Extracting Syntactic Trees from Transformer Encoder Self-Attentions

David Mareček and **Rudolf Rosa**
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University, Prague, Czechia
{marecek,rosa}@ufal.mff.cuni.cz

1 Introduction

Interpreting neural networks is a popular topic, and there are many works focusing on analyzing networks with respect to learning syntax (Shi et al., 2016; Linzen et al., 2016; Blevins et al., 2018).

In particular, Vaswani et al. (2017) showed that the self-attentions in their Transformer architecture may be directly interpreted as syntactic dependencies between tokens. However, there is a potential problem in the fact that the attention mechanism on deeper layers operates on the previous-layer neurons, which already comprise mixed information from multiple source tokens.

Our goal is to infer source sentence tree structures form the encoder's self-attention energies used in the Transfomer neural machine translation (NMT) system. We would like to visualize how the self-attention mechanism connects individual words (or wordpieces) of the sentence, to create various tree structures (e.g. constituency trees, undirected trees, dependency trees), and to discuss their characteristics with respect to the existing syntactic theories and annotations. We would also like to discuss results across various languages and natural language processing (NLP) tasks.

In this abstract, we present our preliminary results, analyzing the encoder in English-to-German NMT within the NeuralMonkey toolkit (Helcl and Libovický, 2017). We introduce aggregation of self-attention through layers to get a distribution over the input tokens for each encoder position and layer (Section 2). We then propose algorithms for constructing two types of syntactic trees (Sections 3 and 4), apply them to 42 sentences sampled from PennTB (Marcus et al., 1993), and compare the resulting structures to established syntax annotation styles, such as that of PennTB, UD (Nivre et al., 2016), or PDT (Böhmová et al., 2003).

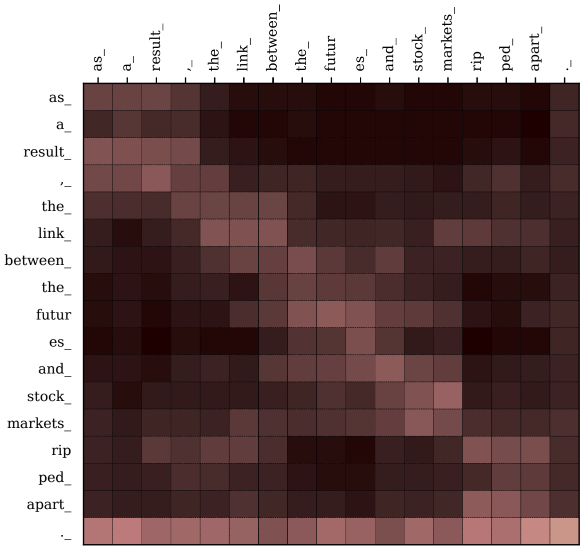

Figure 1: Aggregated encoder's self-attentions after the 6th layer. Each column contains a distribution over the source wordpieces for one encoder position.

2 Aggregated self-attention visualization

We use the default setting: encoder is composed of 6 layers, each consisting of a 16-head self-attention mechanism and a fully connected feed-forward network, both bridged by residual connections. Each position in one layer can attend to all positions in the previous layer; the attention to the same position is boosted by the residual connection. When translating a single sentence by Transformer, we would like to capture how much each input token affects each particular position on each layer in the encoder. This is done by aggregating the attention distributions through the layers. For each layer, we collect the self-attention distribution to the previous layer and add +1 to the same-position attention for the residual connection. The output is then normalized. So far, we take the attention distribution as the average attention over all the 16 heads.

Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 347–349
Brussels, Belgium, November 1, 2018. ©2018 Association for Computational Linguistics

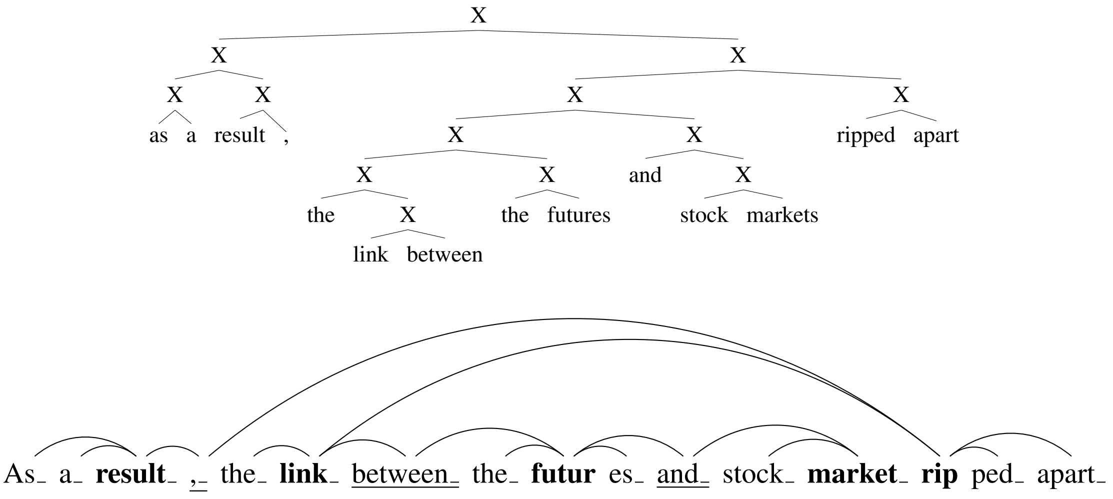
X
as a result ,
the link between
the futures
and
stock markets
ripped apart
As_ a_ result_ ,_ the_ link_ between_ the_ futur es_ and_ stock_ market_ rip ped_ apart_

Figure 2: A binary constituency tree and an undirected tree, generated by the proposed algorithms.

3 Constituency trees extraction

In Figure 1, we can see that the self-attention mechanism is quite strong within phrases. That led us to an idea of extracting phrase-structure trees from that. We define the score of a constituent with span from position i to position j as

$$score(i,j) = \frac{\sum_{x\in[i,\dots,j]} \sum_{y\in[i,\dots,j]} w[x,y]}{j-i+1},$$

where $w[x,y]$ is the attention weight of the token y in the position x. We then build a binary constituency tree by recurrently splitting the sentence. When splitting a phrase with span (i,j), we look for a position k maximizing the scores of the two resulting phrases:

$$\arg\max_k \left(score(i,k) \cdot score(k+1,j) \right).$$

We also rejoin wordpieces into words, assigning zero scores to constituents separating pieces of a single word. One example is shown in Figure 2.

When compared to PennTB, clauses, noun phrases, or shorter verb phrases are often well recognized. The differences are mainly inside them[1] and in composing them together forming clauses.

4 Undirected trees extraction

First, for each pair of tokens i, j, we calculate a *coattention score*, expressing how common it is for the tokens to be attended to at the same time:

$$score(i,j) = \sum_{m\in[1,N-1]} w[m,i] \cdot w[m,j]$$

We then construct an undirected tree [2] maximizing the coattention scores along its edges, using the algorithm of Kruskal (1956); see the bottom tree in Figure 2. We have found the resulting trees to bear surprising similarities to standard syntactic dependency trees (which, however, are directed).

For example, we observe many flat treelets, resembling headed syntactic phrases; the "phrase heads" (**bold**) are mostly content words, while the function words are mostly attached as leaf nodes (as in UD). We hypothesize that the encoder tries to concentrate the representation of the whole phrase onto the position of a single token – ideally one that already carries a lot of meaning.

Furthermore, the phrase treelets are then typically connected to each other via these heads (as in UD), and/or via a sort of connector tokens at phrase boundaries (underlined), such as commas, conjunctions, or prepositions (as in PDT).

5 Future work

In future, we would like to (1) analyze how the trees evolve through layers, (2) employ unsupervised or supervised selection of "more syntactic" heads, and (3) perform the experiments on more language pairs; especially, we hope that translation into multiple languages could push the encoder to use a more syntactic internal representation.

[1] This is also caused by very flat noun phrases representations in PennTB compared to our binary branching.

[2] We also tried to construct directed graphs, (i.e. a standard dependency tree), where the edges could be directly viewed as dependencies, but we did not find any sensible way of defining the coattention scores assymetrically; rather than parent-child dependencies, many relations seem to be more general, without a clear direction.

References

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19, Melbourne, Australia. Association for Computational Linguistics.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague dependency treebank. In *Treebanks*, pages 103–127. Springer.

Jindřich Helcl and Jindřich Libovický. 2017. Neural Monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17.

Joseph B Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Comput. Linguist.*, 19(2):313–330.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portoro, Slovenia. European Language Resources Association.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *EMNLP*, pages 1526–1534.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.