Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks

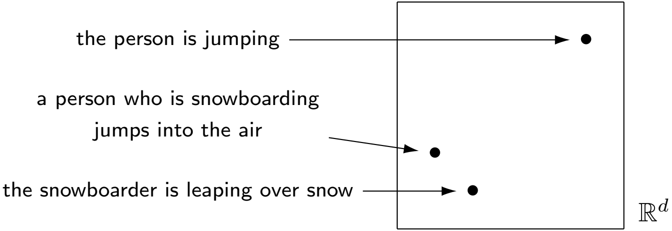
Kai Sheng Tai[†‡], Richard Socher[‡], and Christopher D. Manning[†]

[†]Stanford University, [‡]MetaMind

July 29, 2015

読み手: 岡崎直観（東北大学）

Distributed Sentence Representations

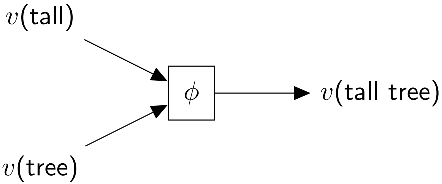


- Like word vectors, represent sentences as real-valued vectors
- What for?
  - Sentence classification
  - Semantic relatedness / paraphrase
  - Machine translation
  - Information retrieval

Our Work

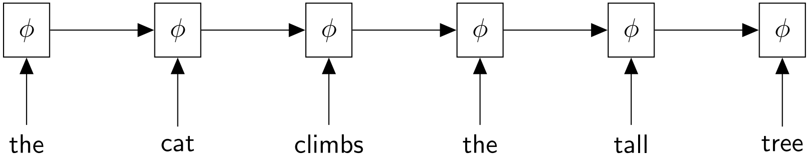
- A new model for sentence representations: Tree-LSTMs
- Generalizes the widely-used chain-structured LSTM
- New state-of-the-art empirical results:
  - Sentiment classification (Stanford Sentiment Treebank)
  - Semantic relatedness (SICK dataset)

Compositional Representations



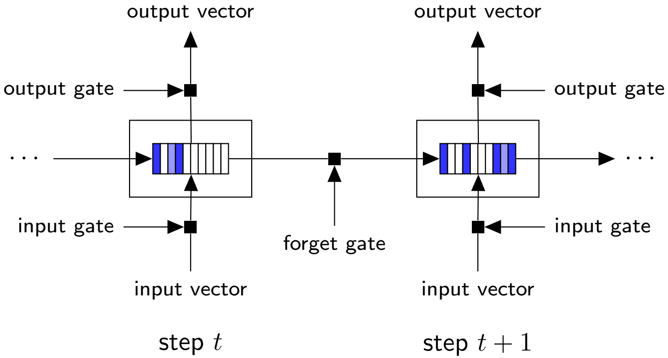
- Idea: Compose phrase and sentence reps from their constituents
- Use a composition function ϕ
- Steps:
  1. Choose some compositional order for a sentence
     - e.g. sequentially left-to-right
  2. Recursively apply ϕ until representation for entire sentence is obtained
- We want to learn ϕ from data

Sequential Composition



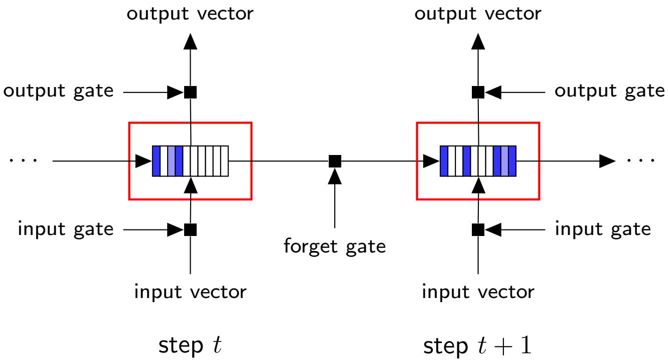
- State is composed left-to-right
- Input at each time step is a word vector
- Rightmost output is the representation of the entire sentence
- Common parameterization: recurrent neural network (RNN)

Sequential Composition: Long Short-Term Memory (LSTM) Networks



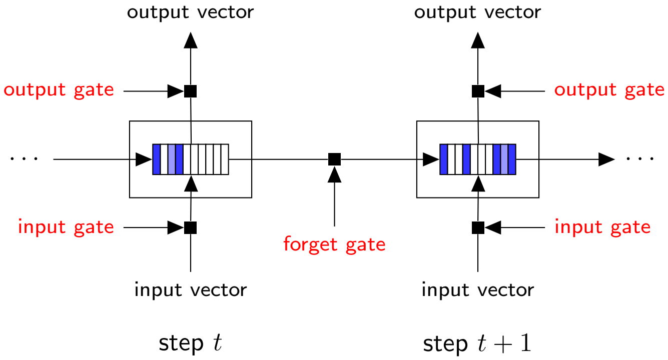
- A particular parameterization of the composition function ϕ
- Recent popularity: strong empirical results on sequence-based tasks
  - e.g. language modeling, neural machine translation

Sequential Composition: Long Short-Term Memory (LSTM) Networks



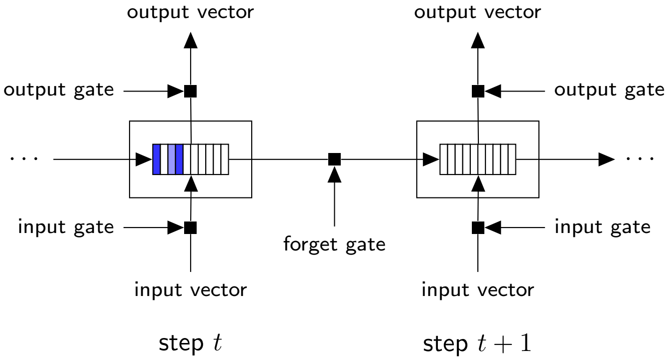
- Memory cell: a vector representing the inputs seen so far
- **Intuition:** state can be preserved over many time steps

Sequential Composition: Long Short-Term Memory (LSTM) Networks

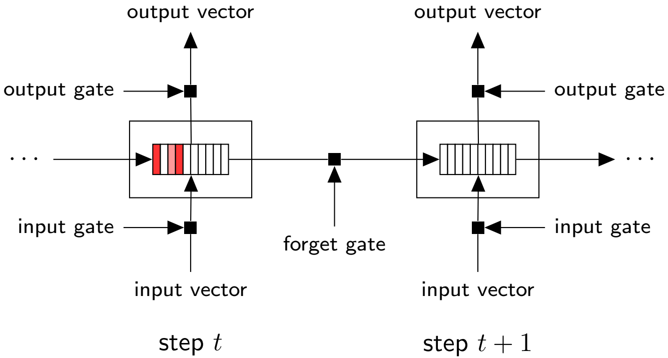


- Input/output/forget gates: vectors in $[0,1]^d$
- Multiplied elementwise ("soft masking")
- **Intuition:** Selective memory read/write, selective information propagation

Sequential Composition:
(Simplified) step-by-step LSTM composition

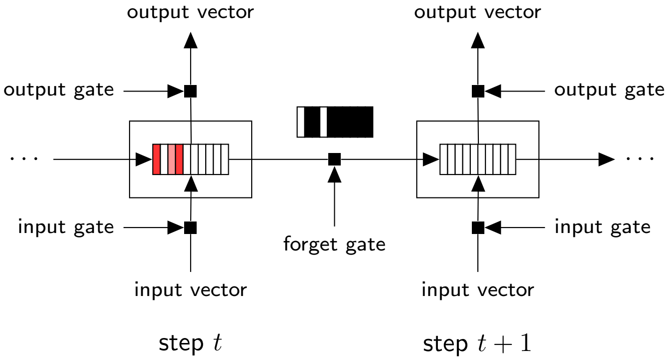


Sequential Composition:
(Simplified) step-by-step LSTM composition



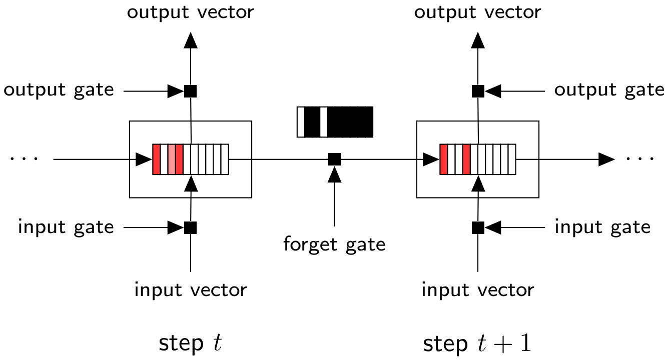
1. **Starting with state at t**

Sequential Composition:
(Simplified) step-by-step LSTM composition



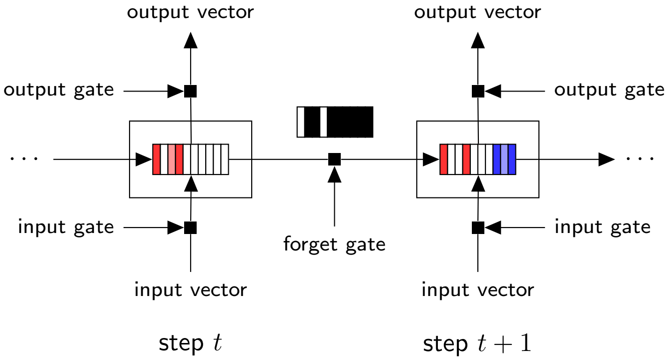
1. Starting with state at t
2. **Predict gates from input and state at t**

Sequential Composition:
(Simplified) step-by-step LSTM composition



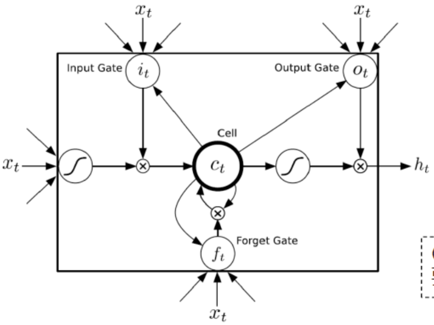
1. Starting with state at t
2. Predict gates from input and state at t
3. **Mask memory cell with forget gate**

Sequential Composition:
(Simplified) step-by-step LSTM composition



1. Starting with state at t
2. Predict gates from input and state at t
3. Mask memory cell with forget gate
4. **Add update computed from input and state at** t

Long Short-Term Memory (LSTM)

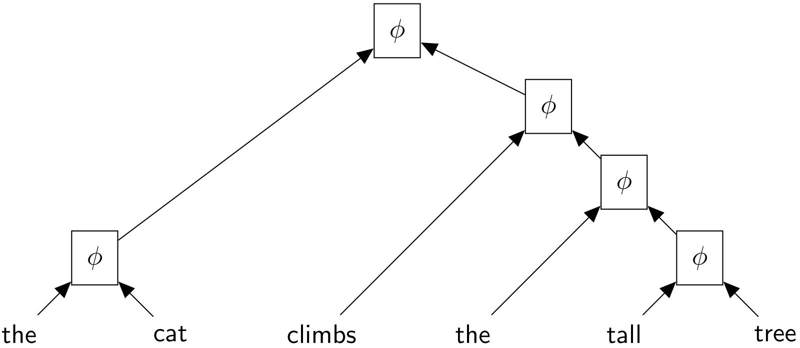


⊗（数式中は⊙）は
要素ごとの積

Alex Graves. (2013) Generating Sequences with Recurrent Neural Networks. arXiv.org

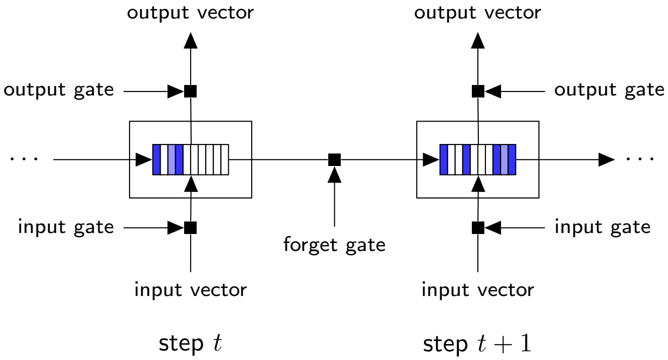
Input gate: $i_t = \sigma\left(W^{(xi)}x_t + W^{(hi)}h_{t-1} + W^{(ci)}c_{t-1} + b_i\right)$
Forget gate: $f_t = \sigma\left(W^{(xf)}x_t + W^{(hf)}h_{t-1} + W^{(cf)}c_{t-1} + b_f\right)$
Cell: $c_t = f_t \odot c_{t-1} + i_t \odot \tanh\left(W^{(xc)}x_t + W^{(hc)}h_{t-1} + b_c\right)$
Output gate: $o_t = \sigma\left(W^{(xo)}x_t + W^{(ho)}h_{t-1} + W^{(co)}c_t + b_o\right)$
Hidden variable: $h_t = o_t \odot \tanh(c_t)$

Tree-Structured Composition



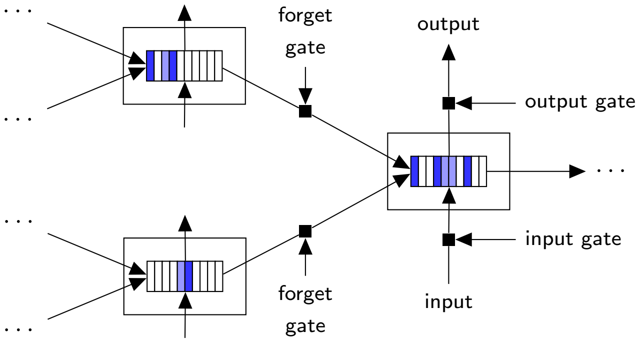
- In this work: compose following the syntactic structure of sentences
  - Dependency parse
  - Constituency parse
- Previous work: recursive neural networks
  (Goller and Kuchler, 1996; Socher et al., 2011)

Generalizing the LSTM



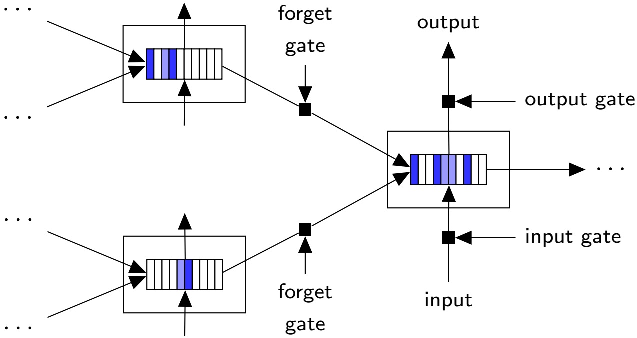
- Standard LSTM: each node has one child
- We want to generalize this to accept multiple children

Tree-Structured LSTMs



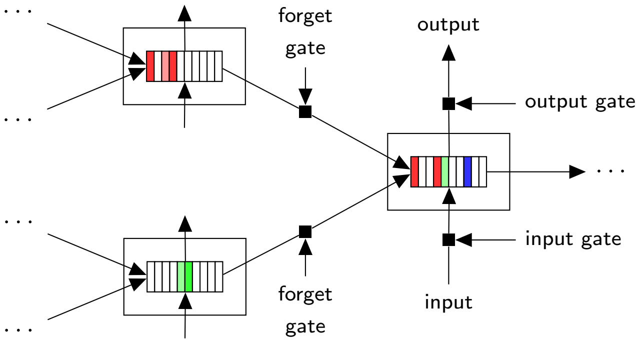
- Natural generalization of the sequential LSTM composition function
- Allows for trees with arbitrary branching factor
- Standard chain-structured LSTM is a special case

Tree-Structured LSTMs



- Key feature: A separate forget gate for each child
- Selectively preserve information from each child

Tree-Structured LSTMs



- Selectively preserve information from each child
- How can this be useful?
  - Ignoring unimportant clauses in sentence
  - Emphasizing sentiment-rich children for sentiment classification

Empirical Evaluation

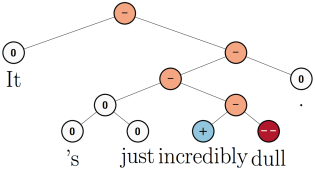
- Sentiment classification
  - Stanford Sentiment Treebank
- Semantic relatedness
  - SICK dataset, SemEval 2014 Task 1

	Relatedness		Sentiment	
LSTM Variant	d	$\|\theta\|$	d	$\|\theta\|$
Standard	150	203,400	168	315,840
Bidirectional	150	203,400	168	315,840
2-layer	108	203,472	120	318,720
Bidirectional 2-layer	108	203,472	120	318,720
Constituency Tree	142	205,190	150	316,800
Dependency Tree	150	203,400	168	315,840

d: 隠れ層の次元

Common Crawlコーパスにおいて Gloveで学習した300次元の単語ベクトル

Evaluation 1: Sentiment Classification



- **Task:** Predict the sentiment of movie review sentences
  - Binary subtask: positive / negative
  - 5-class subtask: strongly positive / positive / neutral / negative / strongly negative
- **Dataset:** Stanford Sentiment Treebank (Socher et al., 2013)
- **Supervision:** head-binarized constituency parse trees with sentiment labels at each node
- **Model:** Tree-LSTM on given parse trees, softmax classifier at each node

Evaluation 2: Semantic Relatedness

"the snowboarder is leaping over white snow" $\overset{?}{\sim}$ "a person who is practicing snowboarding jumps into the air"

- **Task:** Predict the semantic relatedness of sentence pairs
- **Dataset:** SICK from SemEval 2014, Task 1 (Marelli et al., 2014)
- **Supervision:** human-annotated relatedness scores $y \in [1, 5]$
- **Model:**
  - Sentence representation with Tree-LSTM on dependency parses
  - Similarity predicted by NN regressor given representations at root nodes

We first produce sentence representations h_L and h_R for each sentence in the pair using a Tree-LSTM model over each sentence's parse tree. Given these sentence representations, we predict the similarity score $\hat{y}$ using a neural network that considers both the distance and angle between the pair (h_L, h_R):

$$
\begin{aligned}
h_\times &= h_L \odot h_R, \\
h_+ &= |h_L - h_R|, \\
h_s &= \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right), \\
\hat{p}_\theta &= \mathrm{softmax}\left(W^{(p)} h_s + b^{(p)}\right), \\
\hat{y} &= r^T \hat{p}_\theta,
\end{aligned}
\tag{15}
$$

where $r^T = [1\ 2\ \ldots\ K]$ and the absolute value function is applied elementwise. The use of both distance measures $h_\times$ and h_+ is empirically motivated: we find that the combination outperforms the use of either measure alone. The multiplicative measure $h_\times$ can be interpreted as an elementwise comparison of the signs of the input representations.

We want the expected rating under the predicted distribution $\hat{p}_\theta$ given model parameters θ to be close to the gold rating $y \in [1, K]$: $\hat{y} = r^T \hat{p}_\theta \approx y$. We therefore define a sparse target distribution[1] p that satisfies $y = r^T p$:

$$
p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}
$$

for $1 \leq i \leq K$. The cost function is the regularized KL-divergence between p and $\hat{p}_\theta$:

$$
J(\theta) = \frac{1}{m} \sum_{k=1}^{m} \mathrm{KL}\left(p^{(k)} \,\middle\|\, \hat{p}_\theta^{(k)}\right) + \frac{\lambda}{2} \|\theta\|_2^2,
$$

where m is the number of training pairs and the superscript k indicates the kth sentence pair.

Sentiment Classification Results

Method	Fine-grained	Binary
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	46.4 (1.1)	84.9 (0.6)
Bidirectional LSTM	49.1 (1.0)	87.5 (0.5)
2-layer LSTM	46.0 (1.3)	86.3 (0.6)
2-layer Bidirectional LSTM	48.5 (1.0)	87.2 (1.0)
Dependency Tree-LSTM	48.4 (0.4)	85.7 (0.4)
Constituency Tree-LSTM		
– randomly initialized vectors	43.9 (0.6)	82.0 (0.5)
– Glove vectors, fixed	49.7 (0.4)	87.5 (0.8)
– Glove vectors, tuned	**51.0** (0.5)	88.0 (0.3)

←これで十分という説

ノードの数が少なくなるため、学習しづらい

学習時に単語ベクトルも更新すると良い

（著者スライドを大幅に改変）

Bidirectional LSTM

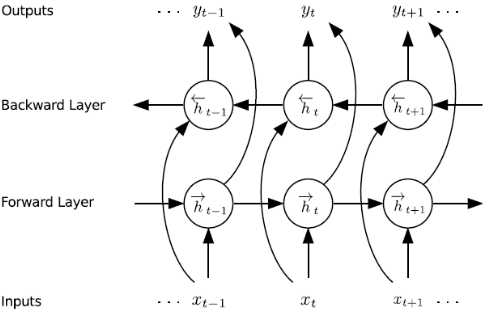


Fig. 2. Bidirectional Recurrent Neural Network

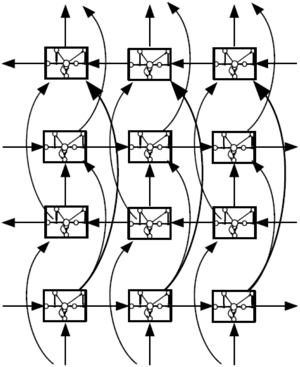


Fig. 4. Deep Bidirectional Long Short-Term Memory Network (DBLSTM)

A. Graves, N. Jaitly, A. Mohamed. Hybrid Speech Recognition with Deep Bidirectional LSTM. ASRU 2013.

Semantic Relatedness Results

Method	Pearson's r		Spearman's ρ		MSE	
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993		0.7538		0.3692	
UNAL-NLP (Jimenez et al., 2014)	0.8070		0.7489		0.3550	
Meaning Factory (Bjerva et al., 2014)	0.8268		0.7721		0.3224	
ECNU (Zhao et al., 2014)	0.8414		–		–	
Mean vectors	0.7577	(0.0013)	0.6738	(0.0027)	0.4557	(0.0090)
DT-RNN (Socher et al., 2014)	0.7923	(0.0070)	0.7319	(0.0071)	0.3822	(0.0137)
SDT-RNN (Socher et al., 2014)	0.7900	(0.0042)	0.7304	(0.0076)	0.3848	(0.0074)
LSTM	0.8528	(0.0031)	0.7911	(0.0059)	0.2831	(0.0092)
Bidirectional LSTM	0.8567	(0.0028)	0.7966	(0.0053)	0.2736	(0.0063)
2-layer LSTM	0.8515	(0.0066)	0.7896	(0.0088)	0.2838	(0.0150)
2-layer Bidirectional LSTM	0.8558	(0.0014)	0.7965	(0.0018)	0.2762	(0.0020)
Constituency Tree-LSTM	0.8582	(0.0038)	0.7966	(0.0053)	0.2734	(0.0108)
Dependency Tree-LSTM	**0.8676**	(0.0030)	**0.8083**	(0.0042)	**0.2532**	(0.0052)

括弧内は標準偏差。dependencyの方が性能が良いのは、おそらくルートまでのエッジの数が少なくて済むため？

（著者スライドを大幅に改変）

Qualitative Analysis

LSTMs vs. Tree-LSTMs:
How does structure help?

It 's actually **pretty good** in the first few minutes , **but** the longer the movie goes , the **worse** it gets .

LSTM	**Tree-LSTM**	**Gold**
–	–	–

What happens when the clauses are inverted?

LSTMs vs. Tree-LSTMs:
How does structure help?

The longer the movie goes , the **worse** it gets , **but**
it 's actually **pretty good** in the first few minutes .

LSTM	**Tree-LSTM**	**Gold**
+	–	–

LSTM prediction switches, but Tree-LSTM prediction does not!

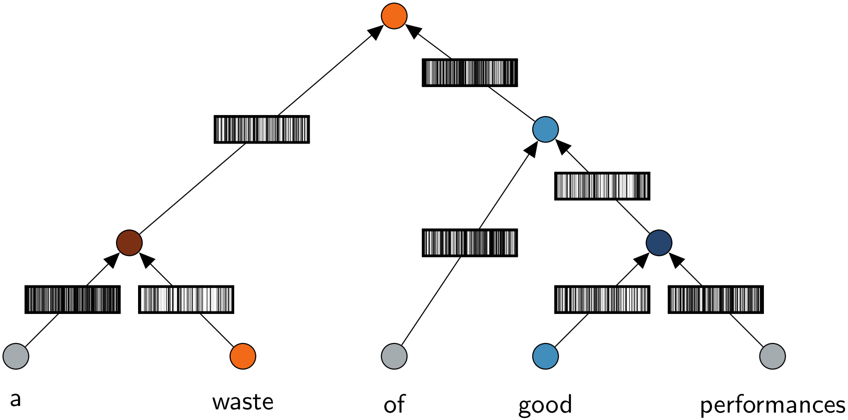
Either LSTM belief state is overwritten by last seen sentiment-rich word, *or* just always inverts the sentiment at "but".

LSTM vs. Tree-LSTM: Hard Cases in Sentiment

If Steven Soderbergh's 'Solaris' is a **failure** it is a **glorious failure**.

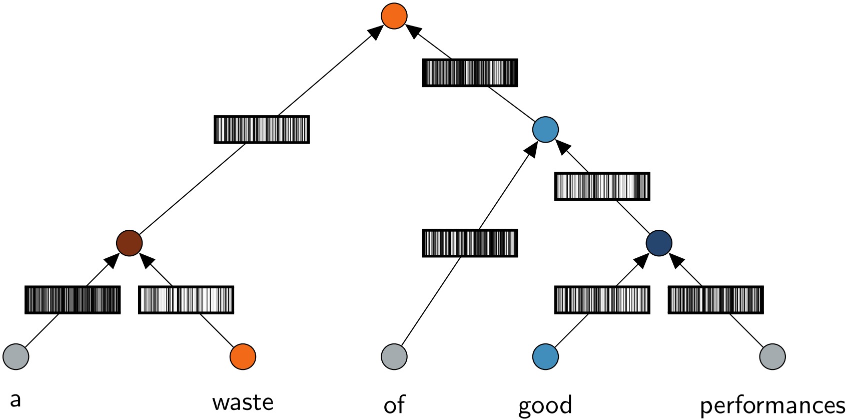
LSTM	Tree-LSTM	Gold
– –	– –	++

Forget Gates: Selective State Preservation



- Striped rectangles = forget gate activations
- More white $\Rightarrow$ more of that child's state is preserved

Forget Gates: Selective State Preservation

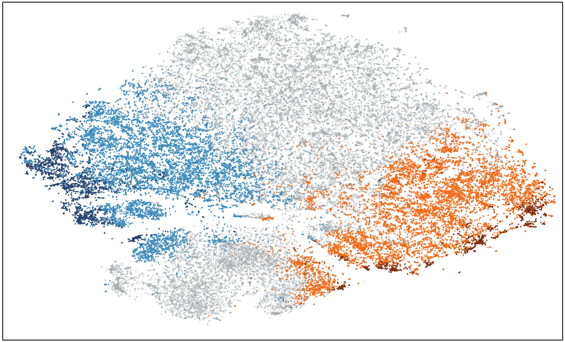


- States of sentiment-rich children are emphasized
  - e.g. “a” vs. “waste”
- “a waste” emphasized over “of good performances”

Conclusion

- We introduce Tree-LSTMs for composing distributed representations of sentences
- Tree-LSTMs outperform previous methods on sentiment, semantic similarity
- By making use of structural information, we can do better than standard sequential LSTMs

Thanks



(t-SNE visualization of Tree-LSTM phrase and sentence representations on the Stanford Sentiment Treebank)

Code

`github.com/stanfordnlp/treelstm`

Contact

Kai Sheng Tai `kst@metamind.io`