Georgia Tech

DATA ANALYTICS USING DEEP LEARNING

GT 8803 // FALL 2018 // CHRISTINE HERLIHY

LECTURE #04: SEQ2SQL: GENERATING STRUCTURED QUERIES FROM NATURAL LANGUAGE USING REINFORCEMENT LEARNING

CREATING THE NEXT®

TODAY'S PAPER

- **Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning**
  - **Authors:**
    - Victor Zhong, Caiming Xiong, Richard Socher
    - They are all affiliated with Salesforce Research
  - **Areas of focus:**
    - Machine translation; deep learning and reinforcement learning for query generation and validation

TODAY'S AGENDA

- Problem Overview
- Context: Background Info on Relevant Concepts
- Key Idea
- Technical Details
- Experiments
- Discussion Questions

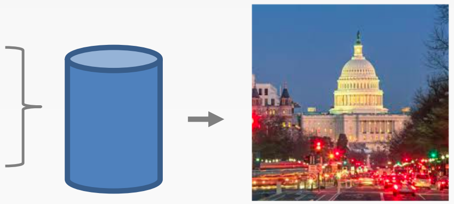
PROBLEM OVERVIEW

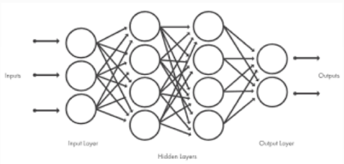
- **Status Quo:**
  – A lot of interesting data is stored in relational databases
  – To access this data, you have to know SQL
- **Objective:**
  – Make it easier for end-users to query relational databases by translating natural language questions to SQL queries
- **Key contributions:**
  – **Seq2SQL model:** a DNN to translate NL questions to SQL
  – **WikiSQL:** annotated corpus containing 80,654 questions mapped to SQL queries and tables from Wikipedia

What is the capital of the United States?

vs.

SELECT capital WHERE country = “United States”





Wikipedia
The Free Encyclopedia

CONTEXT: SQL CONCEPTS

- **SQL** is a declarative query language used to extract information from relational databases; results are returned as rows and columns
- A **schema** is a collection of database objects (here, tables)

- Even basic queries may include several clauses:
  - **Aggregation operation(s)**
    - (e.g., COUNT, MIN, MAX, etc.)
  - **SELECT** column(s) FROM schema.table
  - **WHERE** (condition1) AND (condition2)

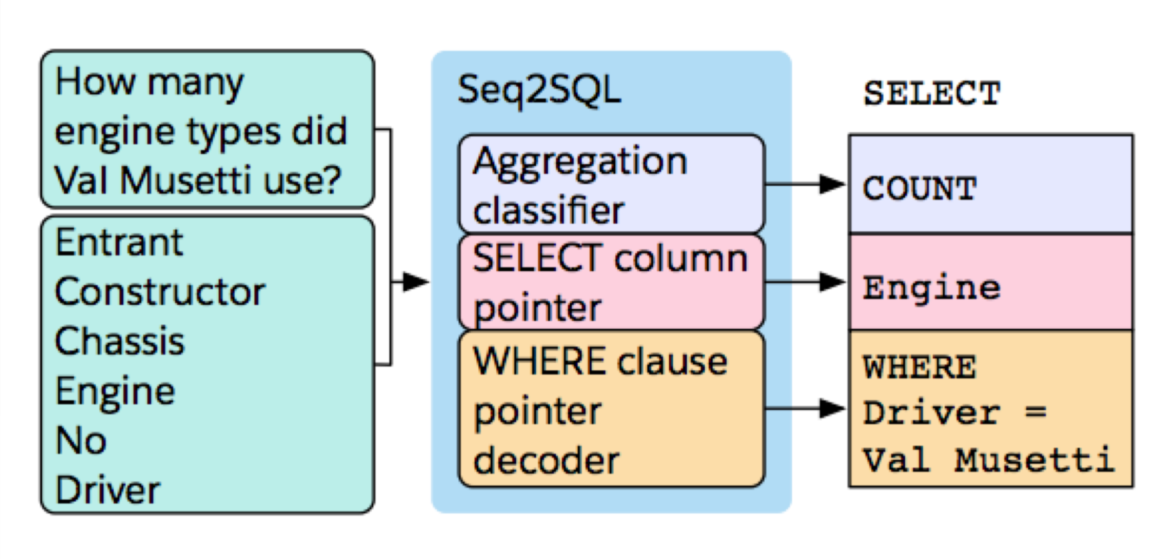


Image: https://arxiv.org/pdf/1709.00103.pdf

CONTEXT: DEEP LEARNING CONCEPTS

- **Recurrent Neural Networks (RNNs):**
  - Neural network architecture containing self-referential loops
  - Intended to allow knowledge/information learned in previous steps to influence the current prediction/output; well-suited for sequential/temporal data

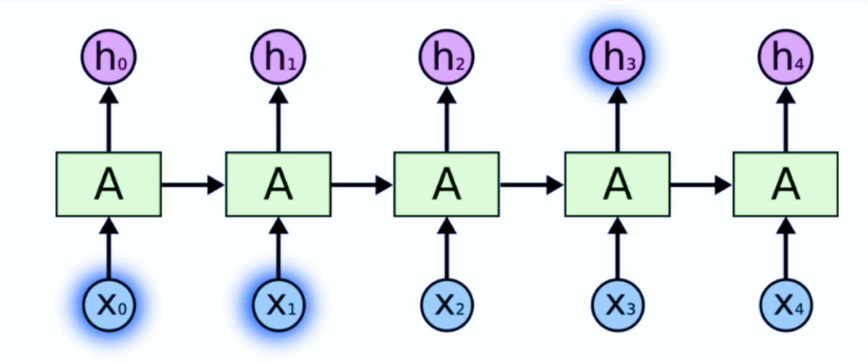


Image: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

CONTEXT: DEEP LEARNING CONCEPTS

- **Long Short-Term Memory (LSTM) architecture:**
  - Intended to mitigate vanishing/exploding gradient problem associated with RNNs
  - Better suited for longer-term temporal dependencies
  - Incorporate a memory cell and forget gate

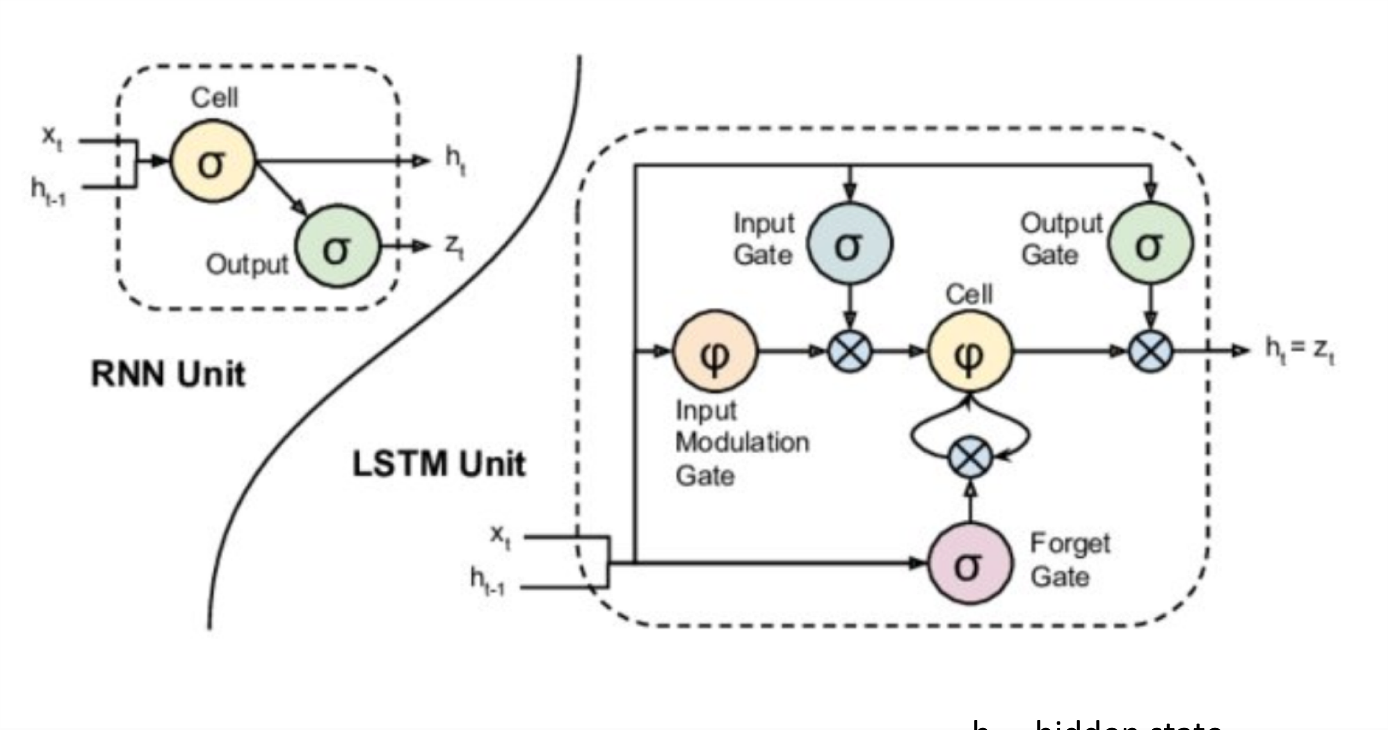


Image: https://www.researchgate.net/publication/319770438_Long-Term_Recurrent_Convolutional_Networks_for_Visual_Recognition_and_Description

h_t = hidden state
z_t = prediction at time step *t*

CONTEXT: DEEP LEARNING CONCEPTS

- **How are LSTMs used in this paper?**
  - Seq2SQL generates SQL queries token-by-token
  - They use LSTMs for encoding the embeddings associated with each word in the input sequence, and decoding each query token, y_s, as a function of the most recently generated token, y_{s-1}

Similar to Seq2Seq, but ∀ output token ∈ input

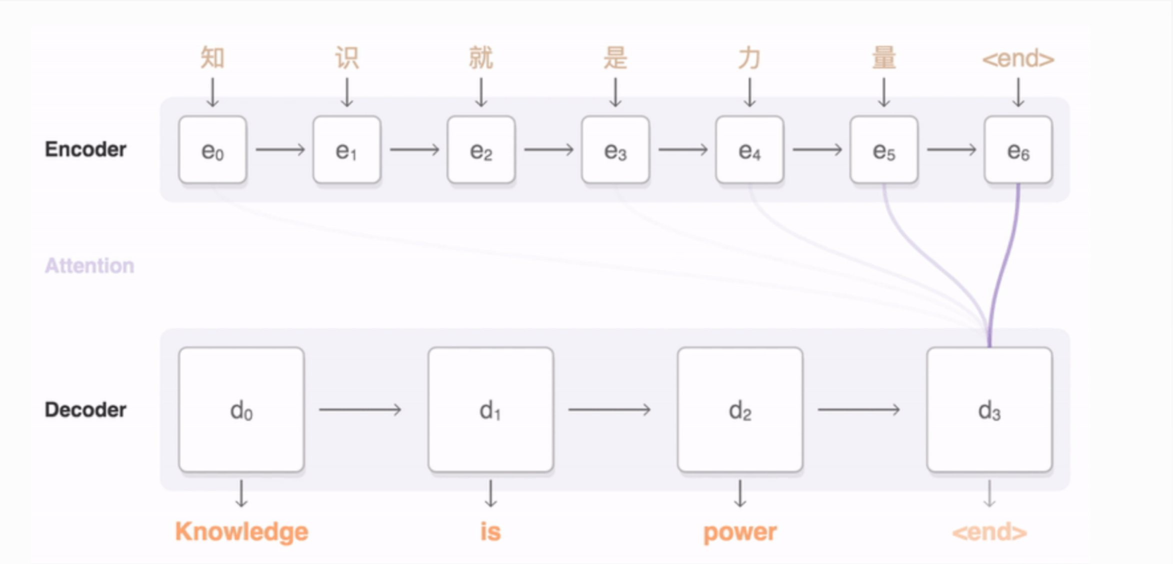
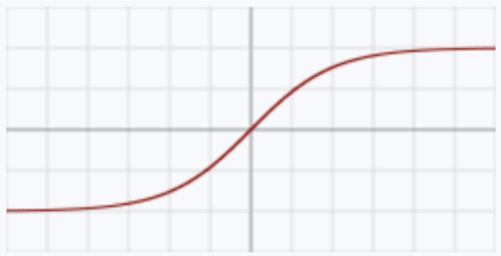


Image: https://google.github.io/seq2seq/

CONTEXT: DEEP LEARNING CONCEPTS

- **Activation functions** define the output of individual neurons in a DNN, given a set of input(s)

- **Relevant activation functions from this paper:**
  - Hyperbolic tangent (tanh):
    - Outputs values in (-1,1); less likely to get "stuck" than logistic sigmoid



$$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

Images and information: https://en.wikipedia.org/wiki/Activation_function

CONTEXT: DEEP LEARNING CONCEPTS

- The **loss function** of a DNN represents the error to be minimized
- **Cross-entropy loss:**
  - Measures the performance of a classifier whose output is a probability value in [0,1]
  - When number of classes = 2 (e.g., {0,1})
    - $-(y\log(p) + (1-y)\log(1-p))$
  - For number of classes, *M* > 2, compute loss for each label per observation, *o*, and sum:
    - $-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$

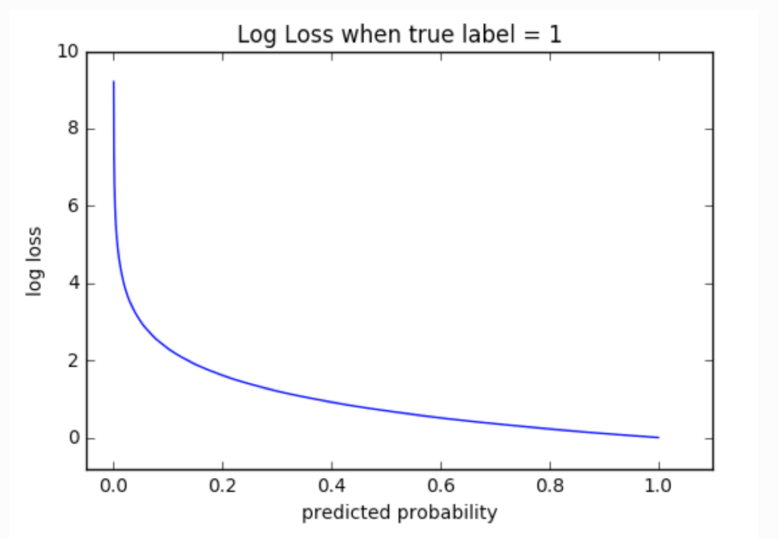


Image: https://google.github.io/seq2seq/

CONTEXT: REINFORCEMENT LEARNING

- **Reinforcement learning:** "learning what to do—how to map situations to actions–so as to maximize a numerical reward signal"
  - Agent must **explore** state space and **exploit** knowledge gained
  - Evaluative feedback based on actions, rather than action-independent instructional feedback

Source: Richard S. Sutton and Andrew G. Barto. 1998. Introduction to Reinforcement Learning (1st ed.). MIT Press, Cambridge, MA, USA.

CONTEXT: REINFORCEMENT LEARNING

- **Policy (π):**
  - "[D]efines the agent's way of behaving at a given time, and is a mapping from perceived states of the environment to actions to be taken when in those states"
  - Classical example is the grid-world problem

Grid-World Example Problem:

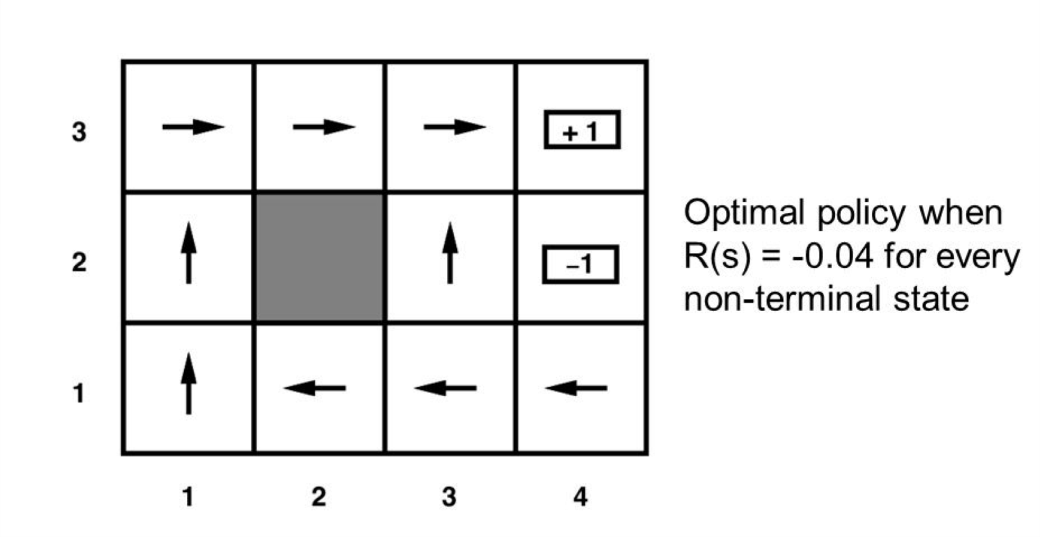


Image: https://slideplayer.com/slide/4757729/

Source: Richard S. Sutton and Andrew G. Barto. 1998. Introduction to Reinforcement Learning (1st ed.). MIT Press, Cambridge, MA, USA.

CONTEXT: REINFORCEMENT LEARNING

- **As applied in the paper:**
  - **States** correspond to the portion of the query generated thus far
  - **Actions** correspond to the selection of the next term in the output sequence, conditional on the input sequence and all terms selected so far
  - **Rewards** are assigned when the generated queries are executed; depend on validity, correctness, and string match

CONTEXT: REINFORCEMENT LEARNING

- **Teacher forcing:**
  - Refers to scenario where, after the model is trained, the actual or expected output sequence token at time step *t* is used as input when predicting token_{t+1}, instead of using the output generated by the DNN
  - In the paper, teacher forcing is used as an initial step when training the model for WHERE clause output
    - Policy is not learned from scratch
    - Rather, with TF as a foundation, they continue to policy learning
    - Why?

Information: https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/

CONTEXT: FOUNDATIONAL WORKS

- **Semantic parsing:**
  – Converting natural language utterance to logical/machine-interpretable representation
- **Baseline model:**
  – Attentional sequence to sequence neural semantic parser: Dong & Lapata (2016)
  – Goal of this paper was also to develop a generalized approach to query generation requiring minimal domain knowledge
  – They develop a sequence-to-tree model to incorporate hierarchical nature of semantic information

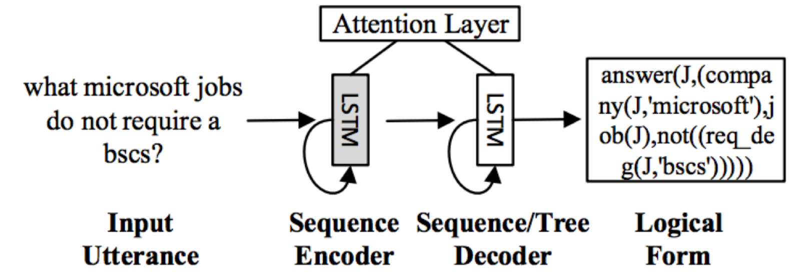
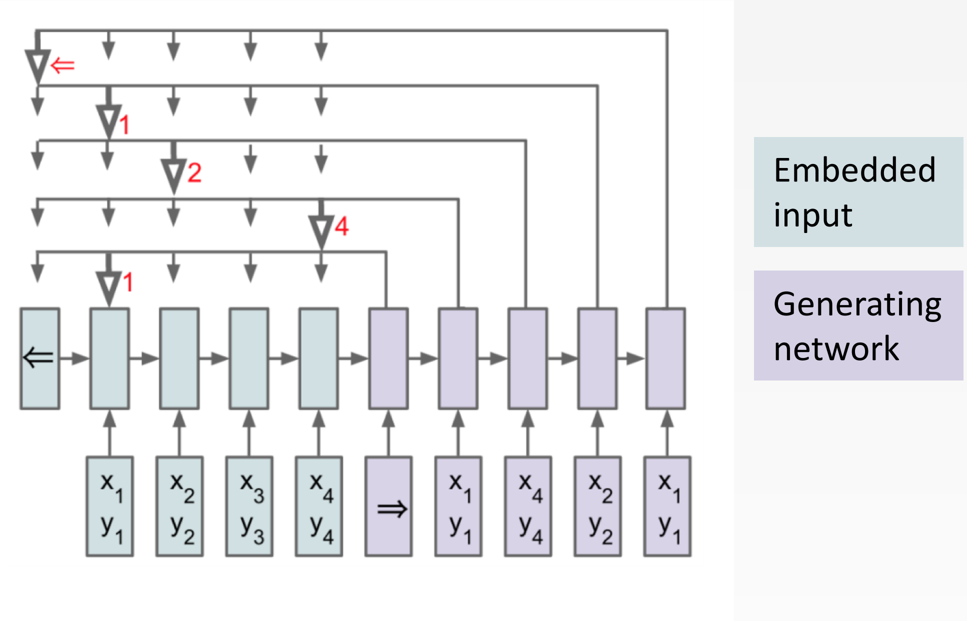


Figure 1: Input utterances and their logical forms are encoded and decoded with neural networks. An attention layer is used to learn soft alignments.

Image: https://arxiv.org/pdf/1601.01280.pdf

CONTEXT: FOUNDATIONAL WORKS

- **Augmented pointer network:**
  - Seq2SQL extends the work of Vinyals et al. (2015)
  - The referenced paper introduced Ptr-Net, a "neural architecture to learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in a [variable-length] input sequence"



(b) Ptr-Net

Image: https://arxiv.org/pdf/1506.03134.pdf

KEY IDEA

- **Objective:** Ingest a natural language question, a set of table column names, and the set of unique words in the SQL vocabulary; output a valid SQL query that returns correct results when compared to results from ground truth query, q_g.
- **How?**

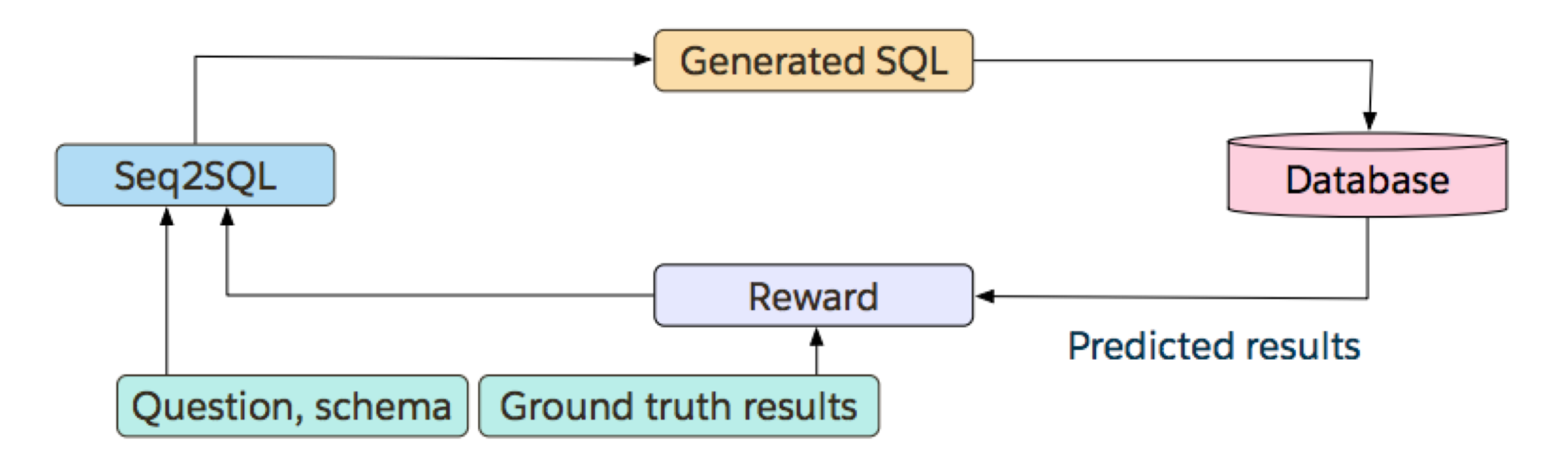
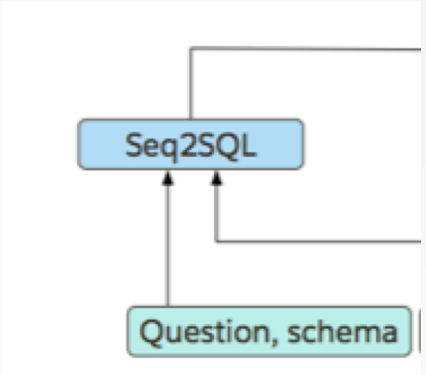


Image: https://arxiv.org/pdf/1709.00103.pdf

TECHNICAL DETAILS

- **Input sequence:**
  - Concatenation of {column names, the terms that form the natural language question, limited SQL vocabulary terms}



$$x = [\texttt{<col>}; x_1^c; x_2^c; \ldots; x_N^c; \texttt{<sql>}; x^s; \texttt{<question>}; x^q]$$

Images: https://arxiv.org/pdf/1709.00103.pdf

TECHNICAL DETAILS

- **Query generation:**
  - SQL queries are generated token-by-token
  - Seq2SQL has 3 component parts:
    - Aggregation operator (Does query need one or not? Which one?)
    - SELECT column required (note, input column tokens provide the alphabet; softmax function used to produce a distribution over possible columns)
    - Construction of the WHERE clause (RL is used for this)

TECHNICAL DETAILS

- **Role of deep learning:**
  - LSTM networks are used to encode vector embeddings of items from the input sequence, and decoded to obtain tokens that, when strung together, constitute the SQL query

Decoder output:

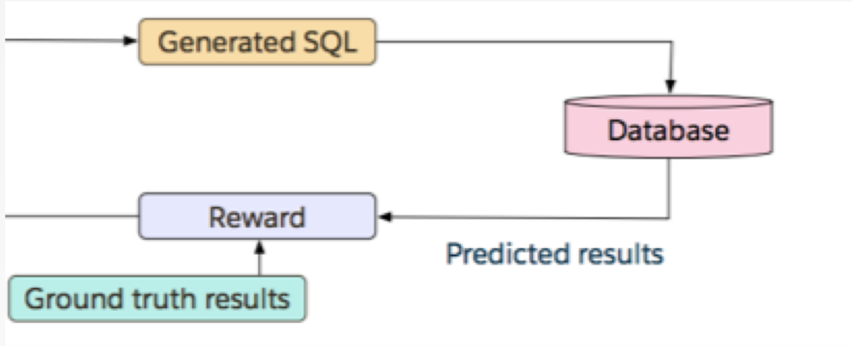
$$\alpha^{\text{ptr}}_{s,t} = W^{\text{ptr}} \tanh\left(U^{\text{ptr}} g_s + V^{\text{ptr}} h_t\right)$$

$a^{ptr}_{s,t}$ = Scalar attention score for each position t of the input sequence

- The next token selected, $y_s = \text{argmax}(a^{ptr}_{s})$

TECHNICAL DETAILS

- **Role of RL:**
  - Intended to address the fact that component pieces of a WHERE clause form an unordered set
  - As a result, it is possible for some generated queries to yield correct results when executed even when they are not perfect string matches with their corresponding ground truth queries



Reward Function Used:

$$R\left(q\left(y\right), q_g\right) = \begin{cases} -2, & \text{if } q\left(y\right) \text{ is not a valid SQL query} \\ -1, & \text{if } q\left(y\right) \text{ is a valid SQL query and executes to an incorrect result} \\ +1, & \text{if } q\left(y\right) \text{ is a valid SQL query and executes to the correct result} \end{cases}$$

Images: https://arxiv.org/pdf/1709.00103.pdf

TECHNICAL DETAILS

- **Resulting objective function:**
  - Model trained using gradient descent to minimize:

    $$L = L^{agg} + L^{select} + L^{where}$$

- The total gradient is the equally weighted sum of:
  - The gradient from the cross-entropy loss in predicting the SELECT column
  - The gradient from the cross-entropy loss in predicting AGG
  - The gradient from policy learning

Image and Information: https://arxiv.org/pdf/1709.00103.pdf

From question to query:

Question Input:

in what place did phil mickelson finish with a total of 282 ?

Database Table: mytable

player	country	year (s) won	total	to par	finish
Tiger Woods	United States	1999 , 2000	270	–18	1
Shaun Micheel	United States	2003	275	–13	2
Phil Mickelson	United States	2005	282	–6	T16
David Toms	United States	2001	282	–6	T16
Davis Love III	United States	1997	286	–2	T34
Rich Beem	United States	2002	291	+3	T49
Bob Tway	United States	1986	296	+8	T65

SQL Vocabulary:

SELECT WHERE FROM AND MAX MIN COUNT SUM AVG = > <

Seq2SQL Output:

SELECT finish FROM mytable WHERE total = 282 AND player = phil mickelson

Execution Result:

t16

Reinforcement Learning Reward:

Correct +1

Image: https://www.salesforce.com/blog/2017/08/salesforce-research-ai-talk-to-data.html

EXPERIMENTS: SETUP

- **Dataset:**
  - **T**he authors use a random SQL generator and Mechanical Turk to develop **WikiSQL**
  - This dataset contains natural language questions mapped to corresponding SQL queries and SQL tables extracted from HTML tables from Wikipedia

Dataset	Size	LF	Schema
WikiSQL	**80654**	**yes**	**24241**
Geoquery	880	yes	8
ATIS	5871	yes*	141
Freebase917	917	yes	81*
Overnight	26098	yes	8
WebQuestions	5810	no	2420
WikiTableQuestions	22033	no	2108

Image: https://arxiv.org/pdf/1709.00103.pdf; LF indicates whether has annotated logical forms

EXPERIMENTS: SETUP

- Example JSON blob from WikiSQL

Question, query and table ID

These files are contained in the `*.jsonl` files. A line looks like the following:


```
{
   "phase":1,
   "question":"who is the manufacturer for the order year 1998?",
   "sql":{
      "conds":[
         [
            0,
            0,
            "1998"
         ]
      ],
      "sel":1,
      "agg":0
   },
   "table_id":"1-10007452-3"
}
```


Image: https://github.com/salesforce/WikiSQL

EXPERIMENT: METRICS

- **Evaluation metrics**
  - N_{ex} : # of queries that produce correct result when evaluated
  - N_{lf} **:** # of queries that have exact string match with ground truth query
  - $Acc_{ex} = \frac{N_{ex}}{N_{lf}}$: evaluation accuracy metric
  - $Acc_{lf} = \frac{N_{lf}}{N}$ logical form accuracy metric (incorrectly penalizes queries that produce correct results but are not perfect string matches with their ground truth queries)

Model	Dev Acc_{lf}	Dev Acc_{ex}	Test Acc_{lf}	Test Acc_{ex}
Baseline (Dong & Lapata, 2016)	23.3%	37.0%	23.4%	35.9%
Aug Ptr Network	44.1%	53.8%	43.3%	53.3%
Seq2SQL (no RL)	48.2%	58.1%	47.4%	57.1%
Seq2SQL	**49.5%**	**60.8%**	**48.3%**	**59.4%**

Image: https://arxiv.org/pdf/1709.00103.pdf

🏆 but … 😕

EXPERIMENT: RESULTS

- Seq2SQL generates higher quality WHERE clauses than baseline

“in how many districts was a successor seated on march 4, 1850?”

Successor seated = seated march 4

vs.

Successor seated = seated march 4 1850

- *Seq2SQL without RL* reduces invalid queries relative to the baseline model
  - Many invalid queries come from the inclusion of column names that are not present in the table

% of generated queries that are invalid: **7.9%** **vs.** **4.8%**

Column names with multiple tokens are particularly problematic (e.g., “Miles (km)”)

- **Seq2SQL with RL** generates higher quality WHERE clauses relative to *Seq2SQL without RL;* order may differ from **ground truth**

“what is the race name of the 12th round Trenton, new jersey race where a.j. foyt had the pole position?”

WHERE rnd = 12 and track = a.j. foyt AND pole position = a.j. foyt

WHERE rnd = 12 AND pole position = a.j. foyt

DISCUSSION QUESTIONS

- What are key strengths of this approach?
- What are key weaknesses/limitations?
- How could this approach be modified to handle more complex/multi-part questions?
- Are there are other domains where applying a model capable of mapping human-interpretable input to machine-interpretable output might be beneficial?
- Are there other methods that the authors could have used in lieu of RL to handle the unordered nature of SQL WHERE clauses?

BIBLIOGRAPHY

- Donahue, Jeff & Anne Hendricks, Lisa & Guadarrama, Sergio & Rohrbach, Marcus & Venugopalan, Subhashini & Saenko, Kate & Darrell, Trevor. (2014). Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. Arxiv. PP. 10.1109/TPAMI.2016.2599174.
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://einstein.ai/research/how-to-talk-to-your-database
- https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/
- https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
- https://slideplayer.com/slide/4757729/
- Li Dong and Mirella Lapata. Language to logical form with neural attention. ACL, 2016.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15), C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 2. MIT Press, Cambridge, MA, USA, 2692-2700.
- Richard S. Sutton and Andrew G. Barto. 1998. Introduction to Reinforcement Learning (1st ed.). MIT Press, Cambridge, MA, USA.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Comput. 9, 9 (November 1997), 1735-1780. DOI=http://dx.doi.org/10.1162/neco.1997.9.8.1735
- *Victor Zhong, Caiming Xiong, and Richard Socher. 2017.* Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning.