木構造自己注意機構による教師あり統語構造解析

成田百花 [1]　持橋大地 [2]　小林一郎 [1]
[1] お茶の水女子大学　[2] 統計数理研究所
{g1820529,koba}@is.ocha.ac.jp　daichi@ism.ac.jp

概要

本研究では，Transformer のエンコーダにおける自己注意機構を入力文の統語構造を反映するように変更し，抽出された特徴量を用いてデコーダにおいて構文木構造を生成する新しい統語構造解析モデルを提案する．提案モデルを用いることで既存のモデルと比較して，WSJ コーパスにおける F_1 スコアをはじめとする評価実験において構文解析精度が向上することが示された．また，提案モデルにおいてエンコーダの中間層の出力を用いた統語構造解析手法についても実験結果を示す．

1　はじめに

近年，自己注意機構を用いた深層学習モデル Transformer [1] が，高い汎用性を持つことから機械翻訳にとどまらず，様々な自然言語処理課題に対して採用され，高い成果を挙げている．自己注意機構とは，文中の各単語間の関連性を表すスコアを算出するもので，潜在的に統語構造情報を捉えている可能性が考えられることから，これを工夫した統語構造解析手法が提案されている [2] [3]．深層学習を統語構造解析に用いることで，文を連続値の情報であるベクトルで扱うことが可能となり，従来の文法規則のマッチングによる処理とは違い，より頑健な処理を実現できるという利点がある．その一方で，統語構造解析研究においては，これまでに構築された多くの統語構造解析結果のデータが存在する．双方の利点を活かすことを考え，本研究では自己注意機構を用いた教師あり統語構造解析モデルを提案する．

2　関連研究

代表的な統語構造解析手法として，CKY アルゴリズムがある．CKY アルゴリズムを模倣した手法として，Stern ら [4] はニューラル手法による Chart Parser を提案している．Stern らのモデルでは，エンコーダの双方向 LSTM によって抽出した入力文の特徴量を用いて，デコーダの Chart Parser が文の “句または節を構成する単語間” と “品詞” それぞれの評価点を算出し，文の最適な構文木を予測する．また，Gaddy ら [5] は Stern らの Chart Parser に修正を加えたモデルを提案している．そして，Kitaev ら [2] は，この Gaddy らのモデルのエンコーダに双方向 LSTM ではなく，自己注意機構を用いたモデルを提案している．Stern ら，Gaddy ら，そして Kitaev らのモデルでは，双方向 LSTM と自己注意機構が統語構造情報を算出することを前提としてデコーダへの入力としている．しかし，双方向 LSTM と自己注意機構の出力には統語構造情報は明示的には反映されていないため，デコーダに用いる情報としてはまだ改善の余地がある．そこで本研究では，エンコーダの出力により確かな統語構造情報を反映するような制約を用いて抽出した特徴量をデコーダの入力とする手法を提案する．

自己注意機構を用いた教師なし統語構造解析モデルとして，Wang ら [3] は Tree Transformer を提案している．Tree Transformer は Transformer のエンコーダを用いたモデルで，自己注意機構に入力文の統語構造情報を反映するように制約を加える．エンコーダは，BERT [6] の学習タスクである，[Mask] トークンに置き換えられた文中の単語を予測する Masked Language Modeling（MLM) により教師なし学習される．しかし，この MLM による学習は，統語構造情報について予測を行いモデルを学習しているのではなく，[Mask] トークンの予測を対象に学習をしているため，統語構造の解析が十分とは言えない．よって，本研究は統語構造情報を含む値を予測値とし，既存の統語構造解析結果のデータを用いた，教師あり統語構造解析モデルを提案する．

3　基本モデル

本節では，提案モデルのベースラインとなる Kitaev ら [2] のアーキテクチャについて説明する．

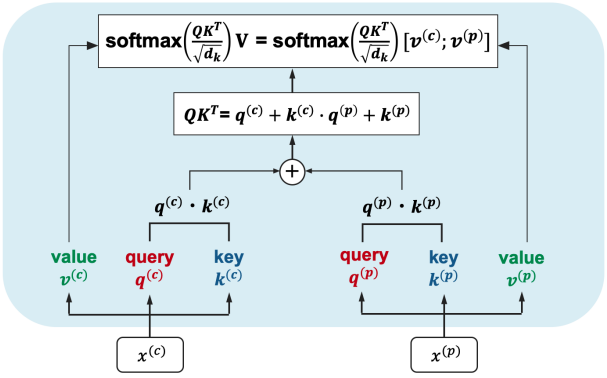

図 1: 分割自己注意機構．入力文の埋め込みベクトル x_c と位置埋め込みベクトル x_p それぞれに対して別々に Attention が計算され，その後に結合される．

エンコーダ部には分割自己注意機構（3.1 節），デコーダ部には Stern ら [4] の Chart Parser に Gaddy ら [5] が修正を加えた Chart Decoder（3.2 節）を用いたモデルである．

N 個の単語を含む入力文の埋め込みベクトル $\boldsymbol{x}^{(c)} = [c_1, c_2, ..., c_N]$ を BERT [6] の最終層の出力から抽出する．また，入力文の位置埋め込みベクトルを $\boldsymbol{x}^{(p)} = [p_1, p_2, ..., p_N]$ とする．

3.1 分割自己注意機構

通常の自己注意機構では埋め込みベクトル $\boldsymbol{x}^{(c)}$ と位置埋め込みベクトル $\boldsymbol{x}^{(p)}$ を足し合わせたベクトル $\boldsymbol{x} = x^{(c)} + x^{(p)}$ に対して Attention を計算する．しかし，分割自己注意機構では $\boldsymbol{x}^{(c)}$ と $\boldsymbol{x}^{(p)}$ それぞれに対して別々に Attention を計算する．分割自己注意機構の入力 $\boldsymbol{x}$，パラメータ行列 W，$c = Wx$ をそれぞれ式 (1)，式 (2)，式 (3) とする．

$$\boldsymbol{x} = [x^{(c)}; x^{(p)}] \tag{1}$$

$$W = \begin{bmatrix} W^{(c)} & 0 \\ 0 & W^{(p)} \end{bmatrix} \tag{2}$$

$$c = [c^{(c)}; c^{(p)}] = [W^{(c)}x^{(c)}; W^{(p)}x^{(p)}] \tag{3}$$

以上より，入力 x から得られる Query，Key，Value はそれぞれ $Q = [q^{(c)}; q^{(p)}]$，$K = [k^{(c)}; k^{(p)}]$，$V = [v^{(c)}; v^{(p)}]$ となる．Query，Key，Value と，Key の次元 d_k 用いて自己注意機構は式 (4) で表される．

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

内積 QK^T は $q^{(c)} \cdot k^{(c)} + q^{(v)} \cdot k^{(v)}$ に分解される．

図 1 に式 (4) の値の導出過程を示す．

3.2 Chart Decoder

構文木 T は文中の i 番目の単語で始まり j 番目で終わる句または節と，その品詞 l の集合によって式 (5) のように表される．$|T|$ は集合の数，すなわちその文に含まれる句または節の合計数である．

$$T = \{(i_t, j_t, l_t) : t = 1, ..., |T|\} \tag{5}$$

また，エンコーダからの出力を用いて，各文の構文木 T に実数値のスコアとして，式 (6) で表される $s(T)$ が生成される．

$$s(T) = \sum_{(i,j,l)\in T} s(i, j, l) \tag{6}$$

$s(i, j, l)$ は品詞 l を持つ文中の i から j 番目の単語間の実数値のスコアである．生成された $s(T)$ のうち最も高いスコアのものが，CKY 法を模倣した式 (7) の推測アルゴリズムによって選ばれ，構文木 T の最適な統語構造として予測される．

$$\hat{T} = \arg\max_{T} s(T) \tag{7}$$

このように Chart Decoder を用いることで，構文木のスコア，すなわち統語構造情報を予測値とすることができる．また Chart Decoder は，CKY アルゴリズムを模倣したシンプルで合理的なアーキテクチャでありながら，WSJ データのみで学習した解析器の中では最も高い性能を達成していることから，本研究でもこれを採用することにする．

4 提案手法

図 2 に提案モデルの全体の構造を示す．エンコーダ部に Wang ら [3] の”Constituent Attention モジュール”を導入した木構造自己注意機構（4.1 節）を用いて，入力文の統語構造情報を抽出する．エンコーダから抽出された情報に基づき，デコーダ部の Chart Decoder（3.2 節）で構文木のスコアを算出し，教師あり学習を行う．

4.1 木構造自己注意機構

木構造自己注意機構は統語構造情報を抽出するために，3.1 節の分割自己注意機構に隣り合う単語間の関連性を捉える”Constituent Attention モジュール”を導入する．”Constituent Attention モジュール”では，N 個の単語を含む文の各単語に対して，文中の句または節（constituent）の区切りを推測する確率 $a = \{a_1, ...a_N\}$ を生成する．確率 a を用いて，同

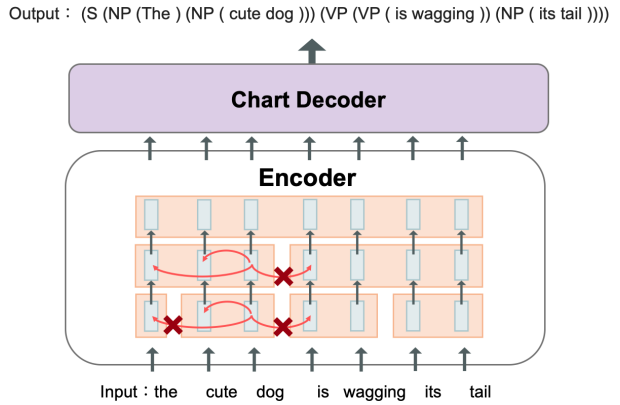

図 2: 提案モデルの全体図．エンコーダには，分割自己注意機構に”Constituent Attention モジュール”を導入した木構造自己注意機構，デコーダには Chart Decoder を用いる．

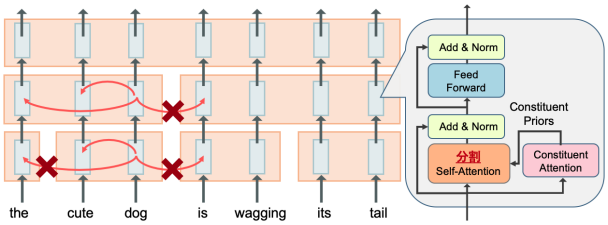

図 3: （左）木構造自己注意機構を用いたエンコーダ．ブロックは入力文から生成された constituent を表している．層を登るにつれ constituent はマージされ徐々に大きくなる．（右）エンコーダ各層の構成．

じ constituent に属する単語同士のみで自己注意機構を作用させる制約 ”Constituent Priors $C_{i,j}$”が生成される．”Constituent Priors $C_{i,j}$”は，式 (8) で表される，文中の i 番目の単語と j 番目の単語が同じ constituent に含まれるかの確率である．

$$C_{i,j} = \prod_{k=i}^{j-1} a_k \tag{8}$$

上層に移るにつれ constituent は隣同士でマージされて徐々に大きくなり，エンコーダ内部で入力文の構文木を形作る（図 3）．式 (8) と式 (4) を用いて，木構造自己注意機構は式 (9) のように求められる．

$$\text{Attention}(Q,K,V)_{tree} = C \odot \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{9}$$

木構造自己注意機構を用いることで，通常の自己注意機構よりも統語構造情報を反映した特徴量を抽出することができる．これにより，Chart Decoder が効率よく文中に含まれる句または節を捉えることを可能とする．

4.2 中間層を用いた統語構造解析

通常の統語構造解析では，エンコーダの最終層の出力に基づいて，デコーダが入力文の構文木を予測する．しかし，入力文によって構文木の深さは異なり，それに合った統語構造情報が必要となると考えられる．そこで，エンコーダ内部に constituent で構文木を形作る木構造自己注意機構の性質を利用して，エンコーダの中間層の出力をデコーダの入力とし，統語構造を解析する手法を提案する．

アルゴリズムを Algorithm 1 に示す．

Algorithm 1 中間層を用いた統語構造解析

```
m ← middle layer id
score_list = []
for layer = ... model.encoder.layers do
    span_score = layer.forward
    score_list.append(span_score)
predicted tree = model.parse(score_list[m])
```

5 実験

本節では，4 章で提案した手法を用いたモデルの性能及び有効性を検証する．

5.1 実験設定

実験では，Penn Treebank [1] の WSJ（英字経済新聞 Wall Street Journal [2]）コーパスを用いて，予測される構文木が人間の作成した構文木とどの程度類似しているかを評価する．コーパスのデータ数はそれぞれ，訓練データ 39,832（2-21 章），開発データ 1,700（22 章），評価データ 2,416（23 章）の標準的分割である．学習の際，1 Epoch おきにデータをシャッフルする．Multi-Head 数を 8，batch 数を 32 とした．

5.2 木構造自己注意機構の有効性検証

本節では，提案手法の木構造自己注意機構の有効性を検証するため，通常の自己注意機構を用いた Kitaev ら [2] のモデル（従来モデル）と，双方向 LSTM を用いた Stern ら [4] と Gaddy ら [5] のモデルと比較を行った．また，同じデータセットを用いたより高い精度を持つモデルとして Zhou ら [7] のモデルも表中に示す．この際，4.2 節の手法は用いず，エンコーダの最終層の出力を用いて統語構造解析を行う．エンコーダの層数を ℓ として，表 1 に実験結

1） https://catalog.ldc.upenn.edu/LDC99T42
2） https://www.wsj.com

表 1: Penn Treebank での実験結果 [%].

	F_1	完全一致	品詞
従来モデル (ℓ=2)	95.75	56.22	**97.45**
従来モデル (ℓ=10)	95.75	55.53	97.24
提案モデル (ℓ=2)	95.58	55.67	97.44
提案モデル (ℓ=8)	95.58	54.71	97.35
提案モデル (ℓ=10)	**95.83**	**56.75**	97.44
提案モデル (ℓ=12)	95.63	54.99	97.33
Stern et al. (2017)	92.56	–	–
Gaddy et al. (2018)	92.08	–	–
Zhou et al. (2020)	96.18	–	–

果を示す．精度の評価尺度には F_1 スコア，再現率と適合率が共に 100%である割合（完全一致），品詞の正解率（品詞）を用いた．表より，提案モデルが従来モデルよりも品詞以外の結果が良いことがわかる．また，双方向 LSTM を用いたモデルと比較しても精度が高い．これらのことから木構造自己注意機構は通常の自己注意機構や双方向 LSTM よりも確かな統語構造情報を抽出し，デコーダでの解析を補助し，性能改善に寄与することがわかる．

また，従来モデル (ℓ=10) が従来モデル (ℓ=2) よりも精度が低い理由は，入力文の埋め込みベクトルの生成に BERT を用いているため，これにより既に入力文の特徴がある程度抽出され，その後に多くのエンコーダを積んでしまうと過学習を引き起こす可能性が考えられる．一方で，提案モデル (ℓ=10) が提案モデル (ℓ=2) よりも精度が高い理由は，エンコーダ内部で構文木を形作る木構造自己注意機構の性質により，入力文の構文木を作るためにはある程度の層数を必要とすることが考えられる．

5.3 文の長さにおける有効性検証

本節では，文の長さにおける各モデルの有効性を検証するため，従来モデル (ℓ=2) と提案モデルの，5.2 節の解析結果から完全一致した文の長さごとの数（正解数）を比較する．結果を表 2 に示す．表中の WSJ は評価データに含まれる文の各長さの内訳である．長さが 40 以下の文に対しては，5.2 節において最も精度の高かった提案モデル (ℓ=10) の正解数が多かった．しかし，長さが 41 以上の文については提案モデル (ℓ=8) の正解数が多かった．これにより文の長さによって構文木の深さが異なり，必要となる統語構造情報が異なることが考えられる．

表 2: 各モデルの解析結果から完全一致した文の長さ $|s|$ ごとの数（WSJ は評価データに含まれる文の各長さの内訳）

	≤ 20	$20 < \|s\| \leq 40$	> 40
従来モデル (ℓ=2)	740	572	40
提案モデル (ℓ=8)	731	536	**51**
提案モデル (ℓ=10)	**743**	**579**	45
提案モデル (ℓ=12)	735	546	41
WSJ	1034	1211	171

表 3: 中間層を用いた統語構造解析結果（提案モデル (ℓ=12) のエンコーダの各中間層の出力を用いた解析結果から，完全一致した文の長さ $|s|$ ごとの数）

中間層	≤ 20	$20 < \|s\| \leq 40$	> 40
Layer 6	677	430	25
Layer 8	714	506	36
Layer 10	730	530	36
Layer 12 (最終層)	735	546	41

5.4 中間層を用いた統語構造解析結果

本節では，提案手法の中間層を用いた統語構造解析の結果を示す．提案モデル (ℓ=12) のエンコーダの各中間層の出力をデコーダの入力とし，構文木を予測する．各中間層を用いた解析結果から完全一致した文の長さごとの数を表 3 に示す．表中の中間層は Algorithm 1 の m である．表より，Layer10 と Layer8 の出力を用いた際の，長さ 41 以上の正解数が同じことから，完全一致した長さ 41 以上の文では Layer8 と Layer10 で出力されるスコアが重複していることが考えられる．以上と 5.3 節の結果から，構文木それぞれに対して適切な中間層から出力される統語構造情報を用いるべきであることが考えられる．

6 まとめ

本論文では，木構造自己注意機構とエンコーダの中間層を用いた統語解析手法を提案した．木構造自己注意機構を用いることで，F_1 スコアと完全一致において従来の自己注意機構や双方向 LSTM を用いたモデルを上回る結果を達成できることを実験により確認した．また，エンコーダの中間層を用いた解析結果から，各構文木に対して適切な中間層の出力を用いるべき事がわかった。このことから今後は，ロジスティック回帰などを用いて効率的に各入力文の構文木に適した中間層を選びとるような手法を提案し，モデルの改良に取り組みたい．

参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of the 31st NIPS*, 2017.

[2] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[3] Yau-Shian Wang, Hung yi Lee, and Yun-Nung (Vivian) Chen. Tree transformer: Integrating tree structures into self-attention. In *EMNLP*, 2019.

[4] Mitchell Stern, Jacob Andreas, and Dan Klein. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 818–827, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[5] David Gaddy, Mitchell Stern, and Dan Klein. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 999–1010, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Junru Zhou, Zuchao Li, and Hai Zhao. Parsing all: Syntax and semantics, dependencies and spans. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4438–4449, Online, November 2020. Association for Computational Linguistics.