DATA ANALYTICS USING DEEP LEARNING

GT 8803 // FALL 2018 // Sneha Venkatachalam

LECTURE #05

SQLNET: GENERATING STRUCTURED QUERIES FROM NATURAL LANGUAGE WITHOUT REINFORCEMENT LEARNING

TODAY'S PAPER

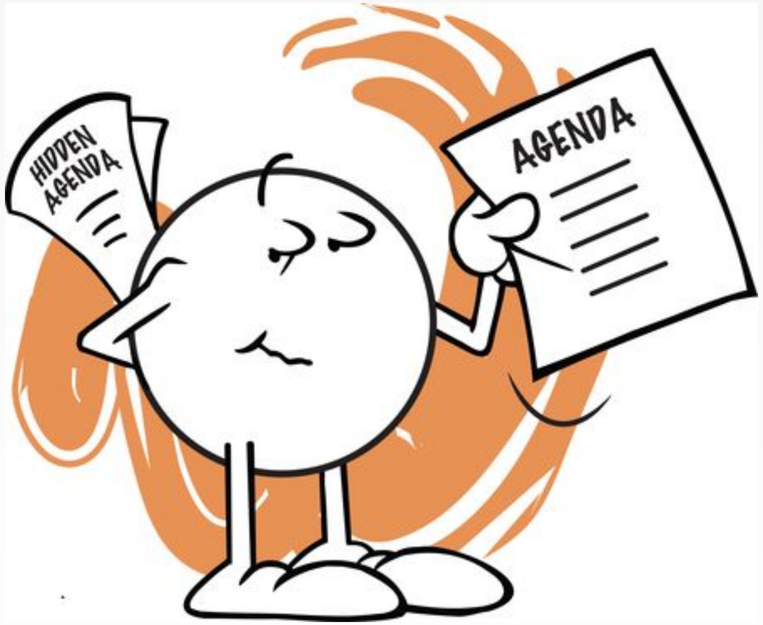
"SQLNet: Generating Structured Queries From Natural Language without using Reinforcement Learning"

- **Authors**

  Xiaojun Xu, Chang Liu, Dawn Song
- **Areas of focus**
  - SQL query synthesis
  - Natural language
  - Deep learning

TODAY'S AGENDA

- Concepts
- Problem Overview
- Key Idea
- Technical Details
- Evaluation
- Related Work
- Conclusion
- Discussion



CONCEPTS

- **Natural Language Processing**

  Analysis of raw texts and transcripts to develop algorithms to process and extract useful information

- **Word Embeddings**

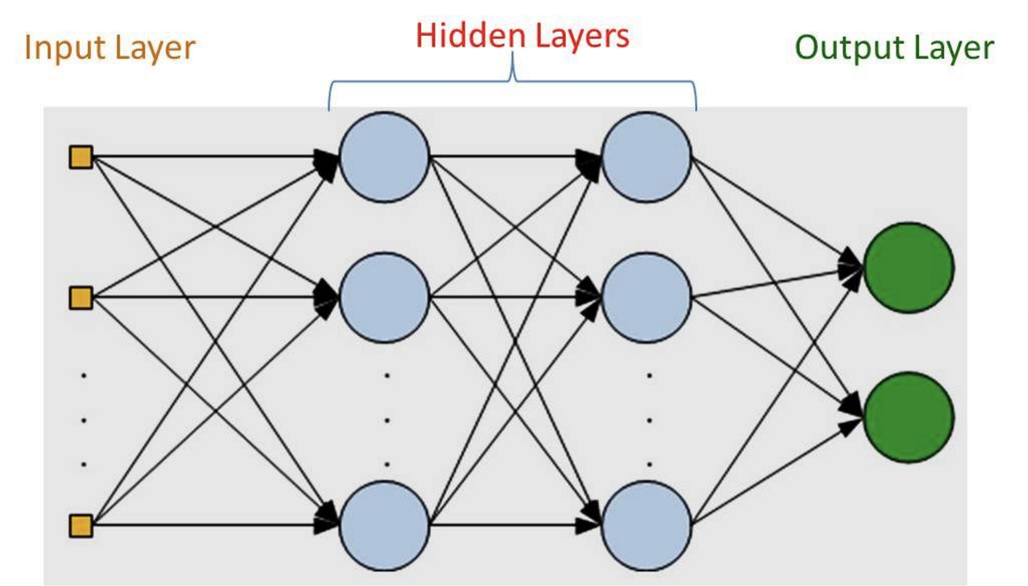
  Word embeddings are a class of techniques where individual words are represented as real-valued vectors in a predefined vector space

  Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.

CONCEPTS

- **MLP Classifier**

  A **multilayer perceptron** (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes

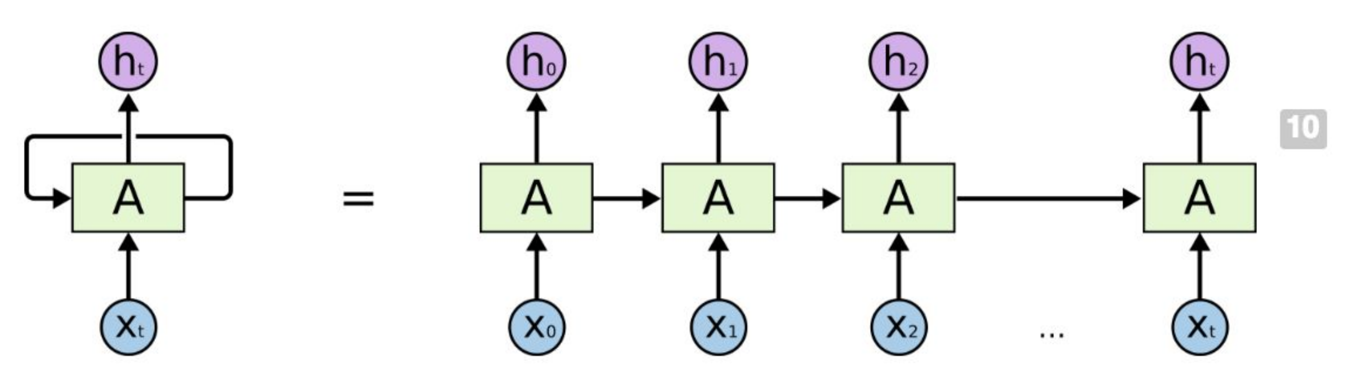


CONCEPTS

- **Recurrent Neural Networks**

  They connect previous information to the present task in a neural network

  A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor



An unrolled recurrent neural network.

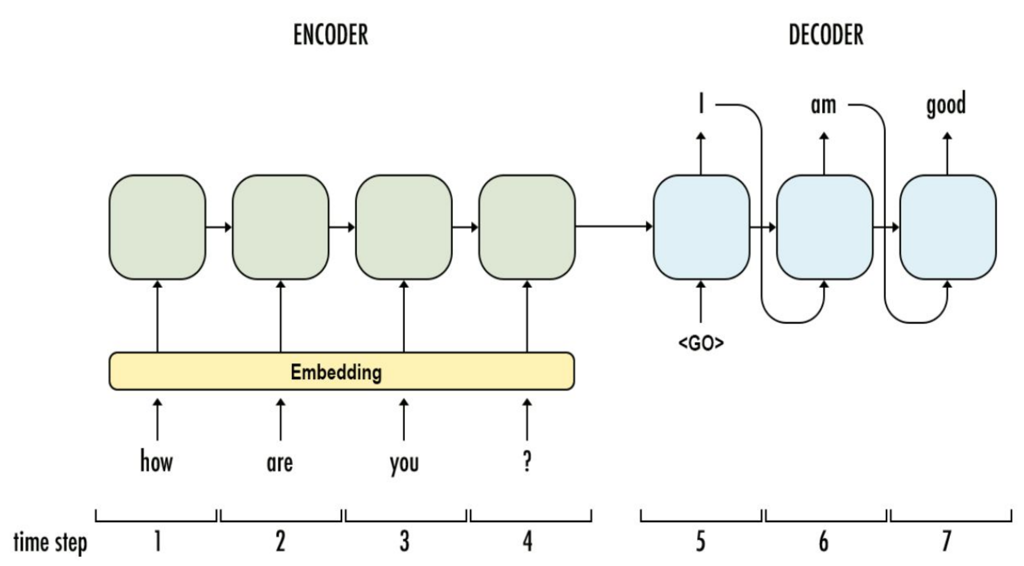
PROBLEM OVERVIEW

"Synthesizing SQL queries from natural language"

- **De facto approach**

  Sequence-to-sequence-style model
- **Problems**
  - Query serialization
  - Order matters
- **State-of-the-art**

  Uses Reinforcement learning



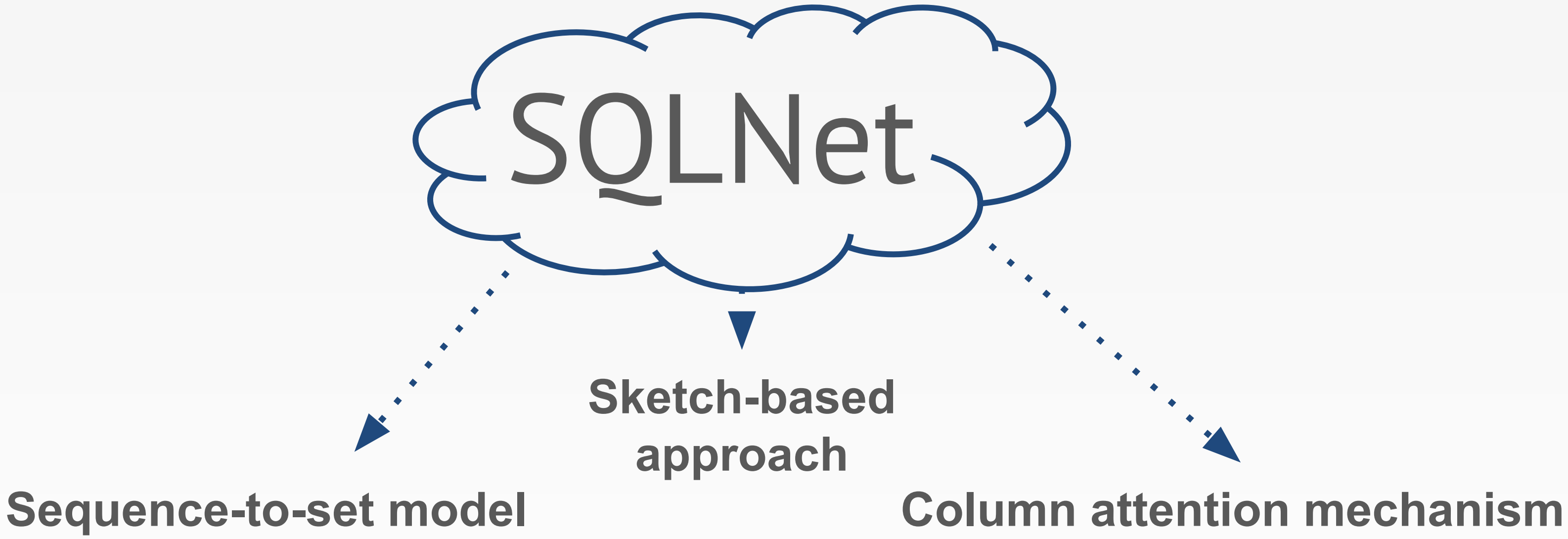
PROBLEM OVERVIEW

Ex.: How many games ended with a 1-0 score and more than 5 goals?

Query 1:	Query 2:
SELECT result	SELECT result
WHERE score=‘1-0’ AND goal=16	WHERE goal=16 AND score=‘1-0’

An example of types of different query syntax for the same task

SOLUTION


SQLNet
Sketch-based
approach
Sequence-to-set model
Column attention mechanism

KEY IDEA: SQLNET

- Novel sketch-based approach
- Avoids the "order-matters" problem
- Avoids the necessity to employ RL algorithms
- Novel column attention structure
- Achieves better results than Seq2seq approaches
- Bypasses previous state-of-the-art by 9 to 13 points on the WikiSQL dataset

KEY IDEA: WIKISQL

- Large-scale dataset for neural networks
- Employs crowd-sourcing
- Overcomes overfitting
- Mitigates the scalability and privacy issues
- Synthesizes query without requiring table’s content
- Training, dev, and test set do not share tables
- Helps evaluate generalization to unseen schema.

KEY IDEA: WIKISQL

- **Input**
  - A natural language question
  - Table schema
    - Name of each column
    - Column type (i.e., real numbers or strings)
- **Output**
  - SQL query

KEY IDEA: WIKISQL

Table

Player	No.	Nationality	Position	Years in Toronto	School/Club Team
Antonio Lang	21	United States	Guard-Forward	1999-2000	Duke
Voshon Lenard	2	United States	Guard	2002-03	Minnesota
Martin Lewis	32, 44	United States	Guard-Forward	1996-97	Butler CC (KS)
Brad Lohaus	33	United States	Forward-Center	1996	Iowa
Art Long	42	United States	Forward-Center	2002-03	Cincinnati

Question:

Who is the player that wears number 42?

SQL:

```
SELECT player
WHERE no. = 42
```

Result:

Art Long

An example of the WikiSQL task

KEY IDEA: SKETCH

- **SQL keywords (Tokens in bold)**
  - SELECT, WHERE, and AND
- **Slots (Tokens starting with "$")**
  - $AGG: empty, SUM or MAX
  - $COLUMN: column name
  - $VALUE: substring of the question
  - $OP: {=, <, >}
- **Regex Notion (...)***
  - Indicates 0 or more AND clauses.

KEY IDEA: SKETCH

,

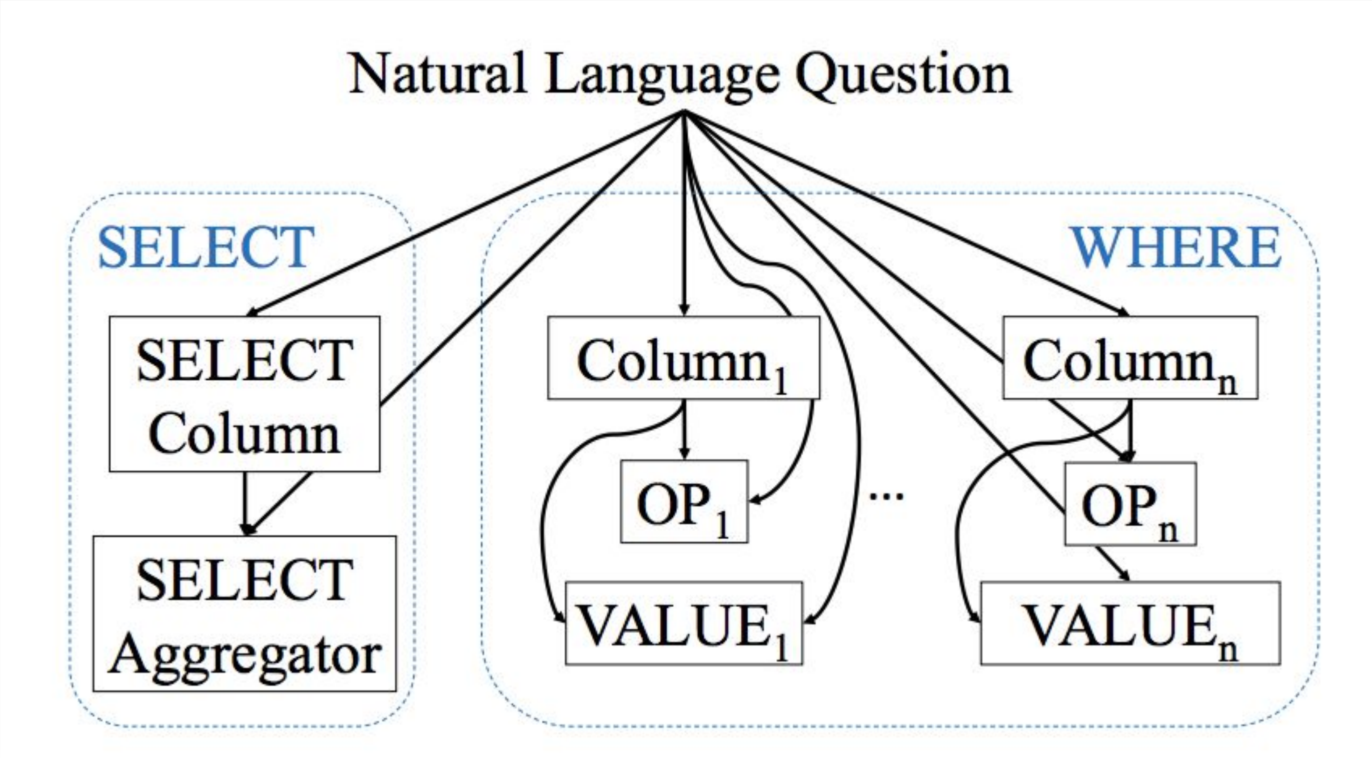
```
SELECT $AGG $COLUMN
WHERE $COLUMN $OP $VALUE
(AND $COLUMN $OP $VALUE)*
```

SQL Sketch

KEY IDEA: DEPENDENCY GRAPH

- Slots depicted by boxes
- Dependency is depicted as a directed edge.
- Independent prediction of constraints
- Helps avoid the “order-matters” problem in a sequence-to-sequence model

KEY IDEA: DEPENDENCY GRAPH



Graphical illustration of the dependency in a sketch

TECHNICAL DETAILS: SEQ2SET

- To determine the most probable columns in a query
- Column names appearing in the WHERE clause constitute a subset of all column names
- Can simply predict which column names appear in this subset of interest
- Can be viewed as a MLP with one layer over the embeddings computed by 2 LSTMs (one for the question, one for the column names)

$$P_{\textbf{wherecol}}(col|Q) = \sigma(u_c^T E_{col} + u_q^T E_Q)$$

- uc and uq are two column vectors of trainable variables

TECHNICAL DETAILS: COLUMN ATTENTION

- E_Q may not be able to remember information used to useful in predicting a particular column name
- **Ex.:**
  - Token "number" is more relevant to predicting the column "No." in the WHERE clause.
  - However, the token "player" is more relevant to predicting the "player" column in the SELECT clause
- Computes an attention mechanism between tokens

$$E_{Q|col} = H_Q w$$

- H_Q is a matrix of d×L, where L is the length of the natural language question.

TECHNICAL DETAILS: COLUMN ATTENTION

- w is a L-dimension column vector, computed by

$$w = \mathbf{softmax}(v) \qquad v_i = (E_{col})^T W H_Q^i \quad \forall i \in \{1, ..., L\}$$

- W is a trainable matrix of size d × d
- H^i_Q indicates the i-th column of H_Q

- The final model for predicting column names in the WHERE clause

$$P_{\mathbf{wherecol}}(col|Q) = \sigma((u_a^{col})^T \mathbf{tanh}(U_c^{col} E_{col} + U_q^{col} E_{Q|col}))$$

- U^{col}_c and U^{col}_q are trainable matrices of size d × d, and u^{col}_a is a d-dimensional trainable vector

TECHNICAL DETAILS: WHERE CLAUSE

- **Column slots:** Use a MLP over P(col|Q) to decide no. of columns and choose column in descending order of P(col|Q)

$$P_{\#\mathbf{col}}(K|Q) = \mathbf{softmax}(U_1^{\#\mathrm{col}}\mathbf{tanh}(U_2^{\#\mathrm{col}}E_{Q|Q}))_i$$

- **OP slot:** Use a MLP to pick the most probable operator (=, <, >)

$$P_{\mathrm{op}}(i|Q, col) = \mathbf{softmax}(U_1^{\mathrm{op}}\mathbf{tanh}(U_c^{op}E_{col} + U_q^{op}E_{Q|col}))_i$$

- **VALUE slot:** Uses a copy/pointer SEQ2SEQ to predict a substring from the input question token, order matters here

$$P_{\mathrm{val}}(i|Q, col, h) = \mathbf{softmax}(a(h))$$

$$a(h)_i = (u^{\mathrm{val}})^T\mathbf{tanh}(U_1^{\mathrm{val}}H_Q^i + U_2^{\mathrm{val}}E_{col} + U_3^{\mathrm{val}}h) \quad \forall i \in \{1, ..., L\}$$

TECHNICAL DETAILS: SELECT CLAUSE

- Only one column is picked, similar to prediction of columns in WHERE clause

$$P_{\textbf{selcol}}(i|Q) = \textbf{softmax}(sel)_i$$

$$sel_i = (u_a^{\text{sel}})^T \textbf{tanh}(U_c^{\text{sel}} E_{col_i} + U_q^{\text{sel}} E_{Q|col_i}) \quad \forall i \in \{1, ..., C\}$$

  – u^{sel}_a, U^{sel}_c, U^{sel}_q are similar to u^{col}_a, U^{col}_c, U^{col}_q
- Aggregation operator selected using a MLP

$$P_{\textbf{agg}}(i|Q, col) = \textbf{softmax}(U^{\text{agg}} \textbf{tanh}(U_a E_{Q|col}))_i$$

,

TECHNICAL DETAILS: TRAINING

- **Input encoding model details**
  - Natural language descriptions and column names treated as a sequence of tokens
  - Stanford CoreNLP tokenizer used to to parse sentences
- **Training details**

$$loss(col, Q, y) = -\left(\sum_{j=1}^{C} (\alpha y_j \log P_{\mathbf{wherecol}}(col_j|Q) + (1 - y_j) \log(1 - P_{\mathbf{wherecol}}(col_j|Q)) \right)$$

(Assume y is a C-dimensional vector where yj = 1 indicates j-th column appears in the ground truth of WHERE; and yj = 0 otherwise)

  - Weighted cross-entropy loss for other sub-models

TECHNICAL DETAILS: TRAINING

- **Weight sharing details**
  - Multiple LSTMs for predicting different slots
  - Shared word embeddings among different models, however different LSTM weights
- **Training the word embedding**
  - GloVe embeddings used
  - Updated during training

CONCEPT: **GloVe**, coined from Global Vectors, is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words.

,

EVALUATION: SETUP

"SQLNet versus Seq2SQL"

- **Dataset**

  WikiSQL
- **Technology**

  PyTorch
- **Evaluation metrics**
  - Logical-form accuracy
  - Query-match accuracy
  - Execution accuracy

EVALUATION: RESULTS

	dev			test		
	$\mathrm{Acc}_{\mathrm{agg}}$	$\mathrm{Acc}_{\mathrm{sel}}$	$\mathrm{Acc}_{\mathrm{where}}$	$\mathrm{Acc}_{\mathrm{agg}}$	$\mathrm{Acc}_{\mathrm{sel}}$	$\mathrm{Acc}_{\mathrm{where}}$
Seq2SQL (ours)	90.0%	89.6%	62.1%	90.1%	88.9%	60.2%
Seq2SQL (ours, C-order)	-	-	63.3%	-	-	61.2%
SQLNet (Seq2set)	-	-	69.1%	-	-	67.1%
SQLNet (Seq2set+CA)	**90.1%**	91.1%	72.1%	**90.3%**	90.4%	70.0%
SQLNet (Seq2set+CA+WE)	**90.1%**	**91.5%**	**74.1%**	**90.3%**	**90.9%**	**71.9%**

EVALUATION: RESULTS

- **Seq2SQL (C-order)** indicates that after Seq2SQL generates the WHERE clause, we convert both the prediction and the ground truth into a canonical order when being compared
- **Seq2set** indicates sequence-to-set technique
- **+CA** indicates column attention is used
- **+WE** indicates word embedding is allowed to be trained
- $\mathbf{Acc}_{\mathbf{agg}}$ and $\mathbf{Acc}_{\mathbf{sel}}$ indicate the accuracy on the aggregator and column prediction accuracy on the SELECT clause
- $\mathbf{Acc}_{\mathbf{where}}$ indicates the accuracy to generate the WHERE clause.

EVALUATION: BREAK-DOWN

- SELECT clause prediction accuracy is around 90%, less challenging than WHERE
- 11-12 points improvement of WHERE clause accuracy over Seq2SQL
- Improvement from using Sequence-to-set architecture is around 6 points
- The column attention further improves a sequence-to-set only model by 3 points
- Allowing training word embedding gives another 2 points' improvement
- Improvements from two clauses add to 14 points total

EVALUATION - WIKISQL VARIANT

- In practice, often when a model is trained, the table in the test set is already seen in the training set
- To mimic this,
  - Data reshuffling
  - All the tables appear at least once in the training set
- **Improved results**

	dev			test		
	Acc_{lf}	Acc_{qm}	Acc_{ex}	Acc_{lf}	Acc_{qm}	Acc_{ex}
Seq2SQL (ours)	54.5%	55.6%	63.8%	54.8%	55.6%	63.9%
SQLNet	-	**65.5%**	**71.5%**	-	**64.4%**	**70.3%**

RELATED WORK

- **Warren & Pereira, 1982; Androutsopoulos et al., 1993; 1995; Popescu et al., 2003; 2004; Li et al., 2006; Giordani & Moschitti, 2012; Zhang & Sun, 2013; Li & Jagadish, 2014; Wang et al., 2017**
  - Earlier work focuses on specific databases
  - Requires additional customization to generalize to each new database
- **Li & Jagadish, 2014; Iyer et al., 2017**

  Incorporates users' guidance

RELATED WORK

- **Pasupat & Liang, 2015; Mou et al., 2016**
  – Incorporates the data in the table as an additional input
  – Scalability and privacy issues
- **Yaghmazadeh et al., 2017**
  – Sketch-based approach
  – Relies on an off-the-shelf semantic parser for natural language translation
  – Employs programming language techniques to iteratively refine the sketch into the final query

RELATED WORK

- **Zhong et al., 2017**
  - Overcoming the inefficiency of a Seq2seq model (RL)
- **Zelle & Mooney, 1996; Wong & Mooney, 2007; Zettlemoyer & Collins, 2007; 2012; Artzi & Zettlemoyer, 2011; 2013; Cai & Yates, 2013; Reddy et al., 2014; Liang et al., 2011; Quirk et al., 2015; Chen et al., 2016**
  - Parse a natural language to SQL queries in logical form
  - Most need to be fine-tuned to the specific domain of interest, may not generalize

CONCLUSION

- Overcomes the ‘order matters’ problem
- Sketch-based approach using dependency graph
- Column attention introduced
- Improves over Seq2SQL on WikiSQL task by 9-13 points

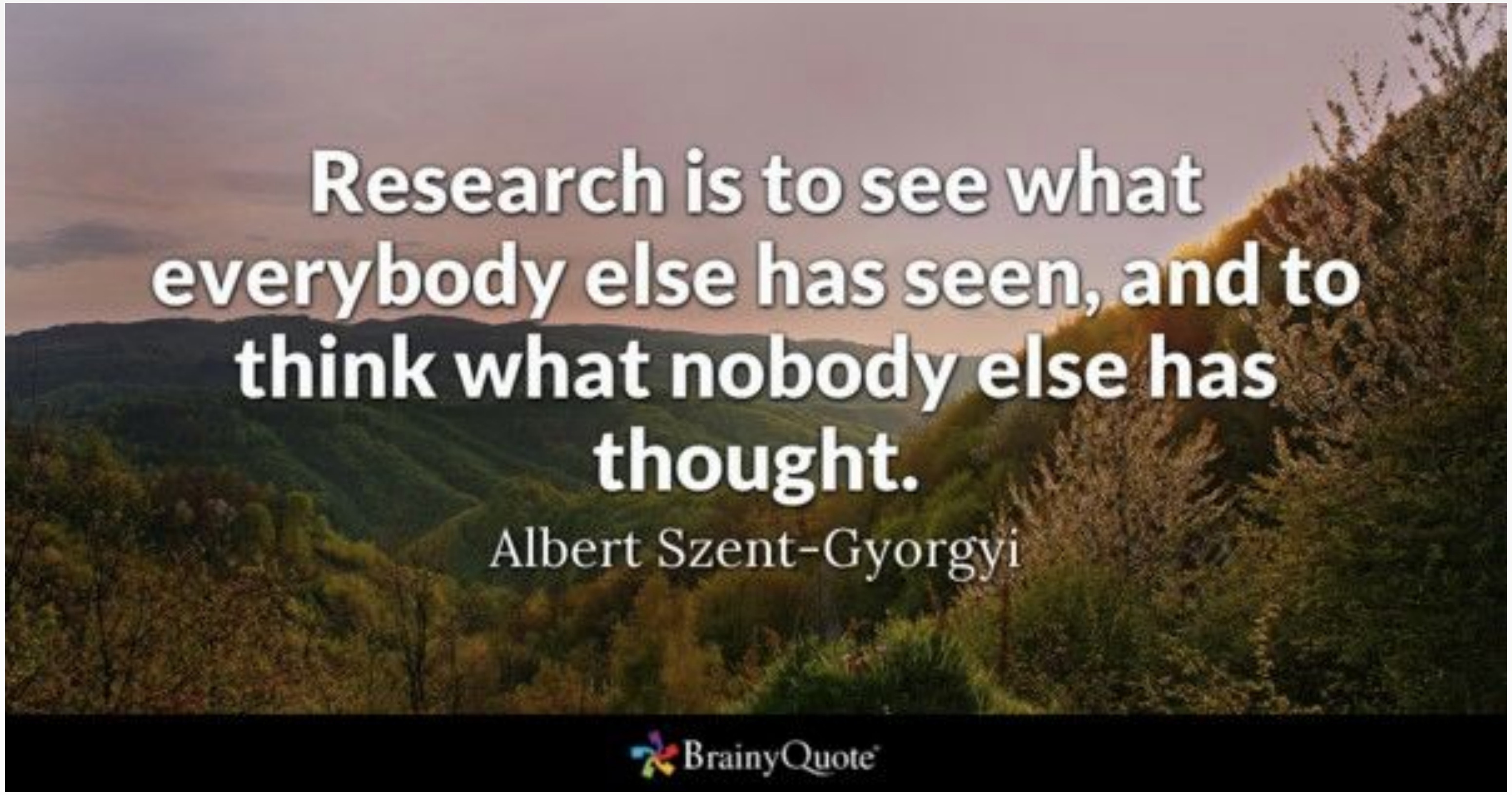
’

QUESTIONS OR COMMENTS?

DISCUSSION

- Dataset used makes very strong simplification assumptions (that every token is an SQL keyword or appears in the NL)
- Not a very challenging SQL dataset
- Is the 'order' issue principally a problem for the Seq2seq model? (Order can be corrected)
- Set prediction approach is not novel
- Sketch-based approach is limited and non-scalable
  - Need for re-constructing SQL query based on grammar pre-defined by the sketch for new type of query

THANK YOU!


Research is to see what everybody else has seen, and to think what nobody else has thought.
Albert Szent-Gyorgyi
BrainyQuote