Project Progress Report
Semantic Parsing Natural Language into SQL via Deep Learning

Ruiyang Xu, Ayush Singh and Xun Peng
{xu.r, singh.ay, peng.xu}@husky.neu.edu
CS6120 NLP Fall 2017

1 Changes

We have changed our project title. Considering the nature of our dataset, it is more achievable to train a parser from natural language to SQL instead of relational algebra. And, more specifically, we would like to implement this parser via state of the art deep learning technologies (will describe it in detail in the Method section).

2 Preprocessing

We are using the WikiSQL dataset, provided by Salesforce. However, since the original dataset is serialized for storage. We have found a way to deserialize the dataset and use them as our training and test data. The dataset itself is very clean, and there is not so much preprocessing work to do for the time being. But, considering our future work (will describe in detail in the Future Work section), we would like to add semantic annotations to the raw dataset and see if those annotations will help the neural network to learn better. We plan to extract those semantic annotations via the Stanford CoreNLP toolkit.

Furthermore, considering the input of our neural network model, we indexed and embedded all the words in the dataset with an embedding layer with size 300. This allows us to use the sequence to sequence model to work on a sequence of words.

3 Method

We chose to improve a sequence to sequence model largely because the nature of a parser is learning a sequence to sequence probabilistic model and a deep neural network is good at doing probabilistic inferences. But for the time being, we just finished the implementation of our baseline model.

Our baseline model is an LSTM based vanilla sequence to sequence model (Sutskever et al.,). It is pretty much the same model from Tensorflow with some modifications to suit our dataset. Specifically, instead of predicting single letters, we treat a whole word as an atom.

The structure of this model is pretty simple: it is a two layer encoder-decoder LSTM neural network (Cho et al., 2014), with 50 hidden states (size of LSTM) in each layer. We also applied gradient clipping to prevent overshooting during gradient descent.

4 Result and Evaluation

For the time being, we have got our baseline output and, amazingly, even such a preliminary model can still generate some valid SQL. However, most of generated SQL is irrelevant or illegal. Here are some samples from our baseline model:

```
Who is the player that wears number 42?
Ground Truth:
SELECT Player FROM table WHERE No. = 42
Generated:
SELECT Airport FROM table WHERE Name = r
AND Year = 1

How many winning drivers were the for the
rnd equalling 5?
Ground Truth:
SELECT COUNT Winning driver FROM table
WHERE Rnd = 5
Generated:
SELECT COUNT Area km 2 FROM table WHERE
Title = united states AND Player
= frank nobilo

What is the exit date for the Dutch Albums
Top 100 Chart?
Ground Truth:
SELECT Date of Exit FROM table WHERE
Chart = Dutch Albums Top 100
Generated:
SELECT COUNT Manner FROM table WHERE
Away team score = north melbourne

Which Allied Force targetted Woensdrecht?
Ground Truth:
SELECT Allied forces FROM table WHERE
Target = Woensdrecht
Generated:
SELECT MIN Sexual (billion $) FROM table WHERE
Name = united states AND Total > 1

Name the losing bonus for 27
Ground Truth:
SELECT Played FROM table WHERE Tries for = 27
Generated:
SELECT COUNT Year FROM table WHERE
Title = victoria park

How many districts had an election of charles
brand (r) 67.2%  and h. e. rice (d) 32.8%?
Ground Truth:
SELECT COUNT District FROM table WHERE
Candidates = Charles Brand (R) 67.2% H. E.
Rice (D) 32.8%
Generated:
SELECT COUNT Attendance FROM table WHERE
Country = united states AND Total = 2
```

To evaluate our result, since two queries can be equivalent even when they are not string matching with each other, we plan to use Cosette (Chu et al., 2017), a tool to check SQL query equivalence. We are still working on dumping the output and load it into Cosette using an automated evaluation pipeline. But for the time being, even without certain evaluation metric, one still can see there is something obviously wrong with the vanilla sequence to sequence model.

5 What is working and What is wrong

Obviously, the baseline model can already capture the basic grammar of output SQL queries. However, most of the time, the output is irrelevant with the input (see the examples above). The problem is within the structure of this vanilla model: the information flow in this neural network is linear, and each output unit all has a dependency on all the input units. This structure tremendously limits the ability of the neural network to generalize. In the end, the neural network converges very fast (validation loss stop dropping after 2 to 3 epochs) to local optimal, and these local optimal are usually some existing queries in the training example.

Therefore, on one hand, we did see that a sequence to sequence model might be working on this task. But, on the other hand, we definitely think some structure improvement should be applied to this vanilla model before it generates something meaningful.

6 Future work

In the latter phase of this project, we are going to implement an attention mechanism (Bahdanau et al., 2016), especially the Pointer Network (Vinyals et al.,) on the basic sequence to sequence model. We think attention mechanism is the core to a parsing task. Another component that we think would drastically improve performance would be instead of learning word2vec ourselves, using a pretrained word2vec on 600B tokens will let the network expand its context on the meaning of incoming sentences.

As we have mentioned before, the problem with a conventional sequence to sequence model is that all input information has been used to infer output. However, in practical, humans can always find that a particular section of input corresponds to a particular section of output but not the others. This fact indicates that inference of words in output might not need to use all information in the input. A pointer network tries to weight those input words and find the most possible correspondence between pairs of words in input and output.

Our current work is pretty much based on the Salesforce's (Zhong et al., 2017) effort of an augmented pointer network. The Salesforce people improve the accuracy by only adding additional database schema information in the input data (such as table name and column name). Therefore we were thinking of adding more specific semantic information (like POS tag, entity categories or semantic dependences) and hoping that effort could further improve the result.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Cho, K., Bahdanau, D., and Bengio, Y. (2014). Learning phrase representation using rnn encoder-decoder for statistical machine translation.

Chu, S., Weitz, K., Cheung, A., and Suciu, D. (2017). Hottsql: Proving query rewrites with univalent sql semantics.

Sutskever, P., Vinyals, O., and V.le, Q. Sequence to sequence learning with neural networks.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.