Syntax + NMT

Xing Wang

2017-02-27

http://blog.csdn.net/wangxinginnlp/article/details/56488921

- Syntax + NMT
  - ➢Integrate **syntax knowledge** into the encoder-decoder architecture

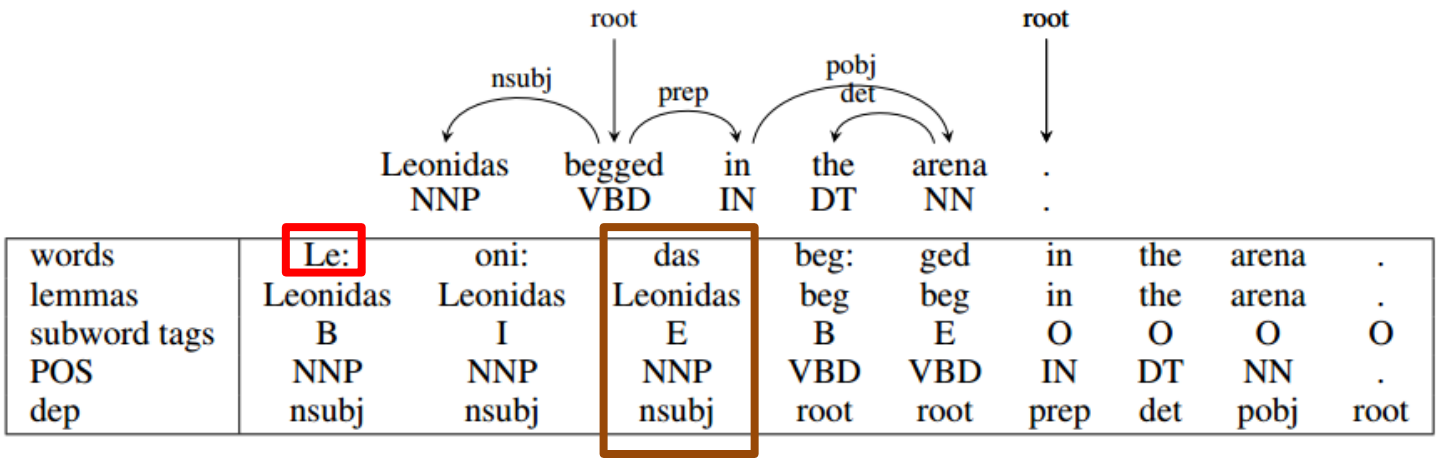
- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

Related Work

1. **Linguistic Input Features Improve Neural Machine Translation (WMT2016)**
2. **Tree-to-Sequence Attentional Neural Machine Translation (ACL2016)**
3. **Multi-task sequence to sequence learning (ICLR 2016)**
4. **Factored Neural Machine Translation**
5. **Learning to Parse and Translate Improves Neural Machine Translation**
6. **Syntax-aware Neural Machine Translation Using CCG**
7. **Neural Machine Translation with Source-Side Latent Graph Parsing**
8. **Language to Logical Form with Neural Attention (ACL 2016)**
9. **Tree-structured decoding with doubly-recurrent neural networks (ICLR 2017)**

Linguistic Input Features Improve Neural Machine Translation (WMT2016)

- Enrich the input unit



words	Le:	oni:	das	beg:	ged	in	the	arena	.
lemmas	Leonidas	Leonidas	Leonidas	beg	beg	in	the	arena	.
subword tags	B	I	E	B	E	O	O	O	O
POS	NNP	NNP	NNP	VBD	VBD	IN	DT	NN	.
dep	nsubj	nsubj	nsubj	root	root	prep	det	pobj	root

Figure 1: Original dependency tree for sentence *Leonidas begged in the arena .*, and our feature representation after BPE segmentation.

Linguistic Input Features Improve Neural Machine Translation (WMT2016)

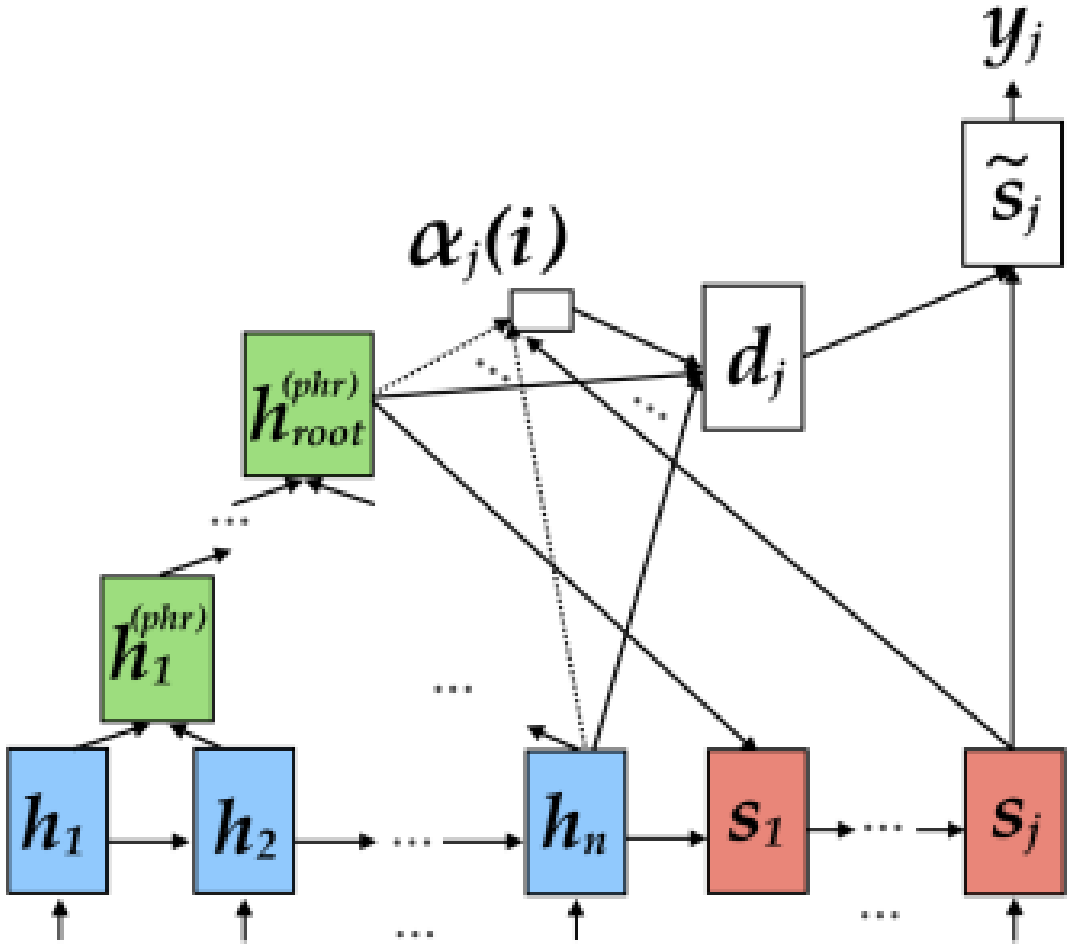
- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

Tree-to-Sequence Attentional Neural Machine Translation (ACL2016)

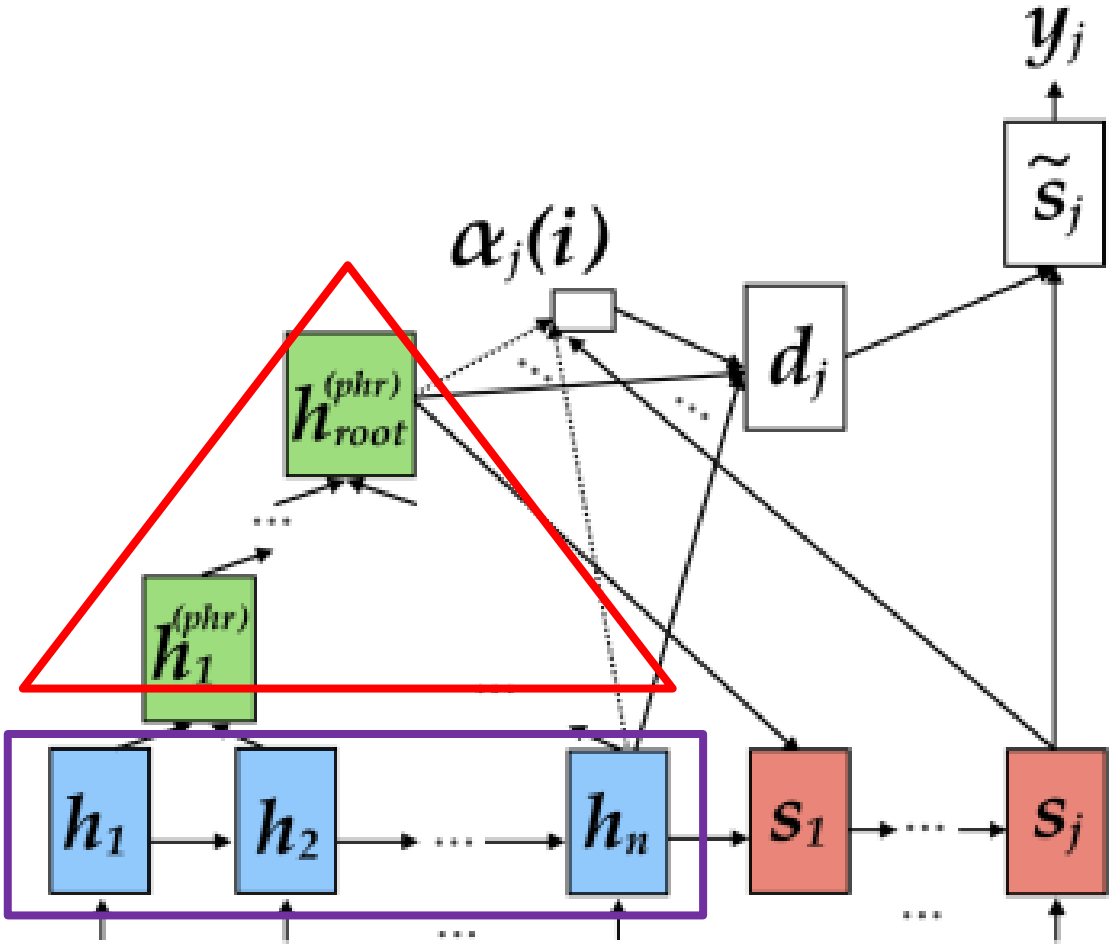
- Tree node representation:
  - The k-th parent hidden unit for the k-th phrase is calculated using the **left and right child hidden units** as follows: $\boldsymbol{h}_k^{(phr)} = f_{tree}(\boldsymbol{h}_k^l, \boldsymbol{h}_k^r)$
- Attention Mechanism:
  - The j-th context vector is composed of the **sequential and phrase vectors** weighted by the attention score:

$$\boldsymbol{d}_j = \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i + \sum_{i=n+1}^{2n-1} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i^{(phr)}$$

Tree-to-Sequence Attentional Neural Machine Translation (ACL2016)



Tree-to-Sequence Attentional Neural Machine Translation (ACL2016)

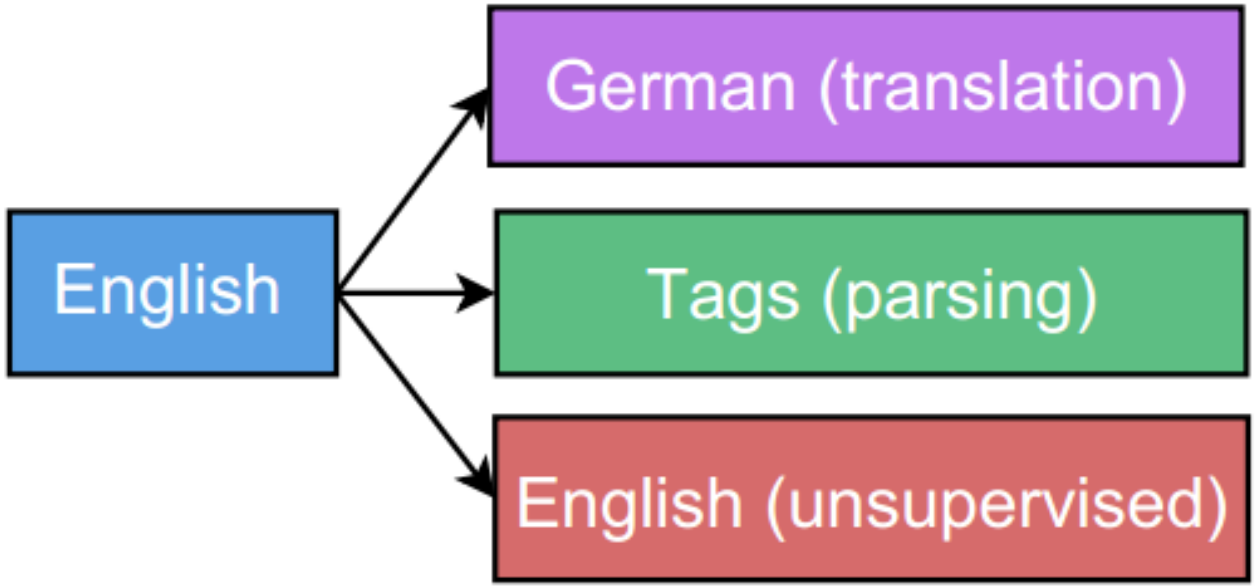


Tree-to-Sequence Attentional Neural Machine Translation (ACL2016)

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

Multi-task sequence to sequence learning (ICLR 2016)

- One-to-many Setting
  - ➢ one encoder, multiple decoders

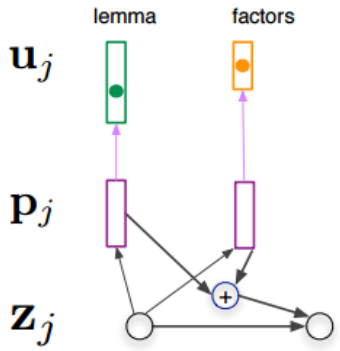


Multi-task sequence to sequence learning (ICLR 2016)

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

Factored Neural Machine Translation

- Generate only two symbols: the lemma and the concatenation of the different factors that we considered



- Lemmas and factors are generated separately, which in some cases, lead to sequences with different length.
  - To solve this problem, we give **priority** to the length of the lemmas. Consequently, we **constraint** the length of the factors sequence to be equal to the length of the lemma sequence

Factored Neural Machine Translation

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

Syntax-aware Neural Machine Translation Using CCG

- Both the source-side and target-side syntactic information are used

Source-side

BPE:	Obama	receives	Net+	an+	yahu	in	the	capital	of	USA
IOB:	O	O	B	I	E	O	O	O	O	O
CCG:	NP	S\NP/PP/NP	NP	NP	NP	PP/NP	NP/N	N	NP\NP/NP	NP

Target-side

NP Obama S\NP/PP/NP receives NP Net+ an+ yahu PP/NP in NP/N the N capital NP\NP/NP of NP USA

Syntax-aware Neural Machine Translation Using CCG

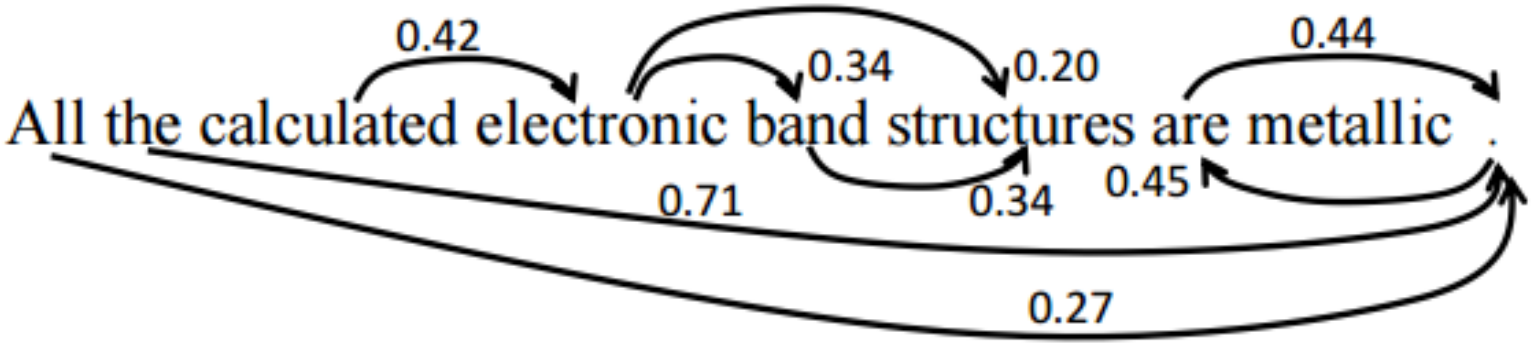
- Target-side syntax
  1. Serializing
     - **Increases the length** of the target sequence which might lead to loss of information learned at lexical level
  2. Multitasking (1) – shared encoder
     - There is **no explicit way to constrain the number** of predicted words and tags to match.
  3. Multitasking (2) – distinct softmax
     - **Does not condition the prediction** of target words on the syntactic context.

Syntax-aware Neural Machine Translation Using CCG

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

Neural Machine Translation with Source-Side Latent Graph Parsing

- Jointly learns translation and source-side latent graph representations of sentences



Neural Machine Translation with Source-Side Latent Graph Parsing

- The encoder of our model is a three-layer LSTM, where the first two layers are bi-directional
  - POS Tagging Layer
  - Dependency Parsing Layer

Neural Machine Translation with Source-Side Latent Graph Parsing

- The latent graph representation is obtained in dependency parsing layer

$$p(H_{w_i} = w_j | w_i) = \frac{\exp\left(m(i,j)\right)}{\sum_{k \neq i} \exp\left(m(i,k)\right)}$$

$$dep(w_i) = \tanh(W_{dep}[h_i^{enc}; \overline{h}(H_{w_i}); p(\ell_{w_i}|w_i)])$$

- Attention mechanism over constituency tree nodes:

$$a'_t = \sum_{i=1}^{N} s'(i,t) dep(w_i) \qquad \tilde{h}_t^{(dec)} = \tanh(\tilde{W}[h_t^{(dec)}; a_t; a'_t])$$

Neural Machine Translation with Source-Side Latent Graph Parsing

- Syntax + NMT
  1. *Enrich the input/output unit*
  2. Multi-task learning
  3. *Tree-to-Sequence* or Sequence-to-Tree

Learning to Parse and Translate Improves Neural Machine Translation

- Background: Recurrent Neural Network Grammars

Input: *The hungry cat meows .*

	Stack	Buffer	Action
0		*The* | *hungry* | *cat* | *meows* | .	NT(S)
1	(S	*The* | *hungry* | *cat* | *meows* | .	NT(NP)
2	(S | (NP	*The* | *hungry* | *cat* | *meows* | .	SHIFT
3	(S | (NP | *The*	*hungry* | *cat* | *meows* | .	SHIFT
4	(S | (NP | *The* | *hungry*	*cat* | *meows* | .	SHIFT
5	(S | (NP | *The* | *hungry* | *cat*	*meows* | .	REDUCE
6	(S | (NP *The hungry cat*)	*meows* | .	NT(VP)
7	(S | (NP *The hungry cat*) | (VP	*meows* | .	SHIFT
8	(S | (NP *The hungry cat*) | (VP *meows*	.	REDUCE
9	(S | (NP *The hungry cat*) | (VP *meows*)	.	SHIFT
10	(S | (NP *The hungry cat*) | (VP *meows*) | .		REDUCE
11	(S (NP *The hungry cat*) (VP *meows*) .)		

Learning to Parse and Translate Improves Neural Machine Translation

- At each time step, the action sLSTM **predicts the next action** based on the (current) hidden states of the **buffer**, **stack** and **action** sLSTM

Input: *The hungry cat meows .*

	Stack	Buffer	Action
0		*The* | *hungry* | *cat* | *meows* | .	NT(S)
1	(S	*The* | *hungry* | *cat* | *meows* | .	NT(NP)
2	(S | (NP	*The* | *hungry* | *cat* | *meows* | .	SHIFT
3	(S | (NP | *The*	*hungry* | *cat* | *meows* | .	SHIFT
4	(S | (NP | *The* | *hungry*	*cat* | *meows* | .	SHIFT
5	(S | (NP | *The* | *hungry* | *cat*	*meows* | .	REDUCE
6	(S | (NP *The hungry cat*)	*meows* | .	NT(VP)
7	(S | (NP *The hungry cat*) | (VP	*meows* | .	SHIFT
8	(S | (NP *The hungry cat*) | (VP *meows*	.	REDUCE
9	(S | (NP *The hungry cat*) | (VP *meows*)	.	SHIFT
10	(S | (NP *The hungry cat*) | (VP *meows*) | .		REDUCE
11	(S (NP *The hungry cat*) (VP *meows*) .)		

Learning to Parse and Translate Improves Neural Machine Translation

- NMT+RG
  - Replace the hidden state of the buffer h buffer with the hidden state of the decoder of the attention-based neural machine translation

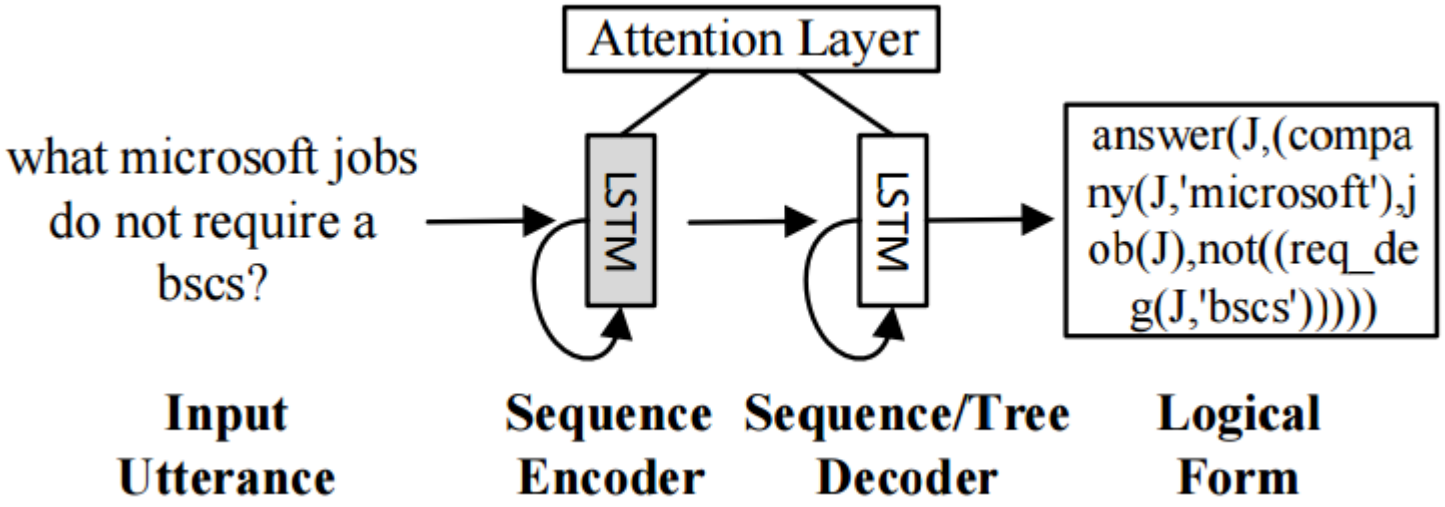
- The generator of RNNG will output a word, when asked by the shift action, according to the conditional distribution defined by the translation decoder

Learning to Parse and Translate Improves Neural Machine Translation

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

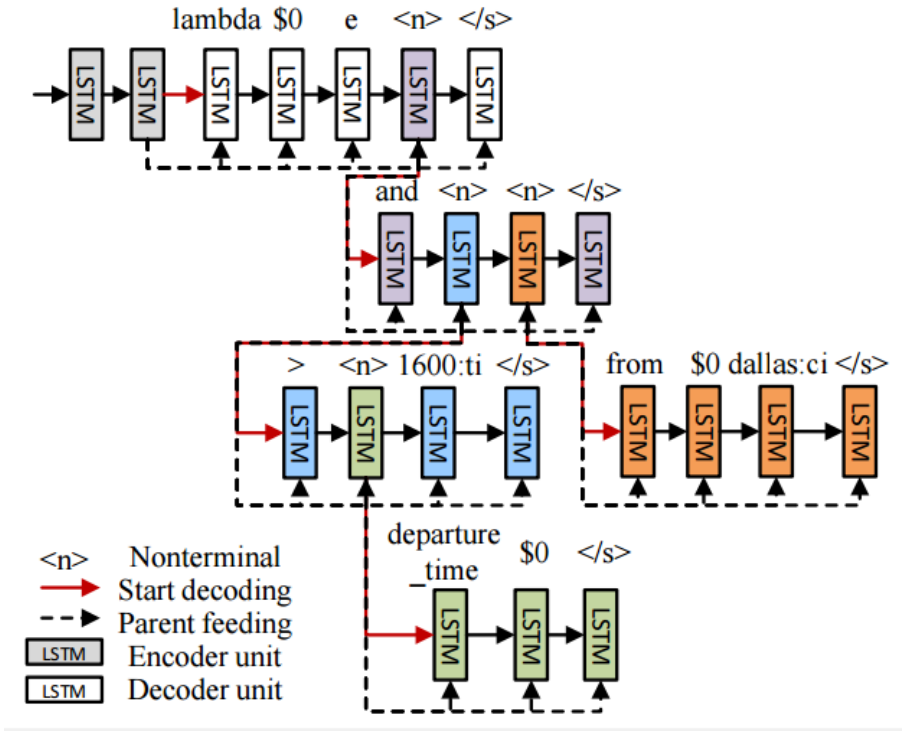
Language to Logical Form with Neural Attention (ACL 2016)

- Use attention-enhanced encoder-decoder model for semantic parsing



Language to Logical Form with Neural Attention (ACL 2016)

- Sequence-to-tree (SEQ2TREE) model with a hierarchical tree decoder (**Breadth First Generation**)



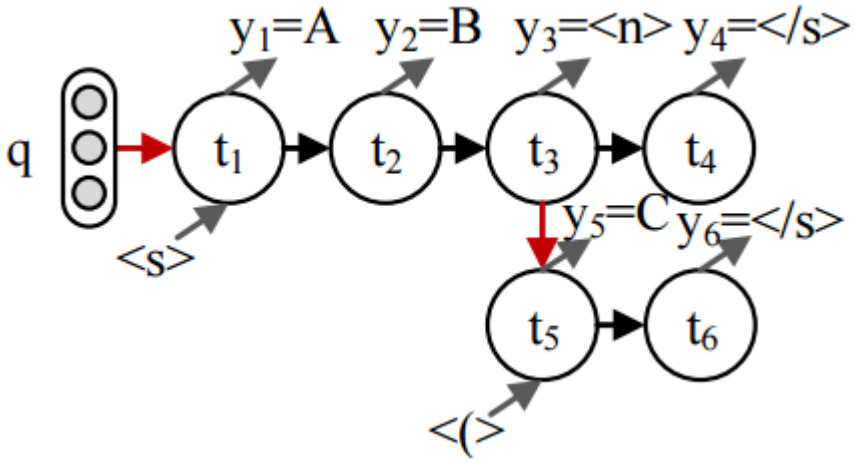
Language to Logical Form with Neural Attention (ACL 2016)

- Tree structure generation
  - Attach an EOS symbol to each subtree. (padding methods)

- Subtree decoding by conditioning on the nonterminal's hidden vector
  - In order to better utilize the parent nonterminal's information, we introduce a **parent-feeding connection**

Language to Logical Form with Neural Attention (ACL 2016)

- Parent-feeding connection



- Previous state of t5
  - Seq-to-Seq/Time view: t4
  - Seq-to-Tree/Structure view: t3

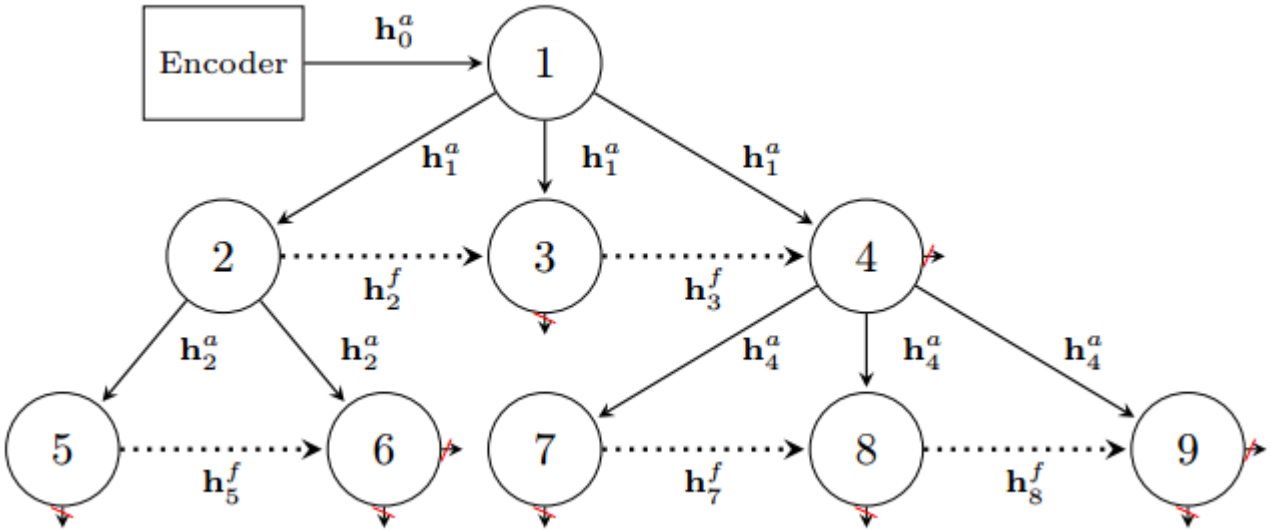
$$p(a|q) = p(y_1 y_2 y_3 y_4|q)\, p(y_5 y_6|y_{\leq 3}, q)$$

Language to Logical Form with Neural Attention (ACL 2016)

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

Tree-structured decoding with doubly-recurrent neural networks (ICLR 2017)

- Both **ancestral** (parent-to-children) and **fraternal** (sibling-to-sibling) flow of information is modeled with recurrent modules.



Tree-structured decoding with doubly-recurrent neural networks (ICLR 2017)

- For a node i $\in$ V with parent p(i) and previous sibling s(i), the **ancestral** and **fraternal** hidden states are updated via :

$$\mathbf{h}_i^a = g^a(\mathbf{h}_{p(i)}^a, \mathbf{x}_{p(i)})$$
$$\mathbf{h}_i^f = g^f(\mathbf{h}_{s(i)}^f, \mathbf{x}_{s(i)})$$

- Once the hidden depth and width states have been updated with these observed labels, they are combined to obtain a predictive hidden state:

$$\mathbf{h}_i^{(pred)} = \tanh\left(\mathbf{U}^f \mathbf{h}_i^f + \mathbf{U}^a \mathbf{h}_i^a\right)$$

Tree-structured decoding with doubly-recurrent neural networks (ICLR 2017)

- Topology prediction
  - The probability that node i has children:

$$p_i^a = \sigma(\mathbf{u}^a \cdot \mathbf{h}_i^{(pred)})$$

  - The probability of stopping fraternal branch growth after the current node i:

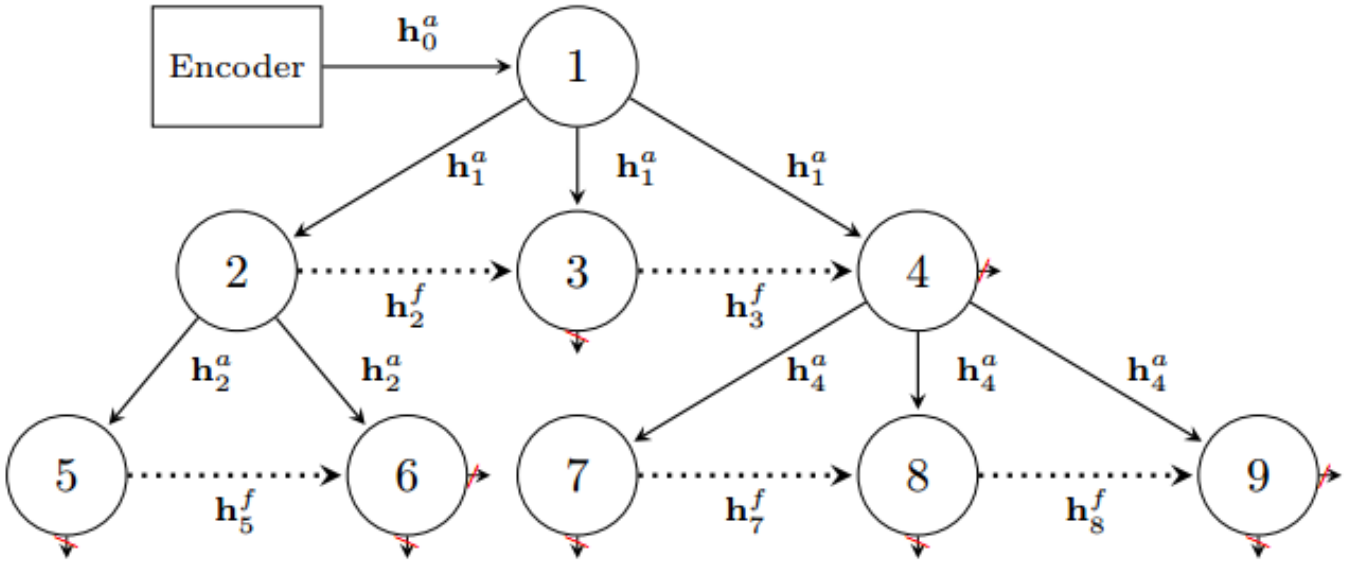
$$p_i^f = \sigma(\mathbf{u}^f \cdot \mathbf{h}_i^{(pred)})$$

  - Feed the topological information to the label prediction layer:

$$\mathbf{o}_i = \text{softmax}(\mathbf{W}\mathbf{h}_i^{(pred)} + \alpha_i \mathbf{v}^a + \varphi_i \mathbf{v}^f)$$

Tree-structured decoding with doubly-recurrent neural networks (ICLR 2017)

- During testing, these values are obtained from p_a , p_f by sampling or beam-search???



Tree-structured decoding with doubly-recurrent neural networks (ICLR 2017)

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

- Syntax + NMT
  1. Enrich the input/output unit
  2. Multi-task learning
  3. Tree-to-Sequence or Sequence-to-Tree

- http://blog.csdn.net/wangxinginnlp/article/details/56488921

Thank you!!!