Do we need recursive subtree composition in dependency parsing?

Miryam de Lhoneux

@mdlhx

UPPSALA
UNIVERSITET

10 December 2019
Workshop on Data-driven Approaches to Parsing and Semantic Composition
Overview

1. Tree vs. sequential LSTMs for parsing
2. BiLSTM parsing
3. Composition in a BiLSTM-parser
4. Composition for Auxiliary Verb Constructions
5. Conclusion
Outline for section 1

1. Tree vs. sequential LSTMs for parsing
2. BiLSTM parsing
3. Composition in a BiLSTM-parser
4. Composition for Auxiliary Verb Constructions
5. Conclusion
Recursive vs recurrent NNs

The largest city in Minnesota

Recurrent subtree composition in parsing
Recursive vs recurrent NNs

Recursive subtree composition in parsing
the largest city
Recursive NN for Transition-Based Parsing

the largest city

left-arc

nmod
Recursive NN for Transition-Based Parsing

Recursive subtree composition in parsing
Recursive composition function in the stack-LSTM parser (Dyer et al., 2015):
Recursive composition function in the stack-LSTM parser (Dyer et al., 2015):

\[ c(h, d, r) = \tanh(W[h; d; r] + b) \]
Recursive composition function in the stack-LSTM parser (Dyer et al., 2015):

\[ c(h, d, r) = \tanh(W[h; d; r] + b) \]
\[ h_i = c(h_{i-1}, d, r) \]
Recursive composition function in the stack-LSTM parser (Dyer et al., 2015):

\[ c(h, d, r) = \tanh(W[h; d; r] + b) \]

\[ city_1 = c(city_0, largest, left - nmod) \]
Recursive composition function in the stack-LSTM parser (Dyer et al., 2015):

\[ c(h, d, r) = \tanh(W[h; d; r] + b) \]

\[ city_1 = c(city_0, largest, left - nmod) \]

\[ city_2 = c(city_1, the, left - det) \]
Recursive vs recurrent NNs

<table>
<thead>
<tr>
<th>Method</th>
<th>English</th>
<th>Chinese</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-LSTM without composition</td>
<td>89.6</td>
<td>83.6</td>
</tr>
<tr>
<td>S-LSTM with composition</td>
<td>90.9</td>
<td>85.7</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>91.2</td>
<td>85.0</td>
</tr>
<tr>
<td>Model</td>
<td>English PTB</td>
<td>Chinese CTB</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Recursive subtree composition</td>
<td>6/38</td>
<td></td>
</tr>
</tbody>
</table>
## Recursive vs recurrent NNs

<table>
<thead>
<tr>
<th></th>
<th>English PTB</th>
<th>Chinese CTB</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-LSTM without comp.</td>
<td>89.6</td>
<td>83.6</td>
</tr>
</tbody>
</table>

Miryam de Lhoneux

Recursive subtree composition in parsing
### Recursive vs recurrent NNs

<table>
<thead>
<tr>
<th></th>
<th>English PTB</th>
<th>Chinese CTB</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-LSTM without composition</td>
<td>89.6</td>
<td>83.6</td>
</tr>
<tr>
<td>S-LSTM with composition</td>
<td>90.9</td>
<td>85.7</td>
</tr>
</tbody>
</table>
## Recursive vs recurrent NNs

<table>
<thead>
<tr>
<th>Model</th>
<th>English PTB</th>
<th>Chinese CTB</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-LSTM without composition</td>
<td>89.6</td>
<td>83.6</td>
</tr>
<tr>
<td>S-LSTM with composition</td>
<td>90.9</td>
<td>85.7</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>91.2</td>
<td>85.0</td>
</tr>
</tbody>
</table>
Recursive Subtree Composition in LSTM-Based Dependency Parsing

Miryam de Lhoneux†  Miguel Ballesteros‡  Joakim Nivre†
† Department of Linguistics and Philology, Uppsala University
‡ IBM Research AI, Yorktown Heights, NY
Recursive Subtree Composition in LSTM-Based Dependency Parsing

Miriam de Lhoneux ♠ Miguel Ballesteros ◇ Joakim Nivre ♠
♠ Department of Linguistics and Philology, Uppsala University
◇ IBM Research AI, Yorktown Heights, NY
Goals

**Recursive Subtree Composition in LSTM-Based Dependency Parsing**

Miryam de Lhoneux ♣ Miguel Ballesteros ◇ Joakim Nivre ♣
♣ Department of Linguistics and Philology, Uppsala University
◇ IBM Research AI, Yorktown Heights, NY

- BiLSTM + composition?
Goals

Recursive Subtree Composition in LSTM-Based Dependency Parsing

Miryam de Lhoneux ♣ Miguel Ballesteros ◇ Joakim Nivre ♣
♣ Department of Linguistics and Philology, Uppsala University
◇ IBM Research AI, Yorktown Heights, NY

- BiLSTM + composition?
- Examine composition in simple architecture
Goals

Recursive Subtree Composition in LSTM-Based Dependency Parsing

Miryam de Lhoneux, Miguel Ballesteros, Joakim Nivre

Department of Linguistics and Philology, Uppsala University
IBM Research AI, Yorktown Heights, NY

- BiLSTM + composition?
- Examine composition in simple architecture
- Typologically diverse languages
Goals

What Should/Do/Can LSTMs Learn
When Parsing Auxiliary Verb Constructions?

Miryam de Lhoneux, Sara Stymne and Joakim Nivre
Goals

What Should/Do/Can LSTMs Learn
When Parsing Auxiliary Verb Constructions?

Miryam de Lhoneux, Sara Stymne and Joakim Nivre
What Should/Do/Can LSTMs Learn
When Parsing Auxiliary Verb Constructions?

Miryam de Lhoneux, Sara Stymne and Joakim Nivre

- Characterise what our parser learns about language
Goals

What Should/Do/Can LSTMs Learn When Parsing Auxiliary Verb Constructions?

Miryam de Lhoneux, Sara Stymne and Joakim Nivre

- Characterise what our parser learns about language
- Examine what our parser learns about auxiliary verb constructions (AVCs)
Characterise what our parser learns about language

Examine what our parser learns about auxiliary verb constructions (AVCs)

Investigate the role of composition for AVCs
Outline for section 2

1. Tree vs. sequential LSTMs for parsing
2. BiLSTM parsing
3. Composition in a BiLSTM-parser
4. Composition for Auxiliary Verb Constructions
5. Conclusion
Transition-Based Parsing using BiLSTMs

Configuration:

STACK

BUFFER

the brown fox jumped root

Scoring:

(score(LEFT-ARC), score(RIGHT-ARC), score(SHIFT), score(SWAP))

Kiperwasser and Goldberg (2016); de Lhoneux et al. (2017)
Xthe
Transition-Based Parsing using BiLSTMs

Recursive subtree composition in parsing
Transition-Based Parsing using BiLSTMs

X_the   X_brown   X_fox   X_jumped   X_root
Transition-Based Parsing using BiLSTMs

Recursive subtree composition in parsing
Transition-Based Parsing using BiLSTMs

\[
\begin{align*}
&\text{Vthe} \\
&\text{Vfox} \\
&\text{Vbrown} \\
&\text{Vjumped} \\
&\text{Vroot}
\end{align*}
\]

\[
\begin{align*}
&\text{concat} \\
&\text{concat} \\
&\text{concat} \\
&\text{concat} \\
&\text{concat}
\end{align*}
\]

\[
\begin{align*}
&\text{LSTM}_f \\
&\text{LSTM}_f \\
&\text{LSTM}_f \\
&\text{LSTM}_f \\
&\text{LSTM}_f
\end{align*}
\]

\[
\begin{align*}
&\text{X}_{\text{the}} \\
&\text{X}_{\text{brown}} \\
&\text{X}_{\text{fox}} \\
&\text{X}_{\text{jumped}} \\
&\text{X}_{\text{root}}
\end{align*}
\]
Transition-Based Parsing using BiLSTMs

Recursive subtree composition in parsing
Transition-Based Parsing using BiLSTMs

(score(LEFT−ARC),score(RIGHT−ARC),score(SHIFT),score(SWAP))

Miryam de Lhoneux

Recursive subtree composition in parsing
Recursive Composition in the BiLSTM parser

Miryam de Lhoneux
Recursive Composition in the BiLSTM parser

Miryam de Lhoneux
Recursive subtree composition in parsing
13/38
Recursive Composition in the BiLSTM parser

Recursive subtree composition in parsing
Recursive Composition in the BiLSTM parser

![Diagram of recursive composition in BiLSTM parser]

- **Concatenation**: \( \text{concat} \)
- **LSTM**
  - Input: \( \{X_{the}, X_{brown}, X_{fox}, X_{jumped}, X_{root}\} \)
  - Output: \( \{Cthe, Cbrown, Cfox, Vthe, Vbrown, Vfox, Vjumped, Vroot\} \)
- **Forward LSTM** (\( \text{LSTM}^f \))
- **Backward LSTM** (\( \text{LSTM}^b \))

**Example Sentence**: "The brown fox jumped root"
Recursive Composition in the BiLSTM parser

\[ C_{\text{fox}} = \tanh(W[C_{\text{fox}}, C_{\text{brown}}, \text{left−nmod}]+b) \]

Vthe \hspace{1cm} Vthe \hspace{1cm} Vbrown \hspace{1cm} Vbrown \hspace{1cm} Vfox \hspace{1cm} Cfox

concat \hspace{1cm} concat \hspace{1cm} concat \hspace{1cm} concat \hspace{1cm} concat \hspace{1cm} concat

LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM

LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM \hspace{1cm} LSTM

X_{\text{the}} \hspace{1cm} X_{\text{brown}} \hspace{1cm} X_{\text{fox}} \hspace{1cm} X_{\text{jumped}} \hspace{1cm} X_{\text{root}}
Recursive Composition in the BiLSTM parser

Miryam de Lhoneux

Recursive subtree composition in parsing
Recursive Composition in the BiLSTM parser

The diagram illustrates the process of recursive subtree composition in parsing. It shows the flow of information through the BiLSTM (Bidirectional Long Short-Term Memory) network, with LSTM (Long Short-Term Memory) units processing different parts of the input sequence. The diagram highlights how the network performs forward (f) and backward (b) passes to capture context from both ends of the sequence, facilitating a more comprehensive understanding of the input.
Recursive Composition in the BiLSTM parser

\[ c_{head} = \tanh(W[h;d;r] + b) \]
Recursive Composition in the BiLSTM parser

\[ c_{head} = \text{tanh}(W[h; d; r] + b) + rc \]
$c_{head} = \tanh(W[h; d; r] + b) + rc$

$c_{head} = \text{LSTM}([h; d; r])$
\[ c_{head} = \tanh(W[h; d; r] + b) + rc \]
\[ c_{head} = \text{LSTM}([h; d; r]) + lc \]
Outline for section 3

1. Tree vs. sequential LSTMs for parsing
2. BiLSTM parsing
3. Composition in a BiLSTM-parser
4. Composition for Auxiliary Verb Constructions
5. Conclusion
Results: BiLSTM + composition

A. Greek
Basque
Chinese
Czech
English
Finnish
French
Hebrew
Japanese
Average

Error LAS

Miryam de Lhoneux
Recursive subtree composition in parsing
Results: BiLSTM + composition

Error LAS

A. Greek
Basque
Chinese
Czech
English
Finnish
French
Hebrew
Japanese
Average

Miryam de Lhoneux
Recursive subtree composition in parsing
Results: BiLSTM + composition
LSTM Feature Extractors

The figure illustrates the architecture of LSTM feature extractors in the context of recursive subtree composition in parsing.
LSTM Feature Extractors

LSTM \( b \)

LSTM \( f \)

\( X_{\text{the}} \)

\( X_{\text{brown}} \)

\( X_{\text{fox}} \)

\( X_{\text{jumped}} \)

\( X_{\text{root}} \)
LSTM Feature Extractors

The diagram illustrates the use of LSTM (Long Short-Term Memory) feature extractors in parsing. Each word is processed through a series of bidirectional LSTMs, with outputs denoted as $X_{the}$, $X_{brown}$, $X_{fox}$, $X_{jumped}$, and $X_{root}$. The process starts with the words 'the', 'brown', 'fox', 'jumped', and 'root' respectively, and continues with bidirectional LSTMs. The final output is labeled 'bw'.
LSTM Feature Extractors

The figure illustrates the use of LSTM feature extractors in a parsing context. The LSTM layers process the input sequence of words (the, brown, fox, jumped, root) through forward (fw) paths, generating the output features $X_{the}$, $X_{brown}$, $X_{fox}$, $X_{jumped}$, and $X_{root}$. This process is crucial for understanding the recursive composition in parsing.
Results: BiLSTM ablations

Error LAS

A. Greek  Basque  Chinese  Czech  English  Finnish  French  Hebrew  Japanese  Average
Results: BiLSTM ablations

A. Greek
Basque
Chinese
Czech
English
Finnish
French
Hebrew
Japanese
Average

Error LAS

Miryam de Lhoneux
Recursive subtree composition in parsing
Results: BiLSTM ablations

A. Greek
Basque
Chinese
Czech
English
Finnish
French
Hebrew
Japanese
Average

Error LAS

Miryam de Lhoneux
Recursive subtree composition in parsing
Results: BiLSTM ablations + composition

![Graph showing error LAS for different ablation combinations.]

- bi
- bw
- fw

Error LAS

Average
Results: BiLSTM ablations + composition

![Graph showing error LAS for different configurations]

- bi
- bi+rc
- bw
- bw+rc
- fw
- fw+rc

Miryam de Lhoneux
Results: BiLSTM ablations + composition

![Bar chart showing the error LAS for different models: bi, bi+rc, bi+lc, bw, bw+rc, bw+lc, fw, fw+rc, fw+lc. The x-axis represents the average, and the y-axis represents the error LAS. The chart compares the performance of these models, with the error LAS ranging from 0 to 30.]
Word representation

Xthe

e(the)  pe(the)

concat

Cb  Cb  Cb

Cf  Cf  Cf

t  h  e

Cb  Cb  Cb
Word representation
## Composition gap recovery

<table>
<thead>
<tr>
<th></th>
<th>[bw+lc]-bw</th>
<th>[fw+lc]-fw</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos+char+</td>
<td>1.4</td>
<td>0.6</td>
</tr>
<tr>
<td>pos+char-</td>
<td>1.3</td>
<td>0.6</td>
</tr>
<tr>
<td>pos-char+</td>
<td>1.6</td>
<td>0.7</td>
</tr>
<tr>
<td>pos-char-</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Average
## Composition gap recovery

<table>
<thead>
<tr>
<th></th>
<th>([bw+lc]-bw)</th>
<th>bi-bw</th>
<th>%rec.</th>
<th>([fw+lc]-fw)</th>
<th>bi-fw</th>
<th>%rec.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pos+char+</strong></td>
<td>1.4</td>
<td>1.6</td>
<td><strong>87.5</strong></td>
<td>0.6</td>
<td>6.3</td>
<td><strong>9.5</strong></td>
</tr>
<tr>
<td><strong>pos+char-</strong></td>
<td>1.3</td>
<td>1.8</td>
<td><strong>72.2</strong></td>
<td>0.6</td>
<td>6.6</td>
<td><strong>9.1</strong></td>
</tr>
<tr>
<td><strong>pos-char+</strong></td>
<td>1.6</td>
<td>1.9</td>
<td><strong>84.2</strong></td>
<td>0.7</td>
<td>7.3</td>
<td><strong>9.6</strong></td>
</tr>
<tr>
<td><strong>pos-char-</strong></td>
<td>2</td>
<td>3.1</td>
<td><strong>64.5</strong></td>
<td>1</td>
<td>8.7</td>
<td><strong>11.5</strong></td>
</tr>
</tbody>
</table>

Average
Subtree composition does not reliably help a BiLSTM transition-based parser.
Subtree composition does not reliably help a BiLSTM transition-based parser.

The backward part of the BiLSTM is crucial, especially for right-headed languages.
Conclusions from this study

- Subtree composition does not reliably help a BiLSTM transition-based parser
- The backward part of the BiLSTM is crucial, especially for right-headed languages
- The forward part of the BiLSTM is less crucial
Conclusions from this study

- Subtree composition does not reliably help a BiLSTM transition-based parser
- The backward part of the BiLSTM is crucial, especially for right-headed languages
- The forward part of the BiLSTM is less crucial
- A backward LSTM + subtree composition performs close to a BiLSTM
Conclusions from this study

- Subtree composition does not reliably help a BiLSTM transition-based parser
- The backward part of the BiLSTM is crucial, especially for right-headed languages
- The forward part of the BiLSTM is less crucial
- A backward LSTM + subtree composition performs close to a BiLSTM
- POS information and subtree composition are two partially redundant ways of constructing contextual information
The study suggests that recursive composition does not help.
Broader perspective

- This study: recursive composition does not help
- Falenska and Kuhn (2019): structural features do not help
Broader perspective

- This study: recursive composition does not help
- Falenska and Kuhn (2019): structural features do not help
- Gontrum (2019): attention does not help
This study: recursive composition does not help
Falenska and Kuhn (2019): structural features do not help
Gontrum (2019): attention does not help

All information needed is in small set of token representations from stack and buffer
Broader perspective

- This study: recursive composition does not help
- Falenska and Kuhn (2019): structural features do not help
- Gontrum (2019): attention does not help

- All information needed is in small set of token representations from stack and buffer
- Token representations encode subtree information?
This study: recursive composition does not help
Falenska and Kuhn (2019): structural features do not help
Gontrum (2019): attention does not help

All information needed is in small set of token representations from stack and buffer
Token representations encode subtree information?
  Do we even need parsing algorithms? (Nivre, 2019)
Broader perspective

- This study: recursive composition does not help
- Falenska and Kuhn (2019): structural features do not help
- Gontrum (2019): attention does not help

- All information needed is in small set of token representations from stack and buffer
- Token representations encode subtree information?
  - Do we even need parsing algorithms? (Nivre, 2019)
  - Trees can be decoded directly from BERT contextual embeddings (Hewitt and Manning, 2019)
Broader perspective

Linzen et al. (2016) and Gulordava et al. (2018): LM LSTMs learn agreement

Ravfogel et al. (2018) and Ravfogel et al. (2019): Yes but using local heuristics

Something like that happening here?
Broader perspective

- Linzen et al. (2016) and Gulordava et al. (2018): LM LSTMs learn agreement
Broader perspective

- Linzen et al. (2016) and Gulordava et al. (2018): LM LSTMs learn agreement
- Ravfogel et al. (2018) and Ravfogel et al. (2019): Yes but using local heuristics
Broader perspective

- Linzen et al. (2016) and Gulordava et al. (2018): LM LSTMs learn agreement
- Ravfogel et al. (2018) and Ravfogel et al. (2019): Yes but using local heuristics
- Something like that happening here?
1. Tree vs. sequential LSTMs for parsing
2. BiLSTM parsing
3. Composition in a BiLSTM-parser
4. Composition for Auxiliary Verb Constructions
5. Conclusion
Dependency Parsing
Dependency Parsing

That could work
Auxiliary Verb Constructions

Dependency Parsing

That could work

Tesnière
Auxiliary Verb Constructions

Dependency Parsing

That could work

Tesnière

That could work
Dependency Parsing

That could work

Tesnière

That could — work

parsing |

Tesnière |
Dependency Parsing

That could work

Tesnière

That could — work

Unit of syntax

parsing

Tesnière
Dependency Parsing

That could work

Tesnière

That could — work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dependency Parsing

That could work

Tesnière

That could ___ work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>nucleus</td>
<td></td>
</tr>
</tbody>
</table>
Auxiliary Verb Constructions

Dependency Parsing

That could work

Tesnière

That could work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>nucleus</td>
<td></td>
</tr>
<tr>
<td>Relations between words</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dependency Parsing

That could work

Tesnière

That could work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td></td>
<td>nucleus</td>
</tr>
<tr>
<td>Relations between words</td>
<td>dependency</td>
<td></td>
</tr>
</tbody>
</table>
Dependency Parsing

That could work

Tesnière

That could work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>nucleus</td>
<td></td>
</tr>
</tbody>
</table>

| Relations between words | dependency | dependency, transfer, junction |
Auxiliary Verb Constructions

Dependency Parsing

Tesnière

That could work

That could — work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>dependency, transfer, junction</td>
<td></td>
</tr>
<tr>
<td>nucleus</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

UD compatible with Tesnière
Dependency Parsing

That could work

Tesnière

That could — work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>dependency</td>
<td>nucleus</td>
</tr>
<tr>
<td>dependency</td>
<td>dependency, transfer, junction</td>
<td></td>
</tr>
</tbody>
</table>

UD compatible with Tesnière
But parsers don’t know that
Dependency Parsing

That could work

Tesnière

That could work

<table>
<thead>
<tr>
<th>Unit of syntax</th>
<th>parsing</th>
<th>Tesnière</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>dependency</td>
<td>nucleus</td>
</tr>
<tr>
<td>Relations between words</td>
<td>dependency, transfer, junction</td>
<td></td>
</tr>
</tbody>
</table>

UD compatible with Tesnière
But parsers don’t know that
Or do they?
Research questions

That could work
Research questions

That could work

That works
Research questions

Do LSTM-based parsers learn the notion of dissociated nucleus?
Research questions

Do LSTM-based parsers learn the notion of dissociated nucleus? Dissociated nucleus ~ nucleus
Research questions

Do LSTM-based parsers learn the notion of dissociated nucleus?

Dissociated nucleus $\sim$ nucleus

Diagnostic classifier
Do LSTM-based parsers learn the notion of dissociated nucleus?
Dissociated nucleus $\sim$ nucleus
Diagnostic classifier
Research questions

Do LSTM-based parsers learn the notion of dissociated nucleus?
Dissociated nucleus $\sim$ nucleus
Diagnostic classifier
Sharon has lost his patience.

- Sharon: nsubj
- has: aux
- Number=3
- Person=Sing
- lost: obj
- his: nmod
- patience: nmod
Sharon has lost his patience.

AVC
Sharon has lost his patience

Number=3
Person=Sing

AVC
Sharon has lost his patience

Transitivity: has object? True/False
Sharon has lost his patience

Transitivity: has object? True/False
Sharon has lost his patience

Agreement: Person + Number (sg/pl + 1/2/3)
Sharon has lost his patience

Number=3
Person=Sing

Agreement: Person + Number (sg/pl + 1/2/3)
Vectors
Vectors
Vectors

Recursive subtree composition in parsing
Vectors
## Dataset

<table>
<thead>
<tr>
<th></th>
<th>FMV train</th>
<th>FMV dev</th>
<th>punct train</th>
<th>punct dev</th>
<th>AVC train</th>
<th>AVC dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalan</td>
<td>14K</td>
<td>2K</td>
<td>7K</td>
<td>964</td>
<td>12K</td>
<td>2K</td>
</tr>
<tr>
<td>Croatian</td>
<td>6K</td>
<td>803</td>
<td>4K</td>
<td>491</td>
<td>5K</td>
<td>653</td>
</tr>
<tr>
<td>Dutch</td>
<td>9K</td>
<td>618</td>
<td>6K</td>
<td>516</td>
<td>5K</td>
<td>251</td>
</tr>
<tr>
<td>Finnish</td>
<td>12K</td>
<td>1K</td>
<td>9K</td>
<td>1K</td>
<td>4K</td>
<td>458</td>
</tr>
<tr>
<td>Catalan</td>
<td>14K</td>
<td>2K</td>
<td>7K</td>
<td>964</td>
<td>12K</td>
<td>2K</td>
</tr>
<tr>
<td>Croatian</td>
<td>6K</td>
<td>803</td>
<td>4K</td>
<td>491</td>
<td>5K</td>
<td>653</td>
</tr>
<tr>
<td>Dutch</td>
<td>9K</td>
<td>618</td>
<td>6K</td>
<td>516</td>
<td>5K</td>
<td>246</td>
</tr>
<tr>
<td>Finnish</td>
<td>10K</td>
<td>1K</td>
<td>8K</td>
<td>850</td>
<td>4K</td>
<td>443</td>
</tr>
</tbody>
</table>
Agreement
Agreement and Transitivity in FMVs

Agreement

did

FMV
Agreement and Transitivity in FMVs

Agreement

Transitivity

did
FMV
Agreement and Transitivity in FMVs

Agreement

Transitivity

Miryam de Lhoneux
Agreement and Transitivity in FMVs
Agreement and Transitivity in FMVs

Agreement

Transitivity

Miryam de Lhoneux

Recursive subtree composition in parsing
Vectors AVC

Miryam de Lhoneux
Recursive subtree composition in parsing
Vectors AVC

Recursive subtree composition in parsing
Vectors AVC

Recursive subtree composition in parsing
Recursive subtree composition in parsing
Agreement and Transitivity in AVCs vs FMVs

Agreement

Transitivity

Recursive subtree composition in parsing
Agreement and Transitivity in AVCs vs FMVs

**Agreement**

- done
- did
- FMV
- NFMV

**Transitivity**

- did
- FMV
- done
- NFMV
Agreement and Transitivity in AVCs vs FMVs

Agreement

Transitivity

Miryam de Lhoneux

Recursive subtree composition in parsing
Agreement and Transitivity in AVCs vs FMVs

Agreement

Transitivity

Recursive subtree composition in parsing

Miryam de Lhoneux
Agreement and Transitivity in AVCs vs FMVs

**Agreement**

```
Agreement

Transitivity
```
Agreement and Transitivity in AVCs vs FMVs

**Agreement**

**Transitivity**

Recursive subtree composition in parsing
Conclusions from this study

- Our parser does not learn the notion of dissociated nucleus
Conclusions from this study

- Our parser does not learn the notion of dissociated nucleus
- Composition helps learning this
Outline for section 5

1. Tree vs. sequential LSTMs for parsing
2. BiLSTM parsing
3. Composition in a BiLSTM-parser
4. Composition for Auxiliary Verb Constructions
5. Conclusion
General conclusions

Conclusions

- Composition does not help accuracy of a BiLSTM parser.
General conclusions

Conclusions

- Composition does not help accuracy of a BiLSTM parser
- Composition might be useful to learn relations of transfer
General conclusions

Conclusions

- Composition does not help accuracy of a BiLSTM parser
- Composition might be useful to learn relations of **transfer**

Future Work

- Token vectors encode subtrees or parser uses heuristics?
## Conclusions
- Composition does not help accuracy of a BiLSTM parser
- Composition might be useful to learn relations of transfer

## Future Work
- Token vectors encode subtrees or parser uses heuristics?
- LSTMs vs Transformer
Thanks!

Thanks!
References


