Constituency Parsing with a Self-Attentive Encoder

Nikita Kitaev and Dan Klein

Overview

State-of-the-art parsing results with self-attention and a novel method for factoring position and content information. Achieves over 95 F_1 on English!

Decoder

Structure

The score of a tree factors over labelled spans:

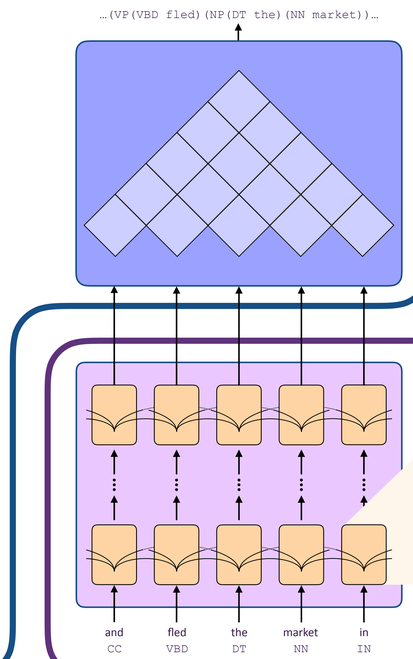
$$s(T) = \sum_{(i,j,l) \in T} s(i,j,l)$$

A feed-forward network computes span scores:

$$s(i,j,\cdot) = M_2 \max(\text{LayerNorm}(M_1 v + c_1), 0) + c_2$$

where $v = [\overrightarrow{y}_j - \overrightarrow{y}_i ; \overleftarrow{y}_{j+1} - \overleftarrow{y}_{i+1}]$

pools vectors near span endpoints



Objective

Train to have the correct tree outscore all others by a margin:

$$s(T^\star) \geq s(T) + \Delta(T, T^\star)$$

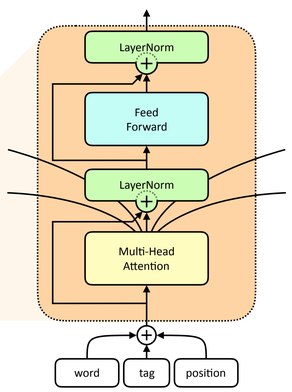
by minimizing a hinge loss for margin violation

Encoder

Architecture

8 layers of stacked self-attention, each followed by a position-wise feed-forward layer

Attention is the only means of information routing between words



Multi-Head Self-Attention

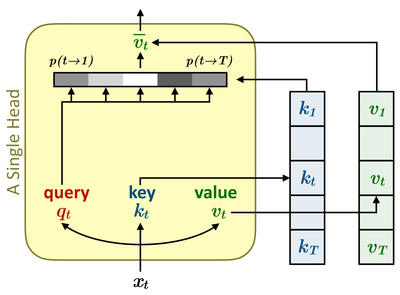
Each head can attend to a different target

Learned position embeddings allow for position-based information routing

$$\text{MultiHead}(X) = \sum_{n=1}^{8} \text{Head}^{(n)}(X)$$

$$\text{Head}^{(n)}(X) = \left[\text{Softmax}\left(\frac{Q^{(n)} K^{(n)\top}}{\sqrt{d_k}} \right) V^{(n)} \right] W_O^{(n)}$$

where $Q^{(n)} = X W_Q^{(n)};\ K^{(n)} = X W_K^{(n)};\ V^{(n)} = X W_V^{(n)}$



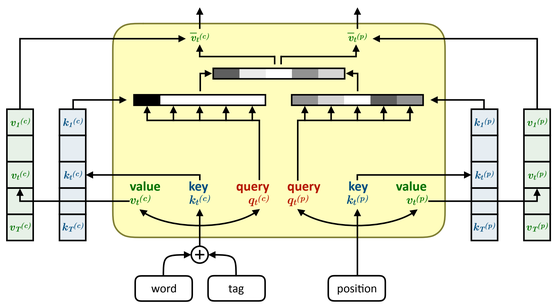
<u>Factored</u> Multi-Head Self-Attention

Vectors encoding position and content information are kept separate

Disallows a particular word (e.g. *the*) from attending to a fixed position (e.g. *word #3*)

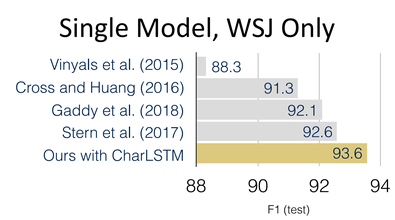
Another interpretation: weight matrix block-sparsity, halving the number of parameters

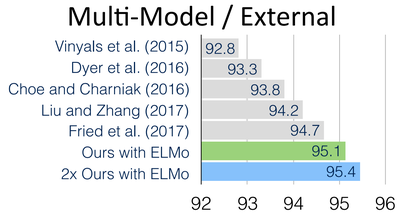
For all weight matrices W, enforce $W = \begin{bmatrix} W_{content} & 0 \\ 0 & W_{position} \end{bmatrix}$



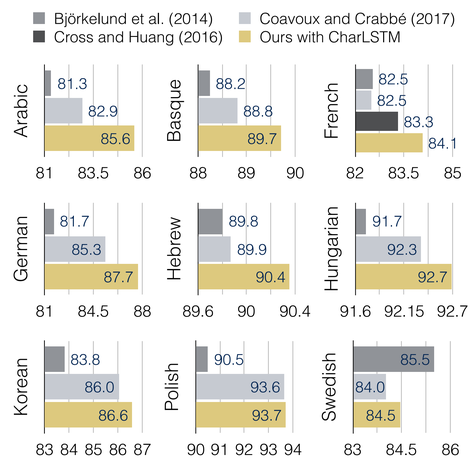
Results

English

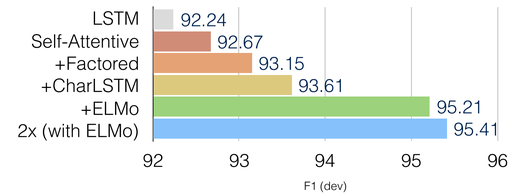




Multilingual (SPMRL)

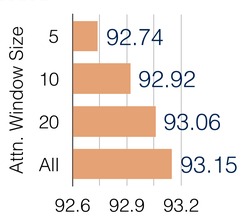


What Helps?



Long-Distance Context Matters

8 layers of self-attention with window size 10 have an effective receptive field of size 80. But the full, un-windowed model is still better!



Open Source Release

Code and models are publicly available

Includes integration with popular NLP toolkits: spaCy and NLTK

<u>Sample Usage (with spaCy)</u>

```
>>> import spacy
>>> from benepar.spacy_plugin import BeneparComponent
>>> nlp = spacy.load('en')
>>> nlp.add_pipe(BeneparComponent("benepar_en"))
>>> doc = nlp(u"Short cuts make long delays.")
>>> sent = list(doc.sents)[0]
>>> print(sent._.parse_string)
(S (NP (JJ Short) (NNS cuts)) (VP (VBP make) (NP (JJ
long) (NNS delays))) (. .))
>>> sent._.labels
('S',)
>>> list(sent._.children)[0]
Short cuts
```

<u>Sample Usage (with NLTK)</u>

```
>>> import benepar
>>> parser = benepar.Parser("benepar_en")
>>> tree = parser.parse("Short cuts make long delays.")
>>> print(tree)
(S
  (NP (JJ Short) (NNS cuts))
  (VP (VBP make) (NP (JJ long) (NNS delays)))
  (. .))
```

More at: `github.com/nikitakit/self-attentive-parser`