Generating SQL from Complex Natural Language Questions

James Ma, Tao Yu, and Dragomir Radev

Department of Computer Science, Yale School of Engineering and Applied Sciences, New Haven, CT

LILY Lab

Introduction

One of the most interesting challenges currently studied in NLP and AI is how to convert natural language into structured, executable programs (NL2SQL). This movement is primarily motivated by the enormous and growing amounts of data that are stored in databases today, which are largely inaccessible to those without knowledge of querying languages. SQLNet is the current state-of-the-art NL2SQL model for the WikiSQL task, in which both natural language questions and their corresponding target queries are simple. My project involves two parts: 1) the proposal of a new NL2SQL task which involves question/query pairs that are much more complex and 2) baseline performance of an extension of the SQLNet model to this new task.

Materials and Methods

This project proposes a new dataset in the domain of NL2SQL prediction, which is currently being put together by myself and other members of LILY. Unlike WikiSQL, this dataset contains thousands of complex natural language questions that query from hundreds of different databases, each of which contain multiple tables. Target queries are also much more realistic, as they include joins, GROUP BY, ORDER BY, and even nested subqueries. My project extends methodologies proposed in SQLNet for predicting WHERE conditions towards the prediction of multiple SELECT columns and AGG operators, as well as GROUP BY and ORDER BY components, in order to establish baseline performance metrics for this new task. This entails using bidirectional LSTMs to encode input information and compute neural attention scores for each component. Accuracy measures are reported both for each component separately as well as for the query as a whole, using query matching of predictions against target queries. The model was run for 300 epochs with a learning rate of 0.001, and individual components were optimized using Cross Entropy loss.

Figure 1. An example of a query in the new dataset of similar difficulty to WikiSQL.

Q:
Find the names of all sculptures located in gallery 226.

SQL:

```
SELECT title
FROM sculptures
WHERE location = “Gallery 226”
```

Figure 2. An example of a query in the new dataset that is much more complicated.

Q:
What are the first and last names of the artists who have not only oil paintings but also paintings in the lithographic medium?

SQL:

```
SELECT T1.lname, T1.fname
FROM artists AS T1 JOIN paintings AS T2
    ON T1.artistID = T2.painterID
WHERE T2.medium = “oil”
INTERSECT
SELECT T1.lname, T1.fname
FROM artists AS T1 JOIN paintings AS T2
    ON T1.artistID = T2.painterID
WHERE T2.medium = “lithograph”
```

Figure 3. Component accuracy for the model on the new dataset for training and testing sets.

Component Accuracy on Full Dataset		
	Train Acc	Test Acc
SELECT #	0.634	0.663
SELECT Cols	0.404	0.248
AGG #	0.998	0.725
AGG Ops	0.988	0.731
WHERE #	0.935	0.763
WHERE Cols	0.850	0.699
WHERE Ops	1.000	0.994
WHERE Conds	0.759	0.456
GROUP #	0.736	0.762
GROUP Cols	0.736	0.734
ORDER #	0.857	0.812
ORDER Cols	0.982	0.971
ORDER Aggs	1.000	1.000
ORDER Par	0.833	0.788

Figure 4. Query matching accuracy for the model on the new dataset for training and testing.

Query Matching Accuracy on Full Dataset	
	QM Accuracy
Train	0.219
Test	0.104

Results

As we can see, the model is able to get a pretty high training fit for most components - in particular the AGG component. It is important to note that training accuracies are slighly inflated, as for each component that depends on another (for example, AGG depends closely on SELECT), target labels are provided to the dependent prediction to help the model learn each component without requiring that the previous step be correct. Not surprisingly, the testing accuracy metrics are generally lower, likely due in part to this reason. In particular, it is interesting that the model is unable to attain a strong fit for SELECT column prediction. One possible explanation for the poor performance on SELECT column prediction is that the current column names stored in the dataset are not clean. For example, tables often include column names like `stuid` that should really be `student id`, or `lname` instead of `last name`. As a result, it becomes much harder for the model to correctly predict these columns as they do not have GloVe embeddings. Query matching accuracy is rather low at ~22%, but is unsurprising given the complexity of the dataset, and that subquery and several other minor prediction components are not implemented.

Conclusion

This project had two main components. First, a new dataset was proposed in the domain of NL2SQL research, which involves question/query pairs that are far more complex and realistic than those found in WikiSQL. Second, baseline performance on this new NL2SQL task was evaluated for a new model, which extends ideas from SQLNet to this new problem. While results were not stellar, this was merely a baseline model which used methods tailored to a simpler task. It is clear from this that in order to truly solve the NL2SQL task, new and more sophisticated model architectures that are tailored towards this problem setting will need to be proposed.

Acknowledgement

I would like to thank Prof. Dragomir Radev for graciously agreeing to be my advisor for this project. I would also like to thank Tao Yu for his guidance and mentorship throughout the past semester.