

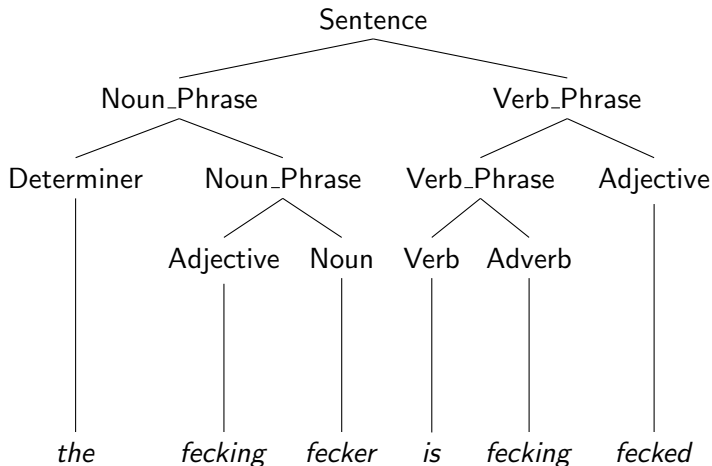
Ordered Neurons: integrating tree structures into recurrent neural networks; Shen, Tan, Sordoni, Courville (1810.09536)

Conor Houghton

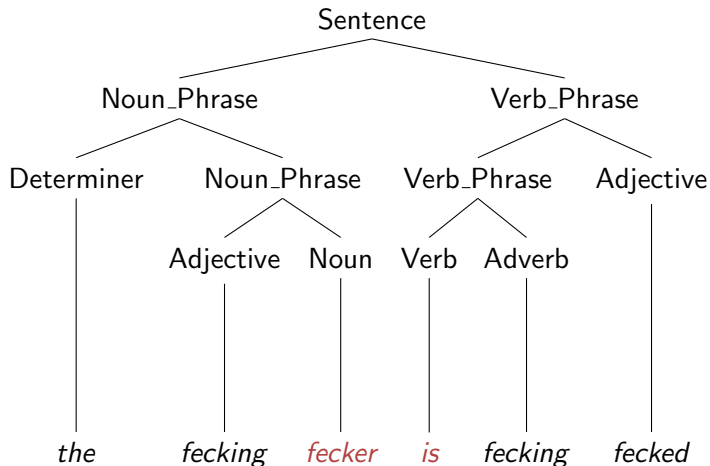
journal club: 1818.09536

2019-06-27

Grammar is a tree



Trees support grammatical rules



Grammar is a tree

S					
NP			VP		
D	NP		VP		Adj
	Adj	N	V	Adv	
the	fecking	fecker	is	fecking	fecked

S					
NP			VP		
D	NP		VP		Adj
	Adj	N	V	Adv	
the	fecking	fecker	is	fecking	fecked

You would guess that there is activity, maybe even an individual neuron, for tracking the opening and closing of this noun phrase.

RNN

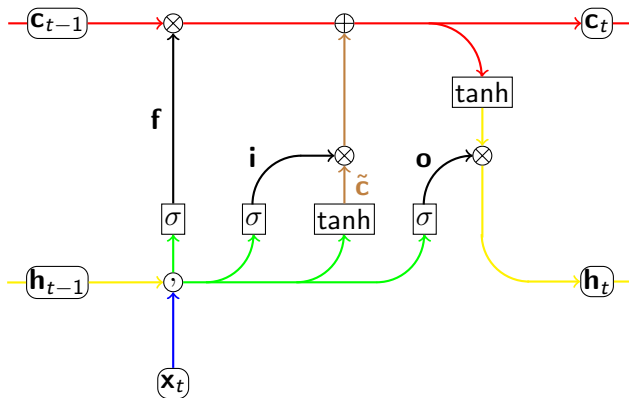
S					
NP			VP		
D	NP		VP		Adj
	Adj	N	V	Adv	
the	fecking	fecker	is	fecking	fecked

You would guess that there is activity, maybe even an individual neuron, for tracking the opening and closing of this noun phrase.

S					
NP			VP		
D	NP		VP		Adj
	Adj	N	V	Adv	
the	fecking	fecker	is	fecking	fecked

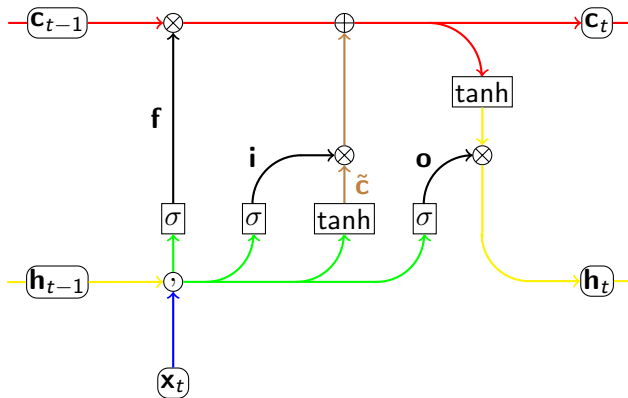
However, the tree structure means that when a higher cell closes, the cells beneath it must close as well. This is what this paper seeks to implement.

LSTM



LSTM

$$\begin{aligned}\mathbf{f} &= \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\ \mathbf{c}_t &= \mathbf{f} \circ \mathbf{c}_{t-1} + \mathbf{i} \circ \tilde{\mathbf{c}}\end{aligned}$$



The forget gate

Morally:

$$\mathbf{f} = (0, 0, 1, 1, 0, 1, 0, 0, \dots, 0, 1, 1, 0)$$

but the sigma-function softens it. The idea in this paper is to introduce a ‘master’ forget gate that does this:

$$\mathbf{f} = (0, 0, 0, 0, \dots, 0, 0, 1, 1, \dots, 1, 1)$$

so there is a change point, with all the neurons on one side zero and the other side one.

The master forget gate

In the usual forget gate the activation $W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f$ works on a neuron-by-neuron basis.

Morally in the master forget gate the maximum component of

$$W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f$$

selects the change point, the point where

$$\mathbf{f} = (0, 0, 0, 0, \dots, 0, 0, 1, 1, \dots, 1, 1)$$

goes from zeros to ones.

Revising soft-max

Recall how soft-max works:

$$\mu_{\beta}(x_i) = \frac{e^{\beta x_i}}{\sum_j e^{\beta x_j}}$$

so for $\beta \rightarrow \infty$

$$\mu_{\beta}(\circ \mathbf{x}) = (0, 0, 0, 0, \dots, 0, 0, 1, 0, \dots, 0, 0)$$

where the one indicates the index of the largest x_i . Thus 'soft-max' is a misnomer, it should be 'soft-max-index'. Finite β softens it, we think of the x_i as having some noise and $\mu_{\beta}(x_i)$ giving the probability that x_i is the largest value in \mathbf{x} .

The master forget gate

So let

$$\mathbf{u} = W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f$$

so that

$$\mu_\beta(\circ \mathbf{u}) = (p_1, p_2, \dots, p_n)$$

is the vector of probabilities with p_i is the probability the index i is the change point. Then the probability that f_j should be one is $\sum_{i=1}^j p_i$. Hence, if S is the cumulative function

$$S(\circ \mathbf{p}) = \left(p_1, \sum_{i=1}^2 p_i, \dots, \sum_{i=1}^j p_i, \dots, \sum_{i=1}^n p_i \right)$$

we define the master forget gate

$$f = S(\circ \mu_\beta(\circ \mathbf{u}))$$

The master forget gate

Note that this is a very different use of $W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f$, before it was used to calculate the individual gating values, now it is being used to find the change point.

Some details

Morally the master forget gate

$$\mathbf{f} = (0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1)$$

and a master include gate

$$\mathbf{i} = (1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)$$

Here there is an overlap region

$$\mathbf{w} = (0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0)$$

In the overlap region they also apply local, old-style include and forget gates. They also coarse-grain the master forget and include gates with a factor of ten to reduce parameters. All this is softened by β .

Results - language modelling

$$\text{perplexity} = \left(\prod_{i=1}^K \frac{1}{\text{probability of } i\text{th word}} \right)^{1/K}$$

Results - language modelling

	parameters	perplexity
CharCNN (2016)	19M	78.9
4-layer LSTM with skip connections (2017)	24M	58.3
3-layer AWD-LSTM (2017)	24M	57.3
Ordered LSTM	25M	56.2
AWD-LSTM-MoS (2017)	22M	54.4

Results - parsing

Look at the forget gate at each time step¹ and infer from that the expected value of the change point, roughly f_i is the cumulative over neurons so $\delta f_i = f_i - f_{i-1}$ is the probability i is the change point and so

$$d = \sum_i i \delta f_i$$

is the expected change point. This measure how much we forget at each time step.

The time step with the largest d , the one that forgets the most, is the first split in the sentence, then iterate from there.

¹a time step is the addition of a word

Results - parsing

The second layer of ON-LSTM achieves state-of-the-art unsupervised constituency parsing results on the WSJ test set, while the first and third layers do not perform as well. One possible interpretation is that the first and last layers may be too focused on capturing local information useful for the language modeling task as they are directly exposed to input tokens and output predictions respectively, thus may not be encouraged to learn the more abstract tree structure. Since the WSJ test set contains sentences of various lengths which are unserved during training, we find that ON-LSTM provides better generalization and robustness toward longer sentences than previous models.

Results - dependency

	LSTM	ON-LSTM
Simple agreement		
The author laugh(s)	1.0	0.99
In an object relative clause		
The farmer that the parents love(*s) swims	0.88	0.84
In an object relative clause		
The farmer the parents love(*s) swims	0.81	0.78
Negative polarity		
(No/*most) authors have ever been famous	0.67	0.07
Across a subject relative clause		
The officers that love the skater smile(*s)	0.6	0.66

Results - logic

Six proposals p_1 to p_6 and three logical operators, **not**, **or** and **and** are used to make random statements:

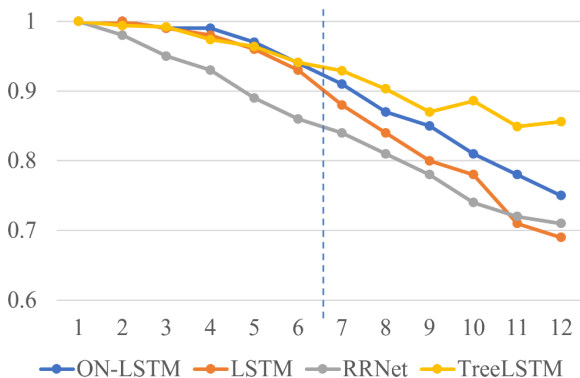
$$(p_1 \text{ or } (p_2 \text{ and } p_3))$$

The relationship between pairs of statements are checked by going through the truth table; these are classified into one of seven relationships such as *implies*, or *is equivalent*, for example

$$p_2 \implies (p_2 \text{ or } p_3)$$

Results - logic

The goal is to find the relationship given the two statements. This is performed by feeding the hidden layers for the two statements to a classifier.



Conclusions

- Cool idea!
- Pretty good at finding tree structures
- Not as good at grammar as you'd expect
 - ▶ Maybe trees don't matter so much to grammar
 - ▶ Should the coarse-graining of the master forget gate have been much coarser?