Literature Review - Week 3

Xiaopeng Lu
PID: 730095318
Research Advisor: Dr. Jaime Arguello

Seq2SQL: Generating structured queries from natural language using reinforcement learning [1]

For the research question "Increasing accuracy of the correct conversion from natural language to Structured Query Language", the first paper I have read was the baseline paper: "Seq2SQL: Generating structured queries from natural language using reinforcement learning"[1]. This paper has two major contributions: first, one fundamental training/test database – WikiSQL[1] – was released for further research purpose. This database is the biggest hand-annotated database in this research area so far: it has 80654 queries with corresponding natural language annotations across 24241 tables. Second, **Seq2SQL** algorithm was introduced. Reinforcement learning (instead of seq2seq attention[2]) were used to build a baseline model. They produced a baseline test accuracy of 59.4% using WikiSQL datasets.

The Seq2SQL model consists of three parts: Aggregation Classifier, SELECT column pointer, and WHERE clause pointer decoder.[1] The biggest change of this algorithm compared with former ones is that this paper utilized reinforcement learning for the third part of the model. The reason is that for "WHERE" clause, there might be more than one query that can yield to the correct output. As a result, cross entropy loss function would not work perfectly here. Therefore, the author defined the reward and further trained this part with "loss equals to negative expected reward over possible WHERE clauses."[1]

Reinforcement learning and pointer network are two parts I did not know much about before. As a result, I spent the rest of the time this week in reading and doing hands on practice in reinforcement learning field.

Language to Logical Form with Neural Attention[2]

This paper was published before the first one. Though the result was not state-of-art right now, it paved the way for the later papers in the semantic parsing area.

Two things that this paper focused on was semantic parsing and encoder-decoder architecture. The biggest advantage using seq2seq model for semantic parsing is that it does not need "predefined templates and manually designed features" [2], one thing that traditional methods heavily depends on.

One method that the author mentioned is **Attention** mechanism. This is a commonly used method to integrate the input information into every time step in order to better predict every token [2].

The biggest contribution of this paper is that it created a **Seq2Tree** model as an improvement of seq2seq. The general idea of seq2tree is that for hierarchical structures appeared in the natural language input, one "nonterminal" $\langle n \rangle$ token will be generated to mark that here is a second level. As a result, the probability of sub-level tokens will be conditioned on the last top-level hidden vectors[2]. As a result, the final probability of the output sequence q given input a will be defined by[2]:

$$p(a|q) = p(y1y2y3y4|q)p(y5y6|y \leq 3, q)$$

$y5$ and $y6$ here are the sub-level tokens derived from $y3$. Therefore, the probability of this should be conditioned by not only the whole input q but also the hidden vectors before it. Experiment upon several datasets show the effectiveness and advantage of this Seq2Tree algorithm.

References

[1] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

[2] Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

Literature Review - Week 4

Xiaopeng Lu
PID: 730095318
Research Advisor: Dr. Jaime Arguello

Learning Phrase Representations using RNN EncoderDecoder for Statistical Machine Translation[1]

This is the first paper introducing encoder-decoder models. The general idea is to encode a variable length sequence to a **fixed length vector**. After that, another RNN is used to decode the generated fixed length vector back into variable length sequences. [2]

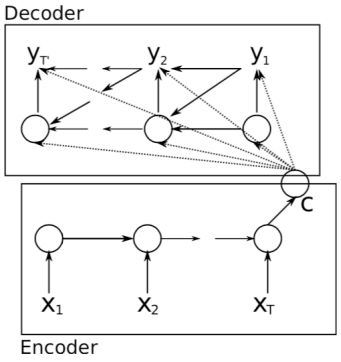
The encoder reads $X_1, X_2, .., X_T$ sequentially and C is the final hidden state. As a result, we can regard C as a fixed length vector saving all sequential information from the input sequence.

After that, another RNN is used to decode this fixed length vector, and each symbol y_T is predicted by both previous hidden state $h_{<t>}$. The difference between this decoder and traditional simple RNN method is that the the hidden state $h_{<t>}$ here is also conditioned on the fix length vector C. In other words, the hidden state $h_{<t>}$ is determined by:

$$h_{<t>} = f(h_{<t-1>}, y_{t-1}, C)$$

Intuitively, this encoder-docoder model is trained to maximize the conditional log-likelihood:

$$\max_{\theta} \sum_{n=1}^{N} \log p_{\theta}(y_n|x_n)$$



This model is fundamental because it is the baseline approach for recent machine translation related research. Since nl2sql task is a special form of machine translation, it is reasonable to believe that encoder-decoder network construction is a good place to start with.

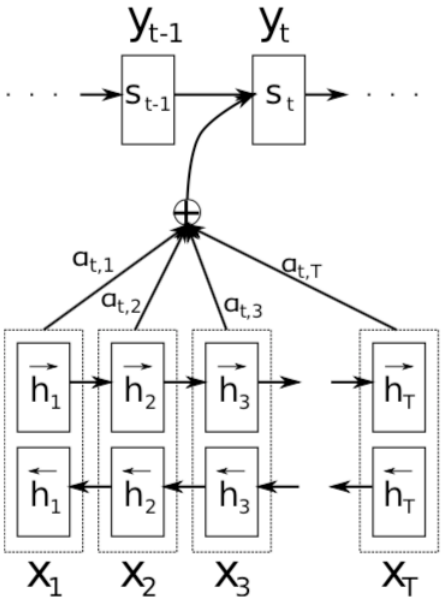
Neural Machine Translation by Jointly Learning to Align and Translate[2]

This paper tackled with one problem produced by the paper above. The basic encoder-decoder model requires the decoder to generate each hidden state with the conditioning of fixed length vector C (shown in Fig. 1). This process, however, is against intuition: certain part of the sequence is not necessarily related to all parts of the input sequence. As a result, attention algorithm was proposed in this paper.

Encoder part did not change much in this paper. The only modification is that the author utilized bidirectional RNN encoder here instead of unidirectional one. The main algorithm improvement was in the decoder side. Instead of computing each hidden state based on previous hidden state $h_{<t-1>}$, previous generated symbol y_{t-1}, and fixed length vector C, the attention mechanism computed hidden state based on specific discrete parts of C_i (together with $h_{<t-1>}$ and y_{t-1})[2].

The basic idea here is to find the context information relations between the input sequence and output symbols near targeted position. In other words, to find "how well the inputs around position j and output of position i match" [2].Since the output should be related to several input symbols instead of one, the function can be written as the weighted sum of the hidden state $h_{<t>}$:

$$C_i = \sum_{j=1}^{T_x} a_{ij} h_j$$



a_j here is like a softmax function for e_{ij}, which is defined by

$$e_{ij} = a(s_{i-1}, h_j),$$

an alignment model did exactly what mentioned before (how well the inputs around position j and output of position i match).

References

[1] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.