr>
</table>

★ 3rd best for WikiSQL (1st is 0.5% better)

★★ Best for Spider

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

Challenges & Research Opportunities

Challenges

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

The text-to-SQL problem is still very hard!

- Different data sets present different intricate characteristics
  - No universal solutions
  - Domain-specific or application-specific solutions: ontologies, knowledge bases
- Understanding the full range of queries: from keywords to NL
  - Different systems allow different query expressivity
  - Combining systems

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).