Height-Deterministic Target Languages and the Encoder-Decoder Parsing Model

Anssi Yli-Jyrä (Helsinki Centre for Digital Humanities – HELDIG, University of Helsinki)

Poster Presented at LxLMS 2019 (the 9th Lisbon Machine Learning School)

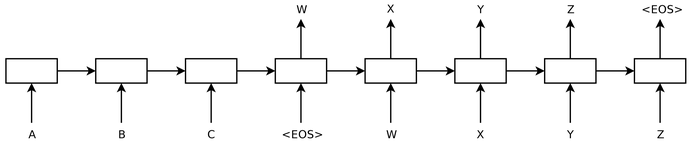
UNIVERSITY OF HELSINKI

Abstract

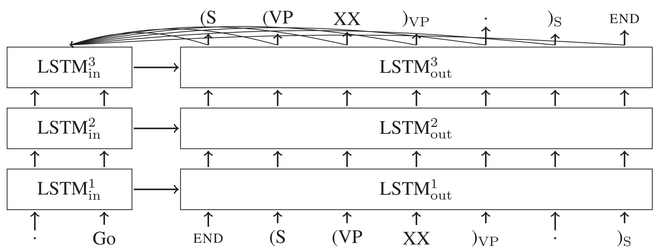
Motivated by Google's *Grammar as a Foreign Language* model (Vinyals et al., 2015) where trees are represented as linear strings, the poster aims at a generalisation for graphs. The poster outlines a framework for studying the **encoder-decoder parsing model** and its simplification – **parsing as sequence labeling** via the simultaneous analysis of the attention mechanism and the linearisation scheme. According to our hypothesis, this is achieved by relating the linearised graphs and the behaviour of the decoder to verious transition systems. The outlined framework is important since the set of labeled ordered graphs contains almost all linguistically motivated structures that can be assigned to natural language or to its translation relations.

Constituency Structure Parsing via Encoder-Decoder Model

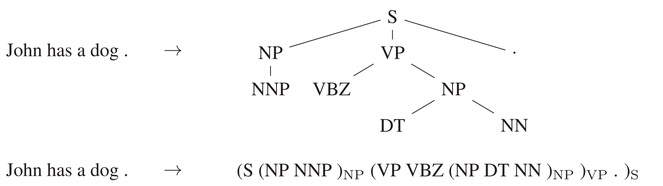
The encoder-decoder model (Sutskever et al., 2014) is an extension of the LSTM model (Hochreiter and Schmidhuber, 1997) to sequence translation tasks. In the decoder network of the model, the output of the previous time step becomes the current input. The model was used to translate from a source language to a target language:



The LSTM+A model (Bahdanau et al., 2015) improves the memory of the decoder network with an attention mechanism. This mechanism retrieves, for each time moment, a weighted sum of the hidden states of the encoder, and concatenates the sum with the input of the decoder.



Vinyals et al. (2015) used LSTM+A (see on the above) to produce a linearisation of the constituency structure of the source sentence as the target sentence:



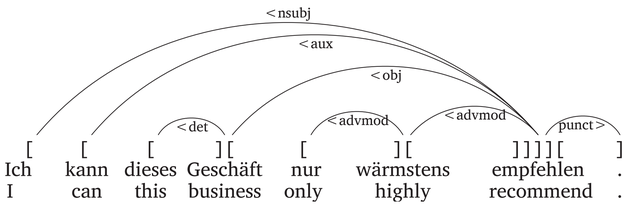
Vinyals et al. (2015) demonstrate that the attention mask is like a pointer that steps to the right every time a terminal node is consumed.

Hypothesis

Since the output of the decoder LSTM is similar to a visibly pushdown language (Alur and Madhusudan, 2009), we hypothesize that the decoder network learns to behave like an automaton that parses deterministic pushdown languages, while the encoder network extends this in a way that resembles an inverse homomorphism, a regular lookahead, and a regular closure (Nederhof and Bertsch, 1996).

Bracketing of Noncrossing Dependency Graphs

The output of the decoder has been varied for dependency parsing. Yli-Jyrä and Gómez-Rodríguez (2017) developed a linearisation for noncrossing dependency trees and graphs:



The linearisation of undirected graphs extends with dependency relations:

```
[<nsubj·[<aux·[<det·][<obj·[<advmod·][<advmod·]]][·]punct>
```

The bracket sequence can be interpreted as a sequence of actions that manipulate a buffer and a stack and produce a dependency graph:

action	brackets	effect
shift	·	advance the buffer head
predict	[	stack.push(the buffer head)
complete	]	*edges* ← *edges* ∪ {(stack.pop(),the buffer head)}

Strzyz et al. (2019) implemented a BiLSTM/LSTM encoder-decoder and used various linearisations of the dependency structure, including the bracketing. The benefits of the approach:

- conceptual simplicity
- a good speed-accuracy tradeoff
- results are competitive with more complex approaches.

Streamlined Bracketing

Yli-Jyrä (2017) presented a streamlined bracketing for noncrossing graphs:

original `[ · [ · [ · ][ · [ · ][ · ]]]][ · ]`
streamlined `⟦ · [ · ⟦ · ⟧[ · ⟦ · ⟧[ · ⟧⟦ · ⟧`

The streamlined bracket sequence can be interpreted as a sequence of actions that manipulate a buffer and a stack as follows:

action	brackets	effect
noshift	·	advance the buffer head
shift	⟦·	stack.push(the buffer head,{the buffer head}); noshift
reduce	⟧	*edges* ← *edges* ∪ {(*i*,the buffer head)| *i* ∈ *stack*.*pop*()[1]}
left	]	*edges* ← *edges* ∪ {(stack.top()[0],the buffer head)}
right	[	stack.top[1] ← stack.top[1] ∪ {the buffer head}

Our first aim is to use the streamlined bracketing to carry out parsing elegantly in the encoder-decoder (Vinyals et al., 2015; Strzyz et al., 2019) or sequence labeling (Gómez-Rodríguez and Vilares, 2018) approaches.

Generalized Linearisation

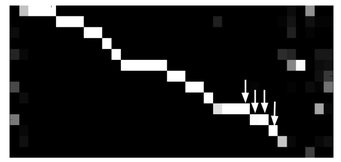
Both bracketing schemes and the corresponding transition systems are restricted to noncrossing graphs, but Strzyz et al. (2019) explored some more general encoding schemes for trees. **Our second aim** is to develop a similar linearisation for dependency graphs via a nontrivial generalization of the prior bracketing schemes. This should offer

- compatible with the streamlined bracketing
- full coverage of dependency annotations for natural language
- conceptual simplicity via the enc-dec model and the sequence labeling task.

The details of the the generalized encoding will be described elsewhere.

Interpretability

If the hypothesis is true, the internals of decoder resembles an automaton that parses deterministic context-free languages, when trained with a corresponding target annotation. This hypothesis was already supported by the analysis of the attention mechanism in Vinyals et al. (2015), indicating the incremental attention:



More general graph structures that are typical for semantic parsing of natural language require a general linearisation scheme and a transition system that extends a visibly pushdown automata. It remains to be seen how the structure of the transition system is reflected by the attention mask and the decoder network.

A further intriguing question is whether a pure LSTM-based decoder could learn to simulate the actions of a transition system that is controlled by an RNN grammar or stack LSTM (Dyer et al., 2015).

Wide Applicability

As all graphs can be linearized with the generalized scheme, the resulting linear strings can be used to represent also sparse translation alignments and multilinear structures. This lets us to generalize the encoder-decoder parsing model and to analyse the decoder networks in the same uniform research framework.

References

R. Alur and P. Madhusudan. 2009. **Adding Nesting Structure to Words**. *J. ACM* 56(3):16:1–16:43. https://doi.org/10.1145/1516512.1516518.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. **Neural machine translation by jointly learning to align and translate**. In *ICLR 2015*. http://arxiv.org/abs/1409.0473.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. 2015. **Transition-Based Dependency Parsing with Stack Long Short-Term Memory**. In *ACL/IJCNLP 2015*. https://www.aclweb.org/anthology/P15-1033.

C. Gómez-Rodríguez and D. Vilares. 2018. **Constituent parsing as sequence labeling**. In *EMNLP 2018*. Association for Computational Linguistics, Brussels, Belgium. https://www.aclweb.org/anthology/D18-1162.

S. Hochreiter and J. Schmidhuber. 1997. **Long short-term memory**. *Neural Computation* 9(8):1735–1780.

M.-J. Nederhof and E. Bertsch. 1996. **An innovative finite state concept for recognition and parsing of context-free languages**. *Natural Language Engineering* 2(4):381–382. https://doi.org/10.1017/S1351324997001666.

M. Strzyz, D. Vilares, and C. Gómez-Rodríguez. 2019. **Viable dependency parsing as sequence labeling**. In *NACL 2019*. Association for Computational Linguistics. https://www.aclweb.org/anthology/N19-1077.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. **Sequence to sequence learning with neural networks**. In *Advances on Neural Information Processing Systems*. http://dl.acm.org/citation.cfm?id=2969033.2969173.

O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. 2015. **Grammar as a foreign language**. In *Advances on Neural Information Processing Systems*.

A. Yli-Jyrä. 2017. **Bounded-depth high-coverage search space for noncrossing parses**. In *FSMNLP 2017*. https://www.aclweb.org/anthology/W17-4004.

A. Yli-Jyrä and C. Gómez-Rodríguez. 2017. **Generic axiomatization of families of noncrossing graphs in dependency parsing**. In *ACL 2017*. Vancouver, Canada, pages 1745–1755. https://doi.org/10.18653/v1/P17-1160.

anssi.yli-jyra@helsinki.fi