# Taking Advantage of BiLSTM Encoding to Handle Punctuation in Dependency Parsing: A Brief Idea

Matteo Grella
matteogrella@gmail.com

January 5, 2018

**Abstract**

In the context of the bidirectional-LSTMs neural parser (Kiperwasser and Goldberg, 2016), an idea is proposed to initialize the parsing state without punctuation-tokens but using them for the BiLSTM sentence encoding. The relevant information brought by the punctuation-tokens should be implicitly learned using the errors of the recurrent contributions only.

I assume the reader is familiar with the formal framework of transition-based dependency parsing originally introduced by Nivre (2003) [1] and the most recent deep-learning techniques applied to the natural language processing; see Goldberg (2017) [2] for an introduction.

Traditional data-driven transition-based dependency parsers analyze punctuation as words. Experimental results showed that parsing accuracy drops on sentences which contain higher ratios of punctuation. The problem is that punctuation is not as consistently annotated in tree-banks as words, and this makes the learning and therefore the parsing processes harder.

Indeed, certain aspects of punctuation are merely stylistic, and punctuation dependencies are a bit hard to defend compared to word-to-word dependency, so that in the standard CoNLL evaluation the arcs leading to the punctuation tokens just do not count.

However, it is out of the question that punctuation (especially commas) plays an essential role in the syntactic analysis as it is vital to resolve many attachment ambiguities. Furthermore, it can completely alter the meaning of a sentence.

In the context of dependency parsing, Ma et al. (2014) [3] suggest treating punctuation as properties of its neighboring words, without involving the punctuation itself directly in any attachment. In their model, before parsing starts, a pre-processing step is used to first set the punctuation marks as properties of their neighboring words, and then remove them from the sentence. During parsing, each transition that creates a dependency relation causes these properties to be propagated from a token to its governor, producing additional features to guide the parser to build the dependency graph.

Ma et al. show that their method reduces errors-propagation[1] and improves parsing performance of about 0.90% UAS over a greedy baseline parser on the English Penn Tree-bank. Grella (2015) [4] obtained comparable results for the Italian language.

I think that a similar strategy could be implemented efficiently in the scheme for dependency parsing based on bidirectional-LSTMs proposed by Kiperwasser and Goldberg (2016) [5]. In that model, each sentence token is associated with a BiLSTM vector representing the token with its surrounding context. The *next best transition* is obtained by using a multi-layer perceptron whose input features are constructed by concatenating a few of these BiLSTM vectors, based on the current configuration of the parsing state.

The **gist of my idea** is to initialize the parsing state without punctuation-tokens but using them for the BiLSTM sentence encoding.[2] In this way, the punctuation will never be directly involved in either the features construction or any attachment, but since the BiLSTM is trained jointly with the parser-objective, **the relevant information brought by the punctuation-tokens should be implicitly learned using the errors of the recurrent contributions only**.

Besides, it might be possible to let the BiLSTM indicate the best head of a punctuation-token, using a sort of attention mechanism to evaluate on which previous/next word it had the overall strongest influence.[3]

Future thoughts can be made to exploit this idea to perform parsing and punctuation prediction jointly.

I encourage researchers to try to implement this technique inside the open-source BIST-parsers.[4]

# References


[1] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*. Citeseer, 2003.

[2] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[3] Ji Ma, Yue Zhang, and Jingbo Zhu. Punctuation processing for projective dependency parsing. In *ACL (2)*, pages 791–796, 2014.

[4] Matteo Grella. Notes about a more aware dependency parser. *arXiv preprint arXiv:1507.05630*, 2015.


[1] A wrong attachment in punctuation may require non-projective transitions to continue the analysis correctly. Even for transition systems that support non-projective, crossing-links remain more subject to errors due to their relative rarity.

[2] In the arc-standard model the initial parsing state would be an empty stack and the input buffer filled with all the sentence tokens except for punctuation marks. The *oracle* shall consider that some tokens will never be reduced.

[3] It might make sense to replace the LSTM [6] with the RAN [7] to take advantage of its highly interpretable outputs.

[4] https://github.com/elikip/bist-parser

[5] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327, 2016.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Kenton Lee, Omer Levy, and Luke Zettlemoyer. Recurrent additive networks. *arXiv preprint arXiv:1705.07393*, 2017.