Exploring Composition Trees in Sequence Classification

A. Drozdov S. Bowman

March 22, 2017

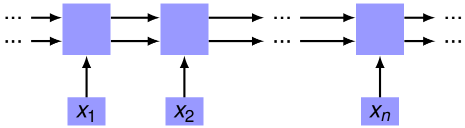
Pretend you are a robot . . .

- Classifying sequences is your purpose.
- Composition functions are your hammer.
- Some sequences have interesting substructure.

Some Definitions

Two structures and composition functions ...

1. Linear Chain and LSTM

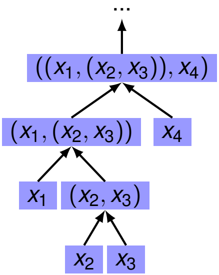


$$
\begin{aligned}
\mathbf{i} &= \sigma(\mathbf{W}^i * [\mathbf{h}_{t-1}, \mathbf{x}_t]) \\
\mathbf{f} &= \sigma(\mathbf{W}^f * [\mathbf{h}_{t-1}, \mathbf{x}_t]) \\
\mathbf{o} &= \sigma(\mathbf{W}^o * [\mathbf{h}_{t-1}, \mathbf{x}_t]) \\
\mathbf{g} &= \tanh(\mathbf{W}^g * [\mathbf{h}_{t-1}, \mathbf{x}_t]) \\
\mathbf{c}_t &= \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g} \\
\mathbf{h}_t &= \tanh(\mathbf{o} \odot \mathbf{c}_t)
\end{aligned}
$$

Some Definitions

Two structures and composition functions . . .

2. Binary Tree and TreeLSTM



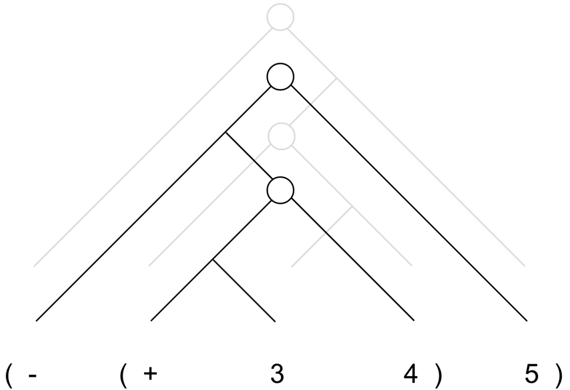
$$\begin{bmatrix} \vec{i} \\ \vec{f}_l \\ \vec{f}_r \\ \vec{o} \\ \vec{g} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W_{\text{comp}} \begin{bmatrix} \vec{h}_s^1 \\ \vec{h}_s^2 \\ \vec{e} \end{bmatrix} + \vec{b}_{\text{comp}} \right)$$

$$\vec{c} = \vec{f}_l \odot \vec{c}_s^{\,2} + \vec{f}_r \odot \vec{c}_s^{\,1} + \vec{i} \odot \vec{g}$$

$$\vec{h} = \vec{o} \odot \tanh(\vec{c})$$

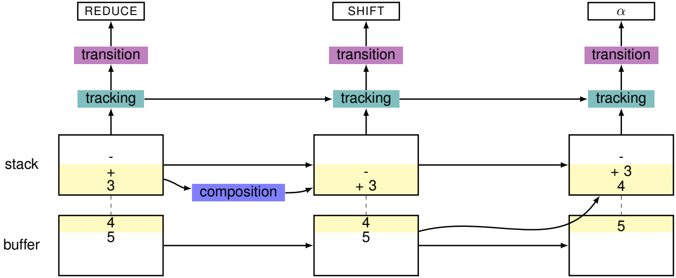
Concrete Task

Equation in prefix notation.



Supervised SPINN

Stack-augmented Parser-Interpreter Neural Network



$$J_n = -\Sigma_i^B \log p(y_i) - \tau \frac{1}{\Sigma_i^B |T_i|} \Sigma_i^B \Sigma_t^{|T_i|} \log p(\alpha_t)$$

REINFORCE

1. Sample the transition during train time (Greedy at test time).
2. Modify the loss function:

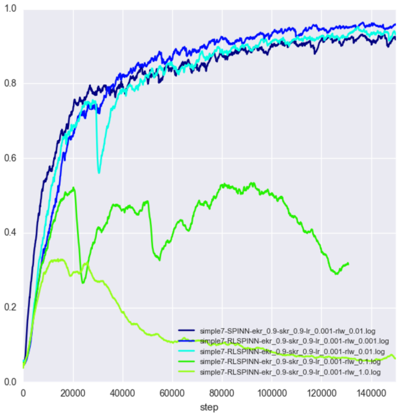
$$J_n = -\Sigma_i^B \log p(y_i) - \tau \frac{1}{\Sigma_i^B |T_i|} \Sigma_i^B \Sigma_t^{|T_i|} \log p(\hat{\alpha}_t)(R_i - b_n)$$

$R_i :=$ Classification Accuracy (0 or 1)
$b :=$ Estimate of the reward using exponential moving average:

$$b_n = (1 - \beta) \times b_{n-1} + \beta \times \mu(R)$$

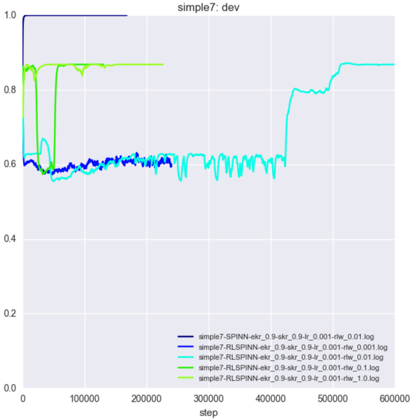
Experiments

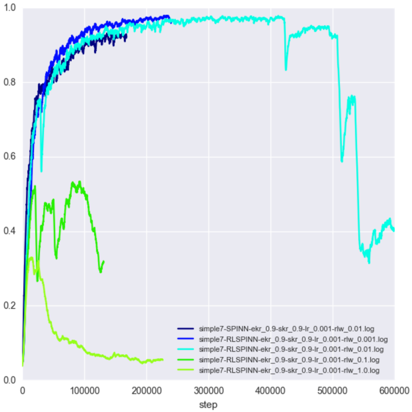
- RMSprop, $\eta = 0.001$, dropout (keep)=0.9, $L_2 = 2.75^{-5}$
- $\tau \in [1.0, 0.1, 0.01, 0.001]$



Experiments

- RMSprop, $\eta = 0.001$, dropout (keep)=0.9, $L_2 = 2.75^{-5}$
- $\tau \in [1.0, 0.1, 0.01, 0.001]$





Why? What?

Why?

1. There is a lot of variance in Reward.
2. It is easy to get stuck in a trivial parsing strategy.

What?

1. Reduce variance using a different baseline and variance normalization.

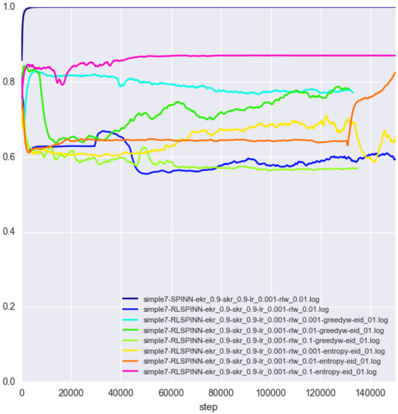
$$\hat{A}_i = \frac{A_i - \mu(A)}{\sigma(A)}, A_i = R_i - b_n$$

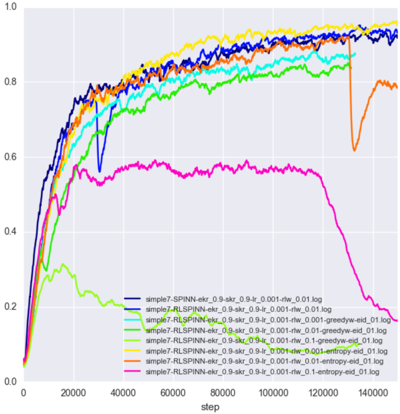
2. Encourage exploration through entropy regularization (subtracting entropy).

$$H = -p(\hat{\alpha}_t) \log(p(\hat{\alpha}_t))$$

Followup

- Greedy Max with Variance Normalization
- Entropy Regularization





Future Work

Natural Language

- Multi-Genre NLI
- Neural Machine Translation

Pre-trained Parser

- Recursive Autoencoder with Part of Speech Tags
- Information Maximization Objective

Thanks

Acknowledgements:
James, John, Alex, Shivam, Theo, CS/CDS, Nvidia.