Tree Transformer: Integrating Tree Structures into Self-Attention

(Wang et al., EMNLP 2019)

出口 祥之 <deguchi@ai.cs.ehime-u.ac.jp>

2020/07/16 二宮研 論文輪読会

Links

Paper:

- https://www.aclweb.org/anthology/D19-1098/

Implementation:

- https://github.com/yaushian/Tree-Transformer

Introduction

- RNN では木構造を扱えるモデルが存在する (Tree-RNNs)
- Transformer ベースのモデルで直接木構造を扱うモデルはない
- シンプルなモジュールを追加するだけ
  - 実装が簡単
  - 教師なし構文解析で良い性能
  - 人間の直感とも一致するような階層構造が得られる
  - 説明可能な Attention を持つようになる

Tree Transformer

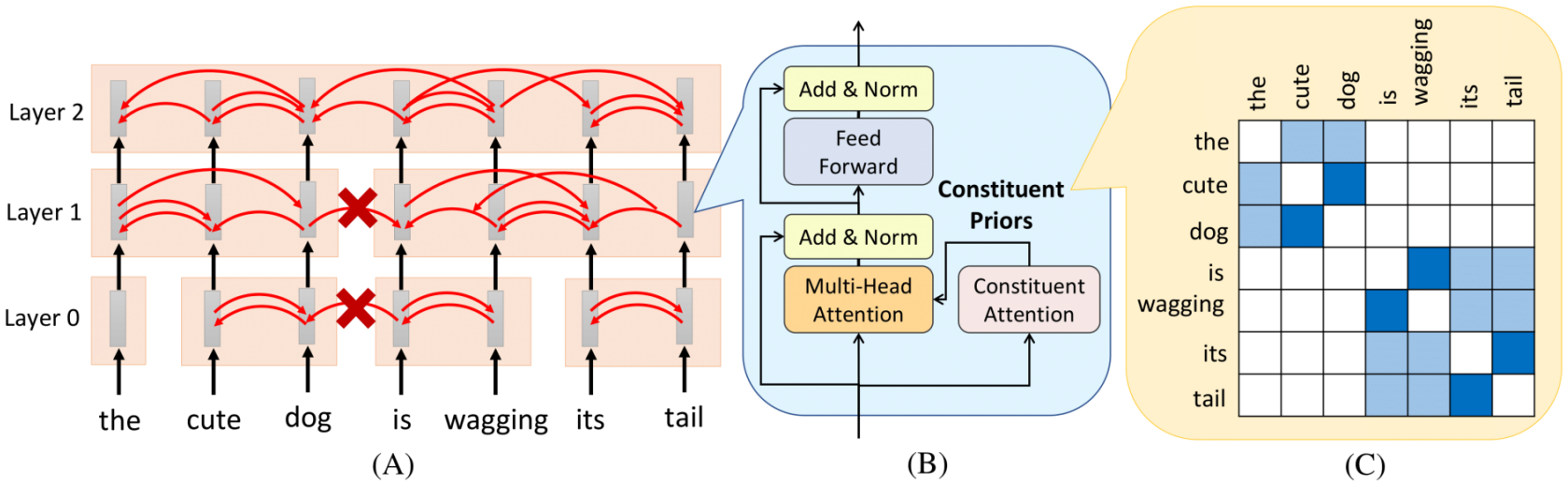


Figure 1: (A) A 3-layer Tree Transformer, where the blocks are constituents induced from the input sentence. The two neighboring constituents may merge together in the next layer, so the sizes of constituents gradually grow from layer to layer. The red arrows indicate the self-attention. (B) The building blocks of Tree Transformer. (C) Constituent prior C for the layer 1.

Constituent Prior

Attention 確率分布行列に Constituent Prior C を掛け合わせる

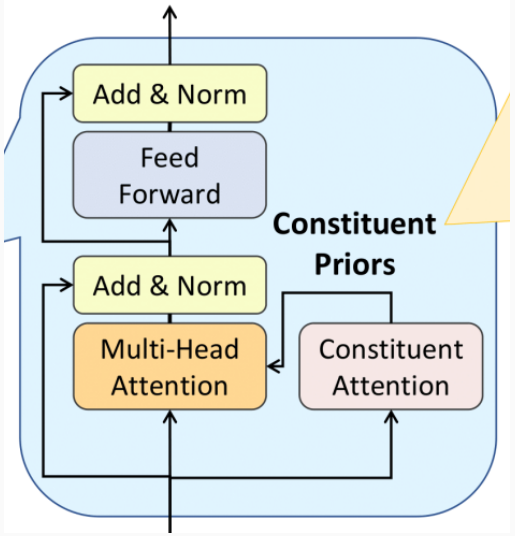
- 通常の Transformer の確率分布行列 E

$$E = \mathrm{softmax}(\frac{QK^\top}{\sqrt{d_k}})$$

- Tree Transformer の確率分布行列 E

$$E = C \odot \mathrm{softmax}(\frac{QK^\top}{\sqrt{d_k}})$$

※ $\odot$ は要素積



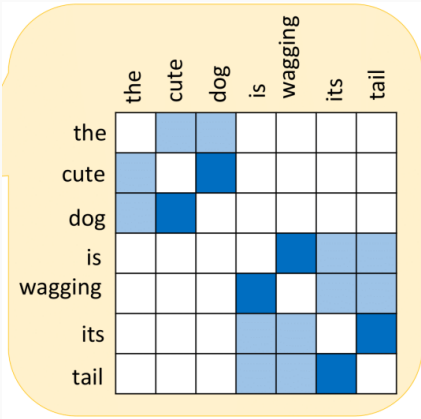
- C は Constituent Attention モジュール (後述) により計算される

Constituent Attention

C を計算するモジュール

- $C_{i,j} (= C_{j,i})$ は単語 w_i から単語 w_j まで (w_j から w_i まで) が同じ構成要素である確率
- 隣り合った単語が同じ構成要素である結合確率 a を後述の手法により計算

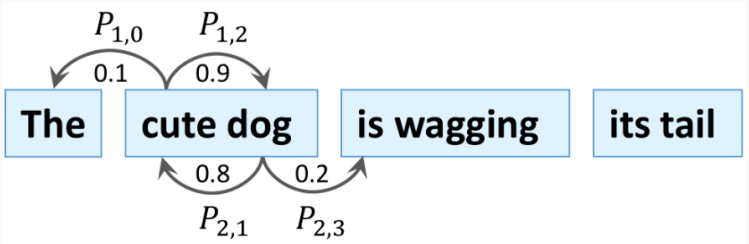
  $a = \{a_1, \ldots, a_i, \ldots, a_N\}$
- a から $C_{i,j} = \prod_{k=i}^{j-1} a_k$ を計算
  - $w_i \sim w_j$ 間の中の $a_{i \leq k < j}$ の値が小さいときに $C_{i,j}$ も小さくなるよう，$C_{i,j}$ は和ではなく積で計算
  - 実際の計算では，勾配消失問題を回避するため $\mathrm{logsumexp}$ により計算



Neighboring Attention

各レイヤで a を計算

1. w_i と w_{i+1} をそれぞれ d_{model} 次元の q_i と k_{i+1} に線形変換
2. スコア $s_{i,i+1} = \frac{q_i k_{i+1}}{d_{model}/2}$ を計算

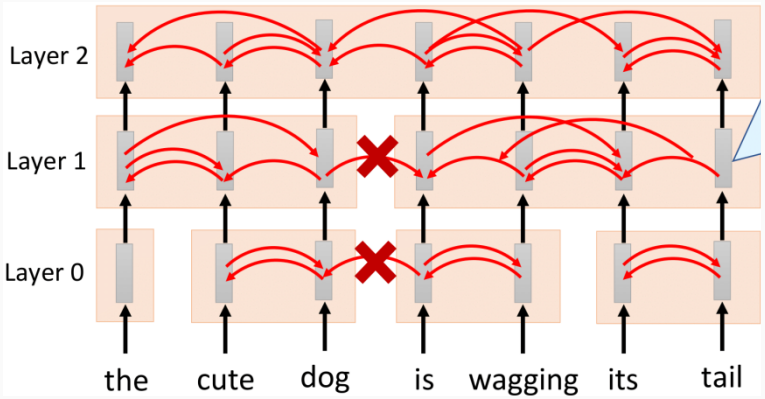


3. $p_{i,i+1}, p_{i,i-1} = \mathrm{softmax}(s_{i,i+1}, s_{i,i-1})$
   - $(p_{i,i+1} + p_{i,i-1}) = 1$ にしないと疎な分布にならないため重要
4. $p_{i,i+1}$ と $p_{i+1,i}$ の幾何平均より $\hat{a}_i$ を計算
   - C を対称行列にするため
5. Hierarchical Constraint (後述) に $\hat{a}_i$ を渡して a_i を計算

Hierarchical Constraint

レイヤ間の階層構造を構築

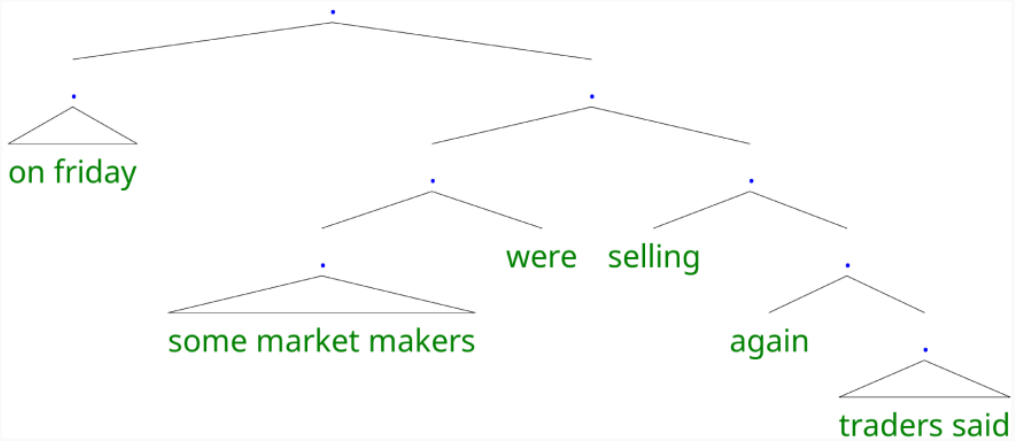
- 上のレイヤほど構成要素の幅を広くする
- レイヤ l，位置 i の隣接単語の結合確率を a_i^l とすると，$a_i^l = a_i^{l-1} + (1 - a_i^{l-1})\hat{a}_i^l$
  - なお，$a_i^{-1} = 0$
  - これにより制約 $a_i^{l-1} < a_i^l$ がかかる
- a^l から C^l を計算



Unsupervised Parsing from Tree Transformer

Algorithm 1 Unsupervised Parsing with Multiple Layers

$a \leftarrow$ link probabilities
$m \leftarrow minimum\ layer\ id$ ▷ Discard the a from layers below minimum layer
$thres \leftarrow 0.8$ ▷ Threshold of breakpoint
procedure BUILDTREE(l, s, e) ▷ l: layer index, s: start index, e: end index
    if $e - s < 2$ **then** ▷ The constituent cannot be split
        return (s, e)
    $span \leftarrow a^l_{s \leq i < e}$
    $b \leftarrow$ **argmin**$(span)$ ▷ Get breakpoint
    $last \leftarrow$ **max**$(l - 1, m)$ ▷ Get index of last layer
    if $a^l_b > thres$ **then**
        if $l = m$ **then**
            return (s, e)
        return BuildTree$(last, s, e)$
    $tree1 \leftarrow$ **BuildTree**$(last, s, b)$
    $tree2 \leftarrow$ **BuildTree**$(last, b + 1, e)$
    return $(tree1, tree2)$ ▷ Return tree



Experiments

- モデルが木構造を捉えられるのか調べるため教師なし句構造解析により文法推論実験

訓練データ	WSJ
訓練法	Masked LM
評価データ	Penn Treebank (WSJ-test / WSJ-10)

Results

F1 スコア (WSJ-test / WSJ-10)

WSJ-test

Model	Data	$F1_{median}$	$F1_{max}$
PRPN	WSJ-train	35.0	42.8
On-lstm	WSJ-train	47.7	49.4
C-PCFG	WSJ-train	**55.2**	**60.1**
Tree-T,L=12	WSJ-train	48.4	50.2
Tree-T,L=10	WSJ-train	49.5	51.1
Tree-T,L=8	WSJ-train	48.3	49.6
Tree-T,L=6	WSJ-train	47.4	48.8
DIORA	NLI	**55.7**	**56.2**
URNNG	Billion	-	52.4
Tree-T,L=10	WSJ-all	50.5	52.0
Random	-	21.6	21.8
LB	-	9.0	9.0
RB	-	39.8	39.8

WSJ-10

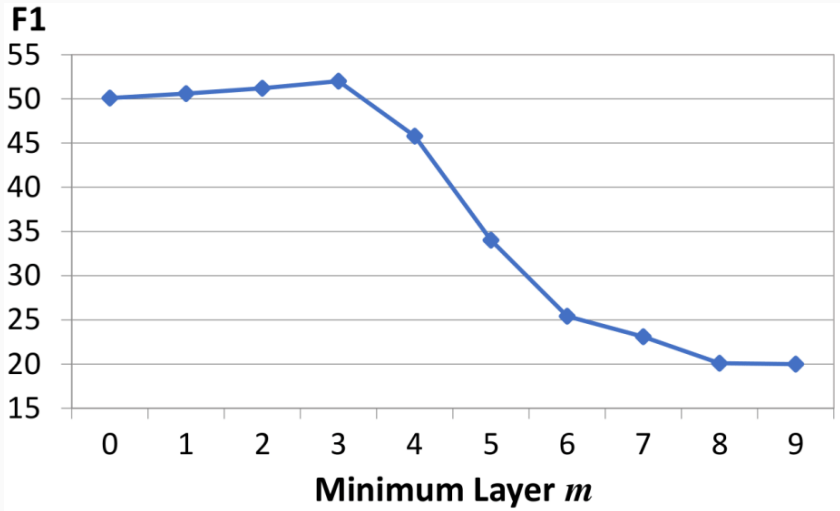
Model	Data	$F1_{median}$	$F1_{max}$
PRPN	WSJ-train	70.5	71.3
On-lstm	WSJ-train	65.1	66.8
C-PCFG	WSJ-train	70.5	-
Tree-T,L=10	WSJ-train	66.2	67.9
DIORA	NLI	67.7	68.5
Tree-T,L=10	WSJ-all	66.2	68.0
CCM	WSJ-10	-	71.9
DMV+CCM	WSJ-10	-	77.6
UML-DOP	WSJ-10	-	**82.9**
Random	-	31.9	32.6
LB	-	19.6	19.6
RB	-	56.6	56.6

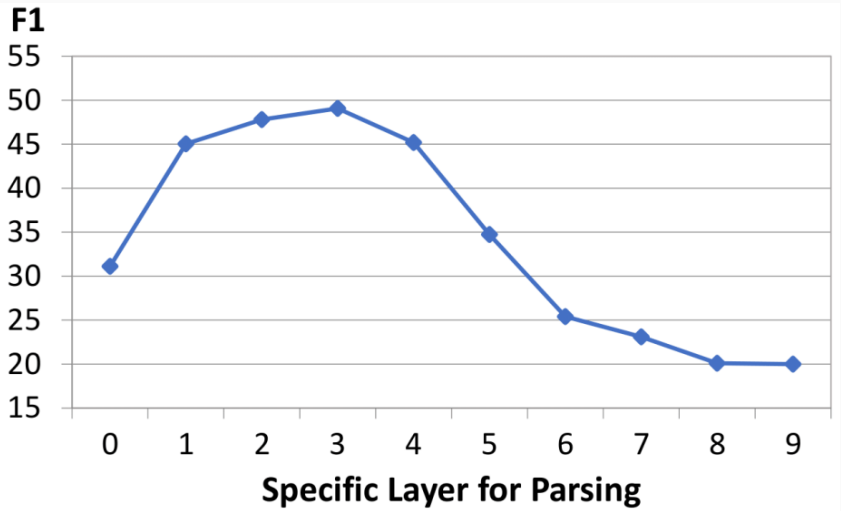
Results

構成要素の Recall (各ラベルで比較)

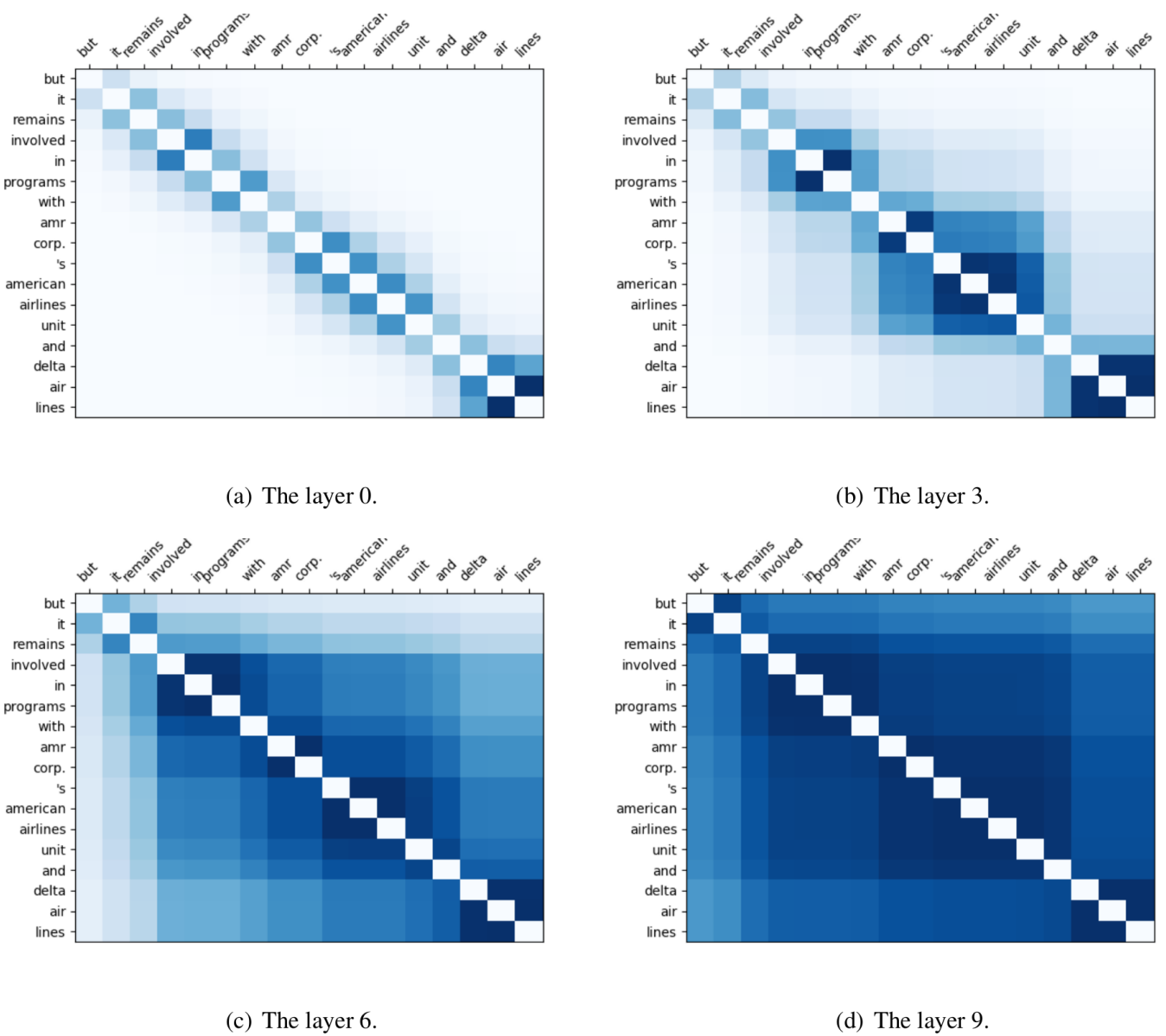
Label	Tree-T	URNNG	PRPN
NP	**67.6**	39.5	63.9
VP	38.5	**76.6**	27.3
PP	52.3	**55.8**	55.1
ADJP	24.7	33.9	**42.5**
SBAR	36.4	**74.8**	28.9
ADVP	**55.1**	50.4	45.1

Analysis


F1
55
50
45
40
35
30
25
20
15
0 1 2 3 4 5 6 7 8 9
Minimum Layer m


F1
55
50
45
40
35
30
25
20
15
0 1 2 3 4 5 6 7 8 9
Specific Layer for Parsing

Interapretable Self-Attention



(a) The layer 0.

(b) The layer 3.

(c) The layer 6.

(d) The layer 9.

Masked Language Modeling

[MASK] トークンの perplexity を評価

Model	L	Params	Perplexity
Transformer	8	40M	48.8
Transformer	10	46M	48.5
Transformer	10-B	67M	49.2
Transformer	12	52M	48.1
Tree-T	8	44M	46.1
Tree-T	10	51M	45.7
Tree-T	12	58M	45.6

- perplexity は通常の Transformer ベースのモデルより Tree Transformer のほうが低い

Limitations and Discussion

- 訓練済み BERT でパラメタ初期化を行うと性能が酷く低下
  - BERT の Attention が Tree Transformer と全く異なる構造を学習していることを示唆
- この１文がよくわからなかった
  - “In addition, with a well-trained Transformer, it is not necessary for the Constituency Attention module to induce reasonable tree structures, because the training loss decreases anyway.”

Conclusion and Future Work

■ Conclusion

- 木構造を Transformer に組み込む初めての試み
- 提案手法の Constituent Attention により木構造を自動的に学習
  - ► 隣接する単語の結合確率から相互に結び付ける
- 教師なし構文解析の性能は一貫した木構造を捉えるという点でモデルの有効性を示した

■ Future Work

- Transformer で木構造を捉える方向性について検討する価値はある
- 解釈可能な Attention はモデルが自然言語を処理する方法を説明し，将来のさらなる改善を導く