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.
• **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

![Diagram of an unrolled recurrent neural network](image-url)
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:
SELECT result
WHERE score='1-0' AND goal=16

Query 2:
SELECT result
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

<table>
<thead>
<tr>
<th>Player</th>
<th>No.</th>
<th>Nationality</th>
<th>Position</th>
<th>Years in Toronto</th>
<th>School/Club Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voshon Lenard</td>
<td>2</td>
<td>United States</td>
<td>Guard</td>
<td>2002-03</td>
<td>Minnesota</td>
</tr>
<tr>
<td>Martin Lewis</td>
<td>32,44</td>
<td>United States</td>
<td>Guard-Forward</td>
<td>1996-97</td>
<td>Butler CC (KS)</td>
</tr>
<tr>
<td>Brad Lohaus</td>
<td>33</td>
<td>United States</td>
<td>Forward-Center</td>
<td>1996</td>
<td>Iowa</td>
</tr>
<tr>
<td>Art Long</td>
<td>42</td>
<td>United States</td>
<td>Forward-Center</td>
<td>2002-03</td>
<td>Cincinnati</td>
</tr>
</tbody>
</table>

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)

\[ P_{\text{wherecol}}(col|Q) = \sigma(u_c^T E_{col} + u_q^T E_Q) \]

- \(u_c\) and \(u_q\) 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

\[ E_{Q|col} = H_Qw \]

- $H_Q$ is a matrix of $d \times L$, where $L$ is the length of the natural language question.
• $w$ is a $L$-dimension column vector, computed by

\[
w = \text{softmax}(v) \quad v_i = (E_{col})^T WH_i^Q \quad \forall i \in \{1, \ldots, L\}
\]

- $W$ is a trainable matrix of size $d \times d$
- $H_i^Q$ indicates the $i$-th column of $H_Q$

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

\[
P_{\text{where}col}(\text{col}|Q) = \sigma((u^\text{col}_a)^T \tanh(U^\text{col}_c E_{col} + U^\text{col}_q E_Q|\text{col}))
\]

- $U^\text{col}_c$ and $U^\text{col}_q$ are trainable matrices of size $d \times d$, and $u^\text{col}_a$ is a $d$-dimensional trainable vector
TECHNICAL DETAILS: WHERE CLAUSE

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

$$P_{\#\text{col}}(K|Q) = \text{softmax}(U_{1\#\text{col}} \tanh(U_{2\#\text{col}} E_{Q|Q}))_i$$

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

$$P_{\text{op}}(i|Q, \text{col}) = \text{softmax}(U_{1\text{op}} \tanh(U_{c\text{op}} E_{\text{col}} + U_{q\text{op}} E_{Q|\text{col}}))_i$$

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

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

$$a(h)_i = (u_{\text{val}})^T \tanh(U_{1\text{val}} H^i_{Q} + U_{2\text{val}} E_{\text{col}} + U_{3\text{val}} h) \quad \forall i \in \{1, ..., L\}$$
TECHNICAL DETAILS: SELECT CLAUSE

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

\[ P_{sel|col}(i|Q) = \text{softmax}(sel)_i \]

\[ sel_i = (u^\text{sel}_a)^T \tanh(U^\text{sel}_c E_{col_i} + U^\text{sel}_q E_{Q|col_i}) \quad \forall i \in \{1, ..., C\} \]

- \( u^\text{sel}_a, u^\text{sel}_c, u^\text{sel}_q \) are similar to \( u^\text{col}_a, u^\text{col}_c, u^\text{col}_q \)

- Aggregation operator selected using a MLP

\[ P_{agg}(i|Q, col) = \text{softmax}(U^\text{agg} \tanh(U_a E_{Q|col}))_i \]
TECHNICAL DETAILS: TRAINING

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

• Training details

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

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

<table>
<thead>
<tr>
<th></th>
<th>dev</th>
<th></th>
<th>test</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{Acc}_{\text{agg}}$</td>
<td>$\text{Acc}_{\text{sel}}$</td>
<td>$\text{Acc}_{\text{where}}$</td>
<td>$\text{Acc}_{\text{agg}}$</td>
</tr>
<tr>
<td>Seq2SQL (ours)</td>
<td>90.0%</td>
<td>89.6%</td>
<td>62.1%</td>
<td>90.1%</td>
</tr>
<tr>
<td>Seq2SQL (ours, C-order)</td>
<td>-</td>
<td>-</td>
<td>63.3%</td>
<td>-</td>
</tr>
<tr>
<td>SQLNet (Seq2set)</td>
<td>-</td>
<td>-</td>
<td>69.1%</td>
<td>-</td>
</tr>
<tr>
<td>SQLNet (Seq2set+CA)</td>
<td>90.1%</td>
<td>91.1%</td>
<td>72.1%</td>
<td>90.3%</td>
</tr>
<tr>
<td>SQLNet (Seq2set+CA+WE)</td>
<td>90.1%</td>
<td>91.5%</td>
<td>74.1%</td>
<td>90.3%</td>
</tr>
</tbody>
</table>
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.
- **Acc\textsubscript{agg}** and **Acc\textsubscript{sel}** indicate the accuracy on the aggregator and column prediction accuracy on the SELECT clause.
- **Acc\textsubscript{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

<table>
<thead>
<tr>
<th></th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc_{lf}</td>
<td>Acc_{qm}</td>
</tr>
<tr>
<td>Seq2SQL (ours)</td>
<td>54.5%</td>
<td>55.6%</td>
</tr>
<tr>
<td>SQLNet</td>
<td>-</td>
<td>65.5%</td>
</tr>
</tbody>
</table>
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
Research is to see what everybody else has seen, and to think what nobody else has thought.

Albert Szent-Gyorgyi