Recursive Subtree Composition in LSTM-Based Dependency Parsing

Miryam de Lhoneux, Miguel Ballesteros and Joakim Nivre
@mdlhx

UPPSALA
UNIVERSITET

4 June 2019

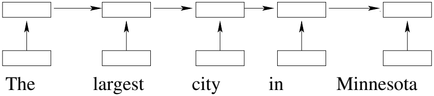
Overview

1. Tree vs. sequential LSTMs for parsing
2. BiLSTM parsing
3. Results
4. Conclusion

Outline for section 1

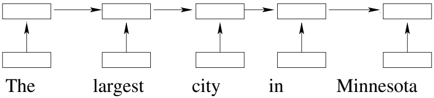
Recursive vs recurrent NNs

Recurrent

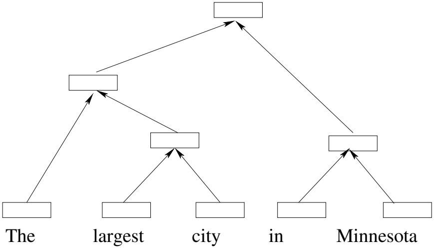


Recursive vs recurrent NNs

Recurrent



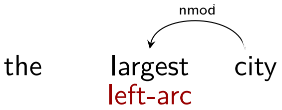
Recursive



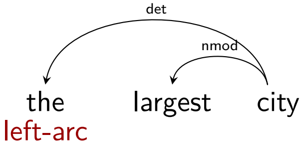
Recursive NN for Transition-Based Parsing

the largest city

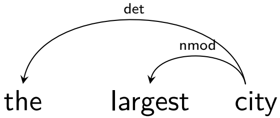
Recursive NN for Transition-Based Parsing



Recursive NN for Transition-Based Parsing

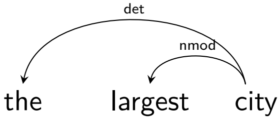


Recursive NN for Transition-Based Parsing



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

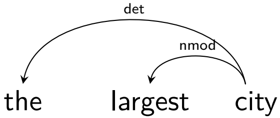
Recursive NN for Transition-Based Parsing



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

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

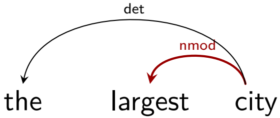
Recursive NN for Transition-Based Parsing



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

$$c(h, d, r) = tanh(W[h; d; r] + b)$$
$$h_i = c(h_{i-1}, d, r)$$

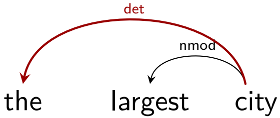
Recursive NN for Transition-Based Parsing



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 NN for Transition-Based Parsing



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

$$
\begin{aligned}
c(h, d, r) &= tanh(W[h; d; r] + b) \\
city_1 &= c(city_0, largest, left - nmod) \\
city_2 &= c(city_1, the, left - det)
\end{aligned}
$$

Recursive vs recurrent NNs

Recursive vs recurrent NNs

	English PTB	Chinese CTB

Recursive vs recurrent NNs

	English PTB	Chinese CTB
S-LSTM without composition	89.6	83.6

Recursive vs recurrent NNs

	English PTB	Chinese CTB
S-LSTM without composition	89.6	83.6
S-LSTM with composition	90.9	85.7

Recursive vs recurrent NNs

	English PTB	Chinese CTB
S-LSTM without composition	89.6	83.6
S-LSTM with composition	90.9	85.7
BiLSTM	91.2	85.0

Recursive vs recurrent NNs

	English PTB	Chinese CTB
S-LSTM without composition	89.6	83.6
S-LSTM with composition	90.9	85.7
BiLSTM	91.2	85.0

Goals

Recursive vs recurrent NNs

	English PTB	Chinese CTB
S-LSTM without composition	89.6	83.6
S-LSTM with composition	90.9	85.7
BiLSTM	91.2	85.0

Goals

- BiLSTM + composition?

Recursive vs recurrent NNs

	English PTB	Chinese CTB
S-LSTM without composition	89.6	83.6
S-LSTM with composition	90.9	85.7
BiLSTM	91.2	85.0

Goals

- BiLSTM + composition?
- Examine composition in simple architecture

Recursive vs recurrent NNs

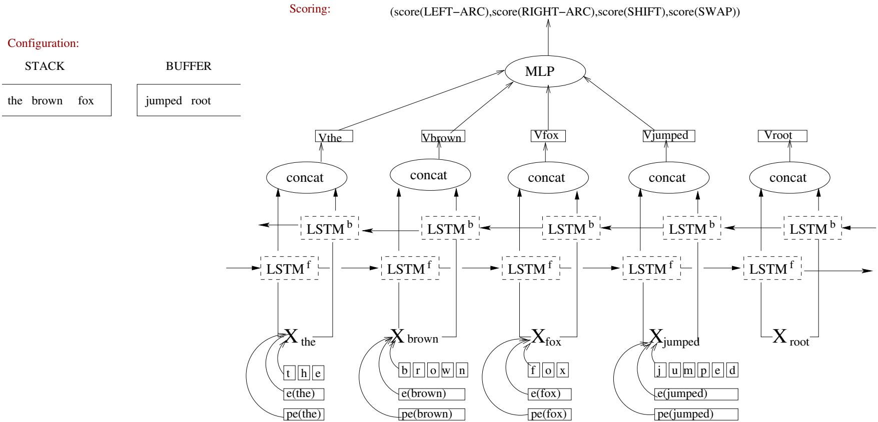
	English PTB	Chinese CTB
S-LSTM without composition	89.6	83.6
S-LSTM with composition	90.9	85.7
BiLSTM	91.2	85.0

Goals

- BiLSTM + composition?
- Examine composition in simple architecture
- Typologically varied languages

Outline for section 2

Transition-Based Parsing using BiLSTMs

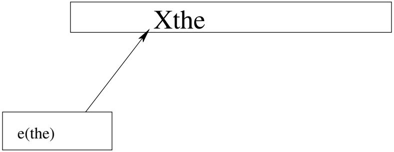


Kiperwasser and Goldberg (2016); de Lhoneux et al. (2017)

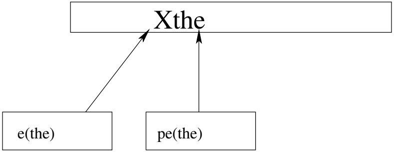
Transition-Based Parsing using BiLSTMs

Xthe

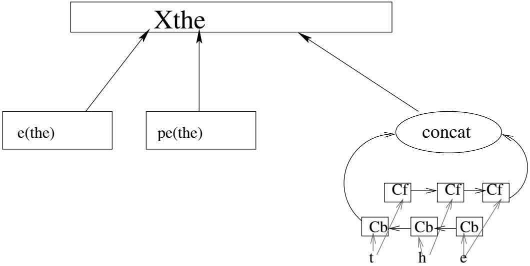
Transition-Based Parsing using BiLSTMs



Transition-Based Parsing using BiLSTMs



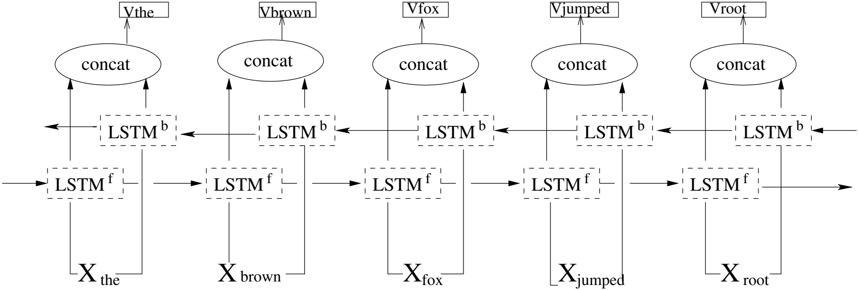
Transition-Based Parsing using BiLSTMs



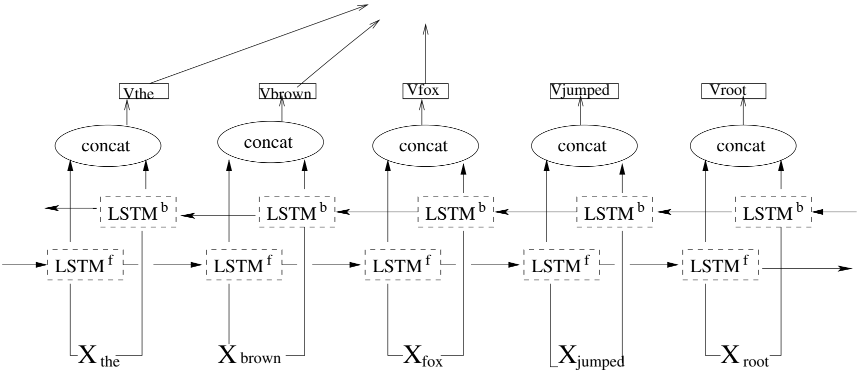
Transition-Based Parsing using BiLSTMs

X_{the} X_{brown} X_{fox} X_{jumped} X_{root}

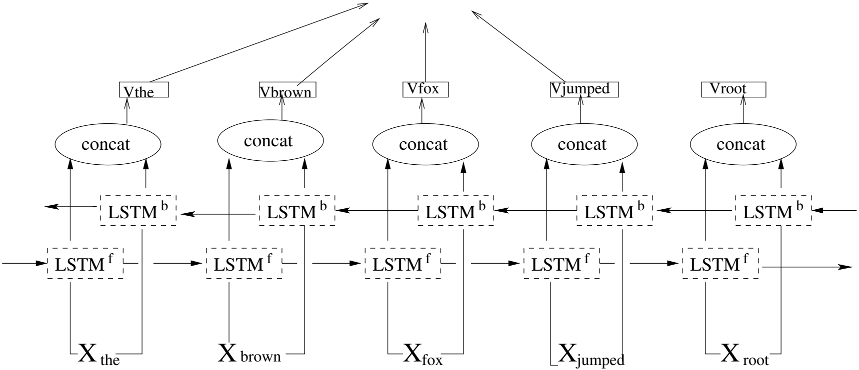
Transition-Based Parsing using BiLSTMs



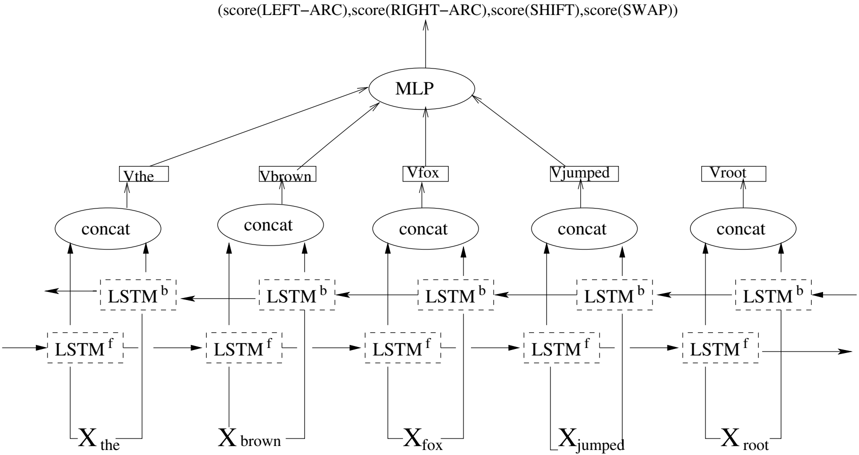
Transition-Based Parsing using BiLSTMs



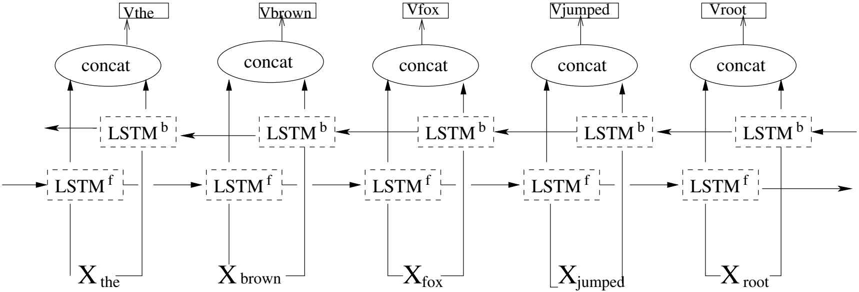
Transition-Based Parsing using BiLSTMs



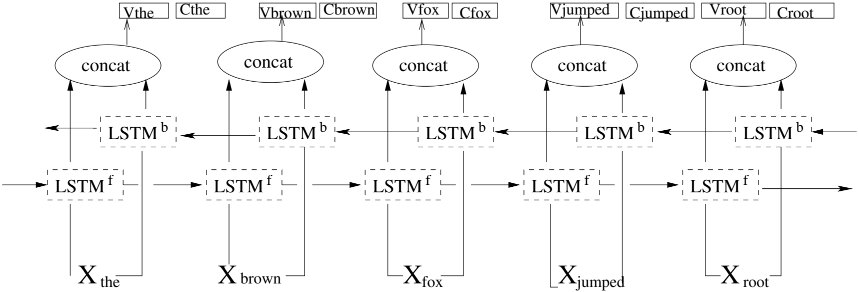
Transition-Based Parsing using BiLSTMs



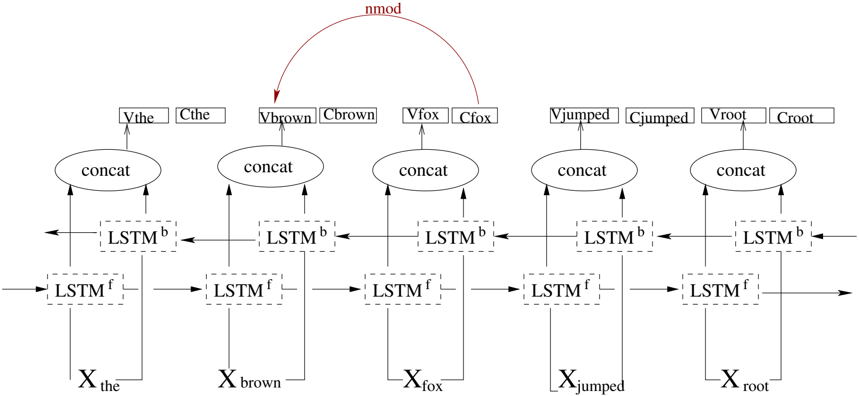
Recursive Composition in the BiLSTM parser



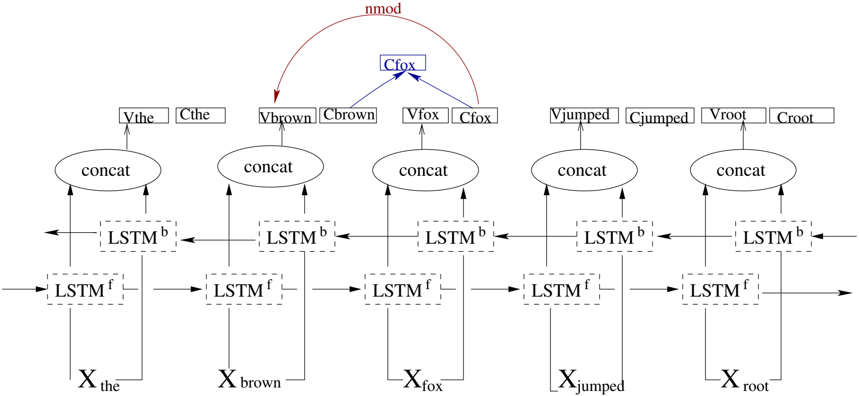
Recursive Composition in the BiLSTM parser



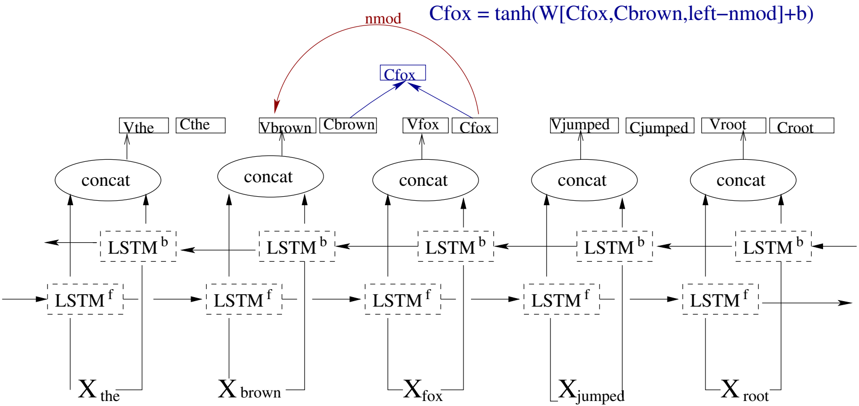
Recursive Composition in the BiLSTM parser



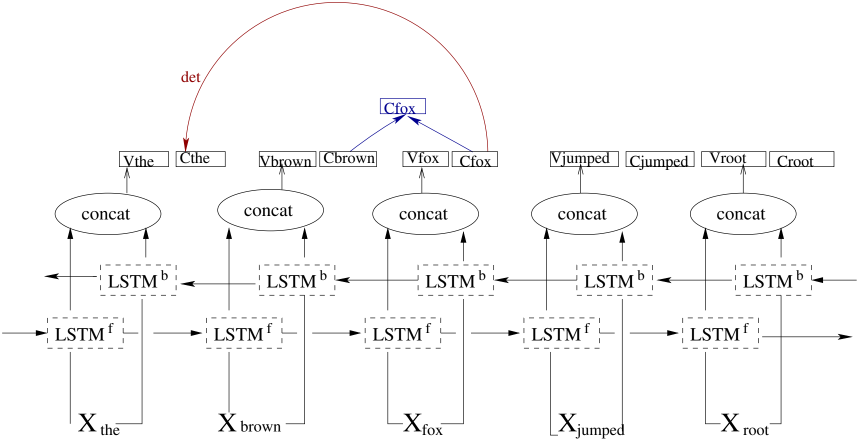
Recursive Composition in the BiLSTM parser



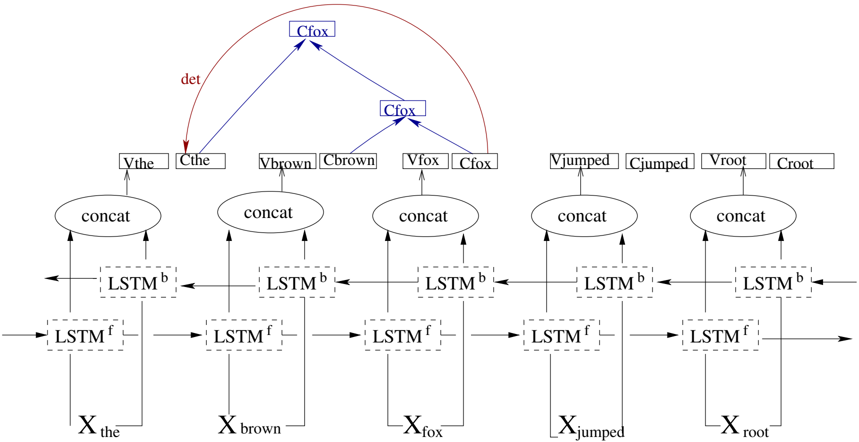
Recursive Composition in the BiLSTM parser



Recursive Composition in the BiLSTM parser



Recursive Composition in the BiLSTM parser



Recursive Composition in the BiLSTM parser

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

Recursive Composition in the BiLSTM parser

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

Recursive Composition in the BiLSTM parser

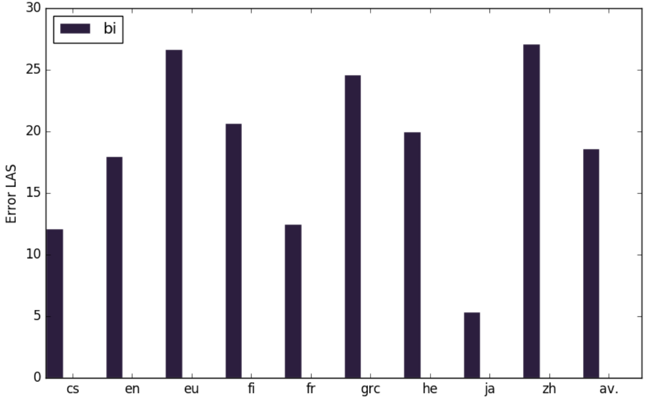
$$c_{head} = tanh(W[h; d; r] + b) + rc$$
$$c_{head} = \textsc{Lstm}([h; d; r])$$

Recursive Composition in the BiLSTM parser

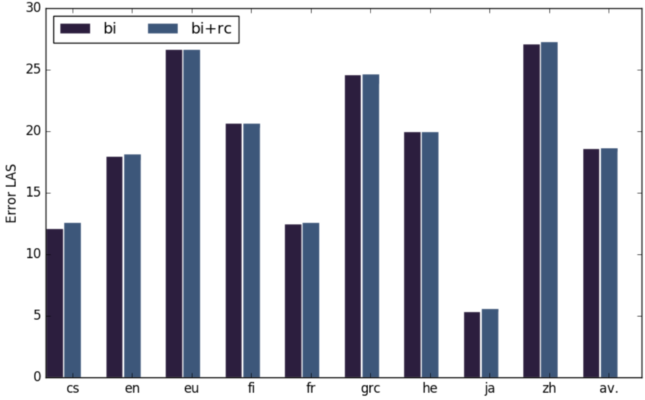
$$c_{head} = tanh(W[h; d; r] + b) + rc$$
$$c_{head} = \textsc{Lstm}([h; d; r]) + lc$$

Outline for section 3

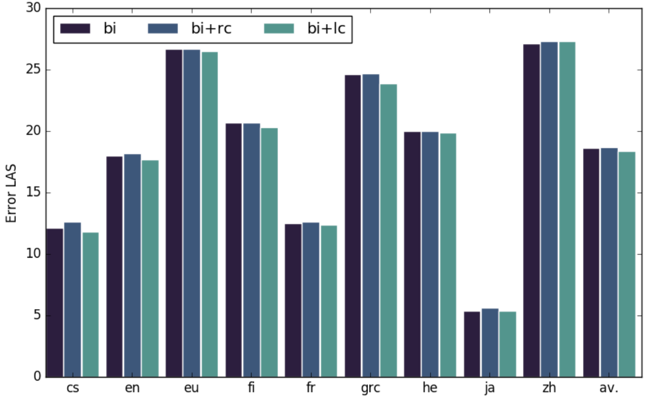
Results: BiLSTM + composition



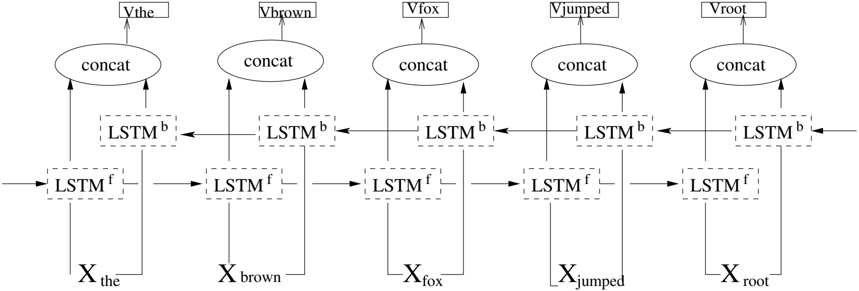
Results: BiLSTM + composition



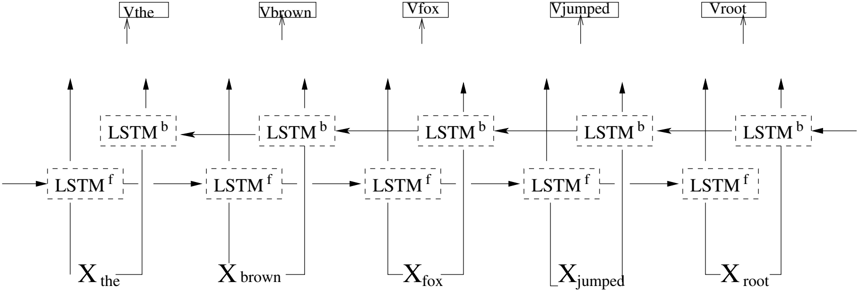
Results: BiLSTM + composition



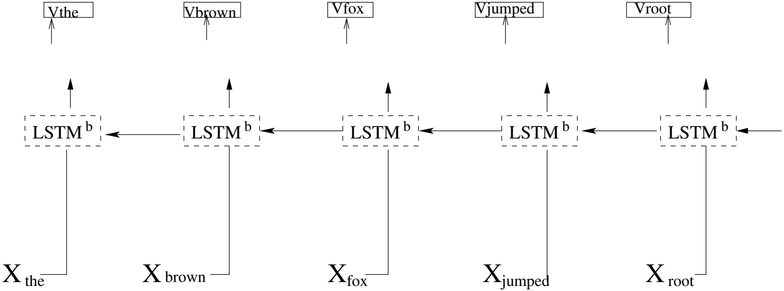
LSTM Feature Extractors



LSTM Feature Extractors

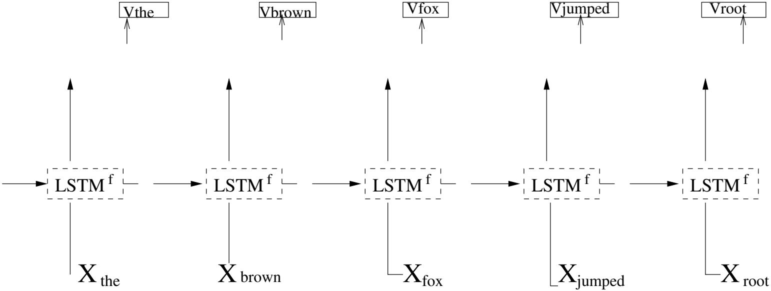


LSTM Feature Extractors



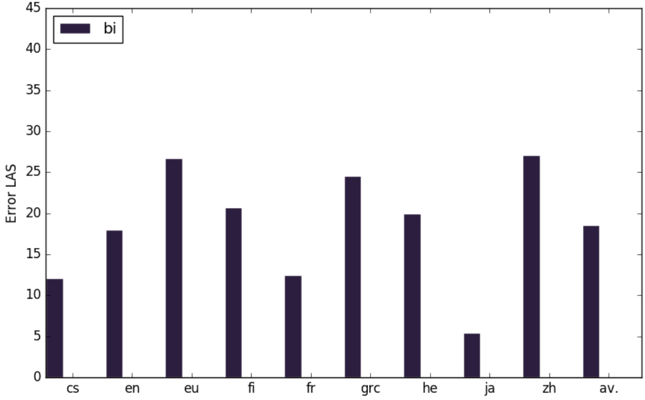
bw

LSTM Feature Extractors

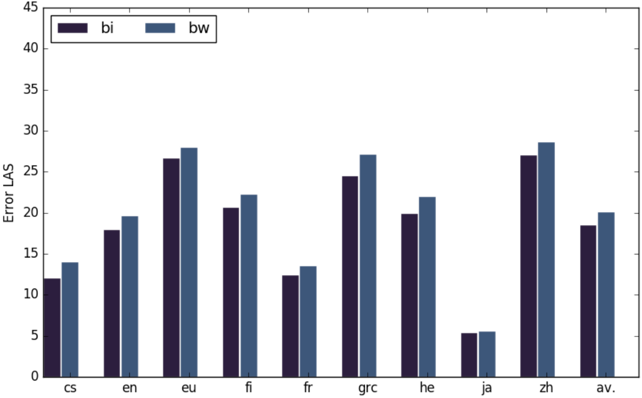


fw

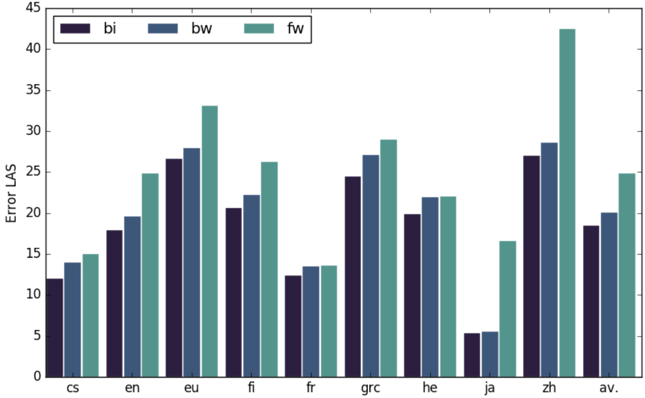
Results: BiLSTM ablations



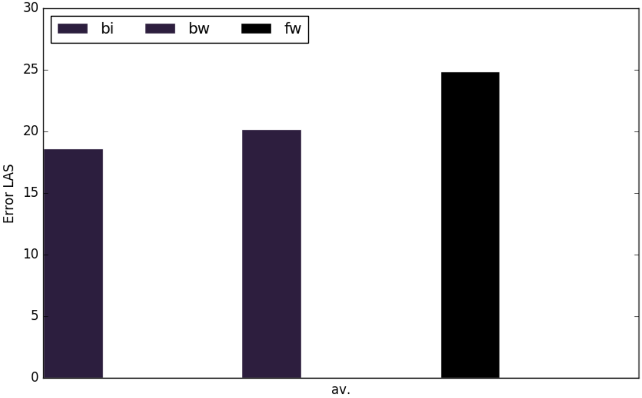
Results: BiLSTM ablations



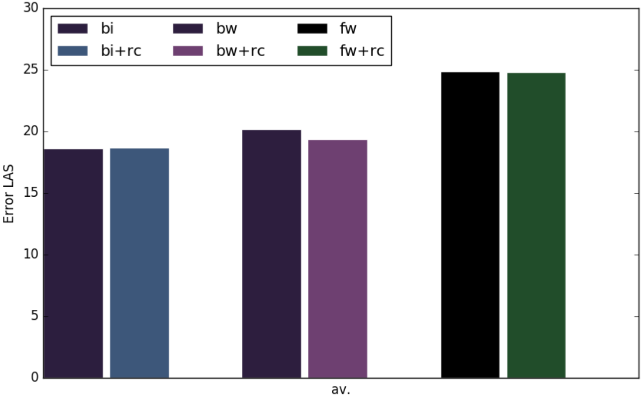
Results: BiLSTM ablations



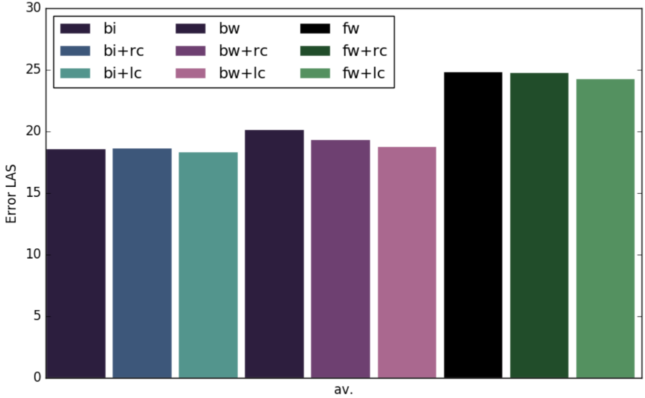
Results: BiLSTM ablations + composition



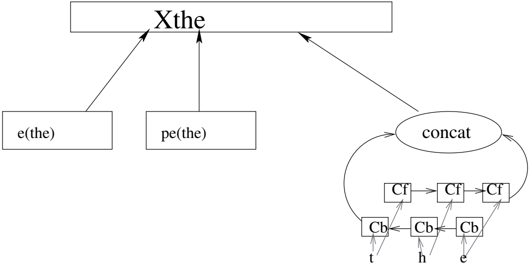
Results: BiLSTM ablations + composition



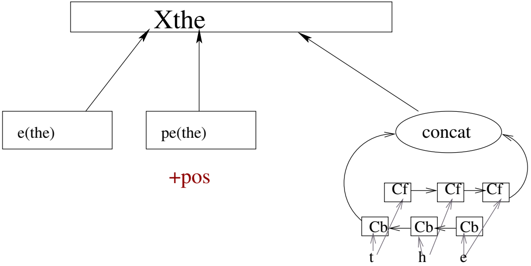
Results: BiLSTM ablations + composition



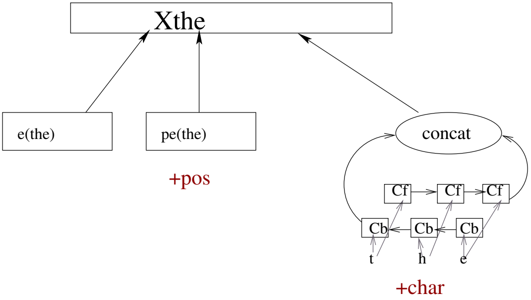
Word representation



Word representation



Word representation



Composition gap recovery

	[bw+lc]-bw	[fw+lc]-fw
pos+char+	1.4	0.6
pos+char-	1.3	0.6
pos-char+	1.6	0.7
pos-char-	2	1

av.

Composition gap recovery

	[bw+lc]-bw	bi-bw	%rec.	[fw+lc]-fw	bi-fw	%rec.
pos+char+	1.4	1.6	87.5	0.6	6.3	9.5
pos+char-	1.3	1.8	72.2	0.6	6.6	9.1
pos-char+	1.6	1.9	84.2	0.7	7.3	9.6
pos-char-	2	3.1	64.5	1	8.7	11.5

av.

Outline for section 4

Conclusion

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

Conclusion

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

Conclusion

- 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

Conclusion

- 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

Conclusion

- 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

References

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017. From raw text to universal dependencies - look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 207–217.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 334–343. http://aclweb.org/anthology/P/P15/P15-1033.pdf.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.