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
Distributed Word Representations

- Representations of words as real-valued vectors
- Now seemingly ubiquitous in NLP
Word vectors and meaning

ice

vs.

snow
But what about the meaning of sentences?

the snowboarder is leaping over snow

vs.

a person who is snowboarding jumps into the air
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 $\phi$
- Steps:
  1. Choose some compositional order for a sentence
     - e.g. sequentially left-to-right
  2. Recursively apply $\phi$ until representation for entire sentence is obtained
- We want to learn $\phi$ 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 $\phi$
- 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

\[ \text{Input/output/forget gates: vectors in } [0, 1]^d \]

\[ \text{Multiplied elementwise ("soft masking")} \]

\[ \text{Intuition: Selective memory read/write, selective information propagation} \]
Sequential Composition:
(Simplified) step-by-step LSTM composition

Step $t$

Step $t + 1$
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 $t$**
Can we do better?
Can we do better?

- Sentences have additional structure beyond word-ordering

- This is additional information that we can exploit
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
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” \(\sim\) “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
## Sentiment Classification Results

<table>
<thead>
<tr>
<th>Method</th>
<th>5-class</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNTN (Socher et al., 2013)</td>
<td>45.7</td>
<td>85.4</td>
</tr>
<tr>
<td>Paragraph-Vec (Le &amp; Mikolov, 2014)</td>
<td>48.7</td>
<td>87.8</td>
</tr>
<tr>
<td>Convolutional NN (Kim 2014)</td>
<td>47.4</td>
<td><strong>88.1</strong></td>
</tr>
<tr>
<td>Epic (Hall et al., 2014)</td>
<td>49.6</td>
<td>–</td>
</tr>
<tr>
<td>DRNN (Irsoy &amp; Cardie, 2014)</td>
<td>49.8</td>
<td>86.6</td>
</tr>
<tr>
<td>LSTM</td>
<td>46.4</td>
<td>84.9</td>
</tr>
<tr>
<td>Bidirectional LSTM</td>
<td>49.1</td>
<td>87.5</td>
</tr>
<tr>
<td><strong>Constituency Tree-LSTM</strong></td>
<td><strong>51.0</strong></td>
<td>88.0</td>
</tr>
</tbody>
</table>

- **Metric:** Binary/5-class accuracy
- * = Our own benchmarks
## Semantic Relatedness Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Pearson’s $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word vector average</td>
<td>0.758</td>
</tr>
<tr>
<td>Meaning Factory (Bjerva et al., 2014)</td>
<td>0.827</td>
</tr>
<tr>
<td>ECNU (Zhao et al., 2014)</td>
<td>0.841</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.853</td>
</tr>
<tr>
<td>Bidirectional LSTM</td>
<td>0.857</td>
</tr>
<tr>
<td>Dependency Tree-LSTM</td>
<td>0.868</td>
</tr>
</tbody>
</table>

- **Metric:** Pearson correlation with gold annotations (higher is better)
- $\star = $ Our own benchmarks
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.

<table>
<thead>
<tr>
<th>LSTM</th>
<th>Tree-LSTM</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>LSTM</th>
<th>Tree-LSTM</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>--</td>
<td>++</td>
</tr>
</tbody>
</table>
Forget Gates: Selective State Preservation

- Striped rectangles = forget gate activations
- More white ⇒ 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