Vol.2018-NL-237 No.9
2018/9/26

文脈自由文法とニューラルネットワークを用いた並列構造木のCKY構文解析

寺西 裕紀[1,a)]　進藤 裕之[1,b)]　松本 裕治[1,c)]

概要：本研究は文脈自由文法を用いて並列構造を木として導出する手法を提案する．現在の最高精度の解析性能を達成しているニューラルネットワークによる並列構造解析の手法は，文中の複数の並列構造や三つ以上の並列句を陽に扱っておらず，解析結果を他のタスク等に利用することが難しい．複数の並列構造や三つ以上の並列句の解析を行う場合，並列構造となりうる句のスパンの組み合わせが指数的に増加するため膨大な計算コストを要するという問題がある．そこで本研究では並列構造を木として導出できる文脈自由文法を定義し，CKY アルゴリズムの適用により計算量を抑えながら並列構造の木をボトムアップで構築する．実験の結果，提案手法によって文中の複雑な並列構造の範囲が競合せず導出できることを保証しながら，個々の並列構造ごとの評価において既存手法と同等以上の解析性能を得た．

1. はじめに

並列構造は等位接続詞などの句を連接させる働きのある語にともなって，句や文が並列して出現する構造である．並列構造は自然言語において高頻度で出現するが，並列関係にある句（並列句）の範囲には曖昧性があり，さらに並列構造によって文が長くなるために，高精度の構文解析器であってもしばしば誤りが生じる．並列構造解析とは並列句の範囲を同定するタスクであり，並列句の範囲同定の精度が高まることで構文解析や機械翻訳への応用が期待される．並列構造は並列する句の数に限りがなく，文中に出現する複数の並列構造の範囲が互いに独立していないため，文における並列構造として解釈可能な範囲の組み合わせの数は膨大となり，そのことが並列構造解析のタスクを困難にしている．

ニューラルネットワークによる自然言語解析の手法が発展し，並列構造の範囲を同定するモデルにおいてもニューラルネットワークが利用されるようになってきた [2], [10]．ニューラルネットワークによる並列構造解析は従来の手法と比較して解析精度は向上しているものの，計算コストを要するために並列構造となりうる句の組み合わせを限定して解析を行っている．より具体的には，並列構造に含まれる並列句の数を二つに限定し，複数の並列構造の範囲を独立して決定することによって計算量を抑えている．Ficler ら [2] や寺西ら [10] のニューラルネットワークによる手法は，並列構造解析の限られた問題設定においては精度向上が示されたが，三つ以上の並列句を含む並列構造や入れ子となる並列構造など，より難解かつ頻出する構造に対する解析において実用上の課題が残っている．

本研究では三つ以上の並列句を含む並列構造や複数の並列構造を解析できる方法を導入し，ニューラルネットワークによる並列構造解析を行う．具体的には，原ら [4] が提案した文脈自由文法を用いて並列構造を構文木として導出する手法を拡張し，CKY アルゴリズムによる確率文脈自由文法の構文解析とニューラルネットワークを組み合わせて並列構造の解析を行う．原らの手法は並列句が三つ以上ある場合や並列構造が入れ子になっている場合であっても動的計画法により並列構造を木として効率良く導出できる反面で，並列構造の木に与えるスコアは並列句の類似性に基づいて系列アラインメントとパーセプトロンを使って計算しており，動詞句や節の並列などの非類似となる傾向のある並列句の同定に課題があった．提案手法では並列句の類似性だけではなく可換性に基づいてニューラルネットワークによって句のペアのスコア計算を行う．

本稿では，はじめに並列構造解析のタスクについて定義し，三つ以上の並列句を含む並列構造や複数の並列構造を解析する困難さについて述べる（2 節）．次に並列構造を木として導出するための文法規則とアルゴリズム，ニューラルネットワークによるスコア付与について説明をする（3 節）．実験の結果，提案手法による解析は既存手法と同等

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] teranishi.hiroki.sw5@is.naist.jp
[b)] shindo@is.naist.jp
[c)] matsu@is.naist.jp

以上の性能を発揮したものの（4 節），分析により提案手法の課題が示された（5 節）．

2. 並列構造解析

2.1 用語・記法の定義

本研究では英語における並列構造解析について取り扱う．本稿では，句を連接する働きを持つ可能性のある語を並列キーと呼び，“$word_{\text{position}}$” の形式で表す．並列キーに属する語として主に等位接続詞が挙げられるが，英語の but が等位接続詞のほかに前置詞となる場合があるように，等位接続詞以外の品詞となり得る語は語の表層形から並列キーが句を連接する働きがあるかどうかを判別することが必ずしもできない．並列キーが句を連接している場合，結びつけられる句を**並列句**と呼び，並列キーと並列句から成る一連の語句を**並列構造**と呼ぶ．反対に並列キーが句を連接する働きがない場合は並列キーによって結びつけられる句や対応する並列構造は存在しない．並列キーに対して並列構造が存在する場合は並列キーを *true* 並列キーと呼び，並列構造が存在しない場合は並列キーを *false* 並列キーと呼ぶ．また，並列キーとその前後の並列句から成る語句がその直前にカンマを伴ってさらに並列する句を持つ場合も並列構造として扱う．このように並列キーに前出して句を結びつける役割を果たす語を補助並列キーと呼び，実際に並列句を伴う場合は *true* 補助並列キー，伴わない場合は *false* 補助並列キーと呼ぶ．本研究では，並列キーの対象となる語を “and”，“or”，“but”，“nor”，“and/or” とし，補助並列キーとなる語は “,” とする．

2.2 タスクの定義

本研究で取り組むタスクは並列構造に含まれる並列句の範囲を同定するタスクである．並列句の範囲は並列キーごとに出力する．並列キーが *true* 並列キーの場合はその並列構造に含まれる全ての並列句の始点・終点を列挙する．このとき，並列構造に含まれる *true* 補助並列キーによって結びつけられた並列句も列挙の対象とする．並列キーが *false* 並列キーの場合は並列構造は存在せず，列挙される並列句も存在しないため，並列キーによって紐付けられた並列句として None を返す．**図 1** はタスクの入出力の例である．

2.3 並列句の範囲の組み合わせ

既存研究の多くが並列構造の形式を限定して解析を行っている [2], [6], [9], [10]．文長 n の k 番目の語として出現する並列キーに対して，並列キーに隣接する前後のスパンのみを対象として並列句ペアの候補を列挙した場合は，候補はたかだか $(k-1)\times(n-k)$ 通りしかないが，並列句の数を限定せず，さらに複数の並列構造を同時に解析した場合は組み合わせが爆発的に増加する．たとえば，“Fred sent Uncle Willy a watch, Aunt Samantha a pearl necklace and_{13} earrings, and_{16} their grandson Kevin old toys and_{22} chocolates.” という文（文長 $n=24$）の場合について考える．ただし，and_i に前出する並列句の範囲は and_i の直前 $i-1$ の語（$i-1$ の語がカンマの場合は $i-2$ の語）を終点とし，and_i に後出する並列句の範囲は and_i の直後 $i+1$ の語で開始するものとする．また，and_i に前出する並列句はカンマを補助並列キーとして二つ以上出現する場合を考慮するが，and_i に後続する並列句は一つのみとする．この場合，and_{13} の並列句となりうる組み合わせは 198 通り，and_{16} は 160 通り，and_{22} は 94 通りとなり，文の並列構造の可能な組み合わせは $198\times160\times94=2,977,920$ 通りのうち，異なる並列構造が一部分のみ重なり合うような組み合わせを除いた（入れ子となる並列構造は許容する）数となる．ニューラルネットワークによる句のペアに対するスコア付けをこれら全ての組み合わせについて行うことは現実的ではない．そこで本研究では並列構造を木として導出する文脈自由文法の規則に CKY アルゴリズムを適用し，動的計画法により並列構造のスコア計算を行う手法を提案する．

入力 “But_1 it said Charles Johnston, ISI chairman and_9 president, agreed to sell his 60% stake in ISI to Memotec upon completion of the tender offer for a combination of cash, Memotec stock and_{37} debentures.”
出力 but_1: NONE
and_9: (8, 8) chairman ; (10, 10) president
and_{37}: (33, 33) cash ; (35, 36) Memotec stock ; (38, 38) debentures

図 1 タスクの入出力の例

3. 提案手法

並列構造の階層関係を木構造で表したものを本稿では並列構造木と呼ぶ（**図 2**）．並列構造のノードは coord と表し，並列句（conj），並列キー（cc），その他のノードから成る．並列句のノード conj は，さらに他の並列構造を子として持つことができる．並列構造木による表現を用いることで，ある並列構造が他の並列構造に含まれているか，並列構造がいくつの並列句から成り立っているかに依存せず，一貫して記述ができる．

本研究では並列構造木を導出するための文脈自由文法の規則を定義する．文脈自由文法を用いる理由は，並列構造を導出しないスパンの組み合わせや複数の並列構造の範囲が競合するような組み合わせを並列句の候補から排除することができ，並列構造木を導出可能な句の組み合わせのみを探索することができるためである．また，文脈自由文法を用いた構文木の導出は，CKY アルゴリズムやチャート法などを用いて効率的に行うことができる．本節では，並列構造木に含まれるノードにスコアを付与し，CKY アルゴリズムを用いてスコアの合計が最も高い並列構造木を得

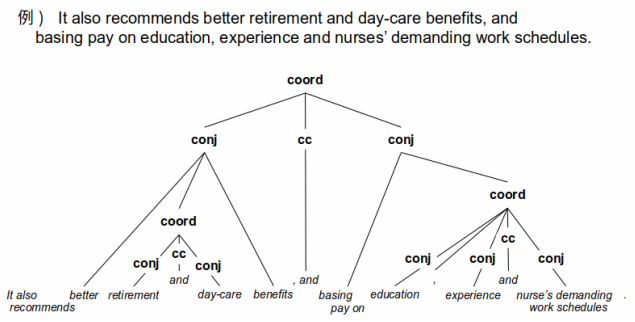

図 2 並列構造木の例

る方法について示す．

3.1 文脈自由文法規則の定義

本研究で用いる文脈自由文法の規則を**表 1** に示す．文脈自由文法の規則は原ら [4] が提案した文法規則を拡張したものである．表 1 の文法規則では，文と並列構造が与えられたときにその並列構造木を導出する構文木が一意に決まる．規則 (1)(2) は並列構造を導出する規則であり，(3) は導出済みの並列構造に対して前出の並列句を加える規則である．また，入れ子となる並列構造は規則 (4)(5)(8)(9) から，文全体の並列構造木は規則 (6)(7) から導出される．なお，並列キーが *false* 並列キーとなる（並列構造が存在しない）場合を考慮して，(14) の右辺の*は (12)(13) の右辺の語を含むあらゆる語を表すものとする．

表 1 並列構造木導出の文法規則

非終端記号			
COORD	Coordination; 並列構造		
CJT	Conjunct; 並列句		
CC	Coordinating Conjunction; 並列キー		
CC-SEP	CC-separator; 補助並列キー		
W	Word; 語		
N	Non-coordination; 非並列構造		
S	Sentence; 文		
並列構造に関する規則			
(1)	COORD	→	CJT CC CJT
(2)	COORD	→	CJT CC-SEP CC CJT
(3)	COORD	→	CJT CC-SEP COORD
(4)	CJT	→	COORD
(5)	CJT	→	N
並列構造以外の規則			
(6)	S	→	COORD
(7)	S	→	N
(8)	N	→	COORD N
(9)	N	→	W COORD
(10)	N	→	W N
(11)	N	→	W
前終端記号の規則			
(12)	CC	→	(and|or|but|nor|and/or)
(13)	CC-SEP	→	,
(14)	W	→	*

3.2 並列構造木へのスコア付与

並列構造木のスコアは木のノードに付与されたスコアの合計とする．スコア付与の対象となるノードは並列構造のノード（coord）のみとする．並列構造のノードのスコアは，並列構造に含まれる n 個の並列句の隣接する $n-1$ ペアごとにスコアを計算し，その合計を並列構造のスコアとする．並列構造木のスコア計算のルールを表 1 の文法規則で導出した構文木に適用すると，並列構造のスコアは構文木に含まれる COORD ノードに対して付与される．**図 3** は表 1 の文法規則から導出した構文木へのスコア付与の例である．並列構造木のスコアは構文木に含まれる COORD ノードのスコアの合計となる[*1]．したがって，提案手法によって入力文 x に対して出力される並列構造木を表す構文木 $\hat{y}$ は次式より求められる．

$$s(t) = \sum_{v \in t; v=COORD} \textsc{Score}(v)$$
$$\hat{y} = \arg\max_{t \in \mathcal{T}_G(x)} s(t) \quad (1)$$

ただし，G は文法規則の集合，$\mathcal{T}_G$ は文法 G から導出可能な入力文 x の構文木の集合，v は構文木 t に含まれるノードを表す．$\mathcal{T}_G$ からスコア最大となる構文木 $\hat{y}$ を効率的に導出する方法については 3.3 節，並列構造のノード v にスコアを付与する関数 Score については 3.4 節で説明をする．

3.3 CKY アルゴリズムによる確率文脈自由文法の構文解析

2.3 節で示したとおり，文の全ての並列構造から成る組み合わせは並列キー・補助並列キーの存在によって指数的に増加するため，表 1 の規則 G から導出可能な並列構造木 $t \in \mathcal{T}_G$ を全て列挙してスコア付けをすることは計算資源の点で現実的ではないうえに効率が悪い．そこで，表 1 の文法規則をチョムスキー標準形に変換することによって CKY アルゴリズムを適用する．構文木のスコアは木が持つ COORD ノードのスコアの合計であり，各 COORD ノードまでの部分木は部分木のスコアが最も高くなるよう再帰的に計算されて決定される．そのため，CKY アルゴリズムによる確率文脈自由文法の構文解析の手法を適用することができ，最もスコアの高い並列構造の構文木 $\hat{y}$ を求める式 1 は動的計画法により時間計算量 $\mathcal{O}(n)$（n は文長）で計算がされる．なお，本研究では並列構造に属する並列句を最右から導出するため，三つ以上の非終記号から成る規則は右分解によってバイナリの規則に分解する．

[*1] 表 1 の規則から導出できない並列構造木であっても，並列構造に含まれる並列句の隣接ペアごとのスコア計算は可能なため，並列構造木のスコアを計算することができる．

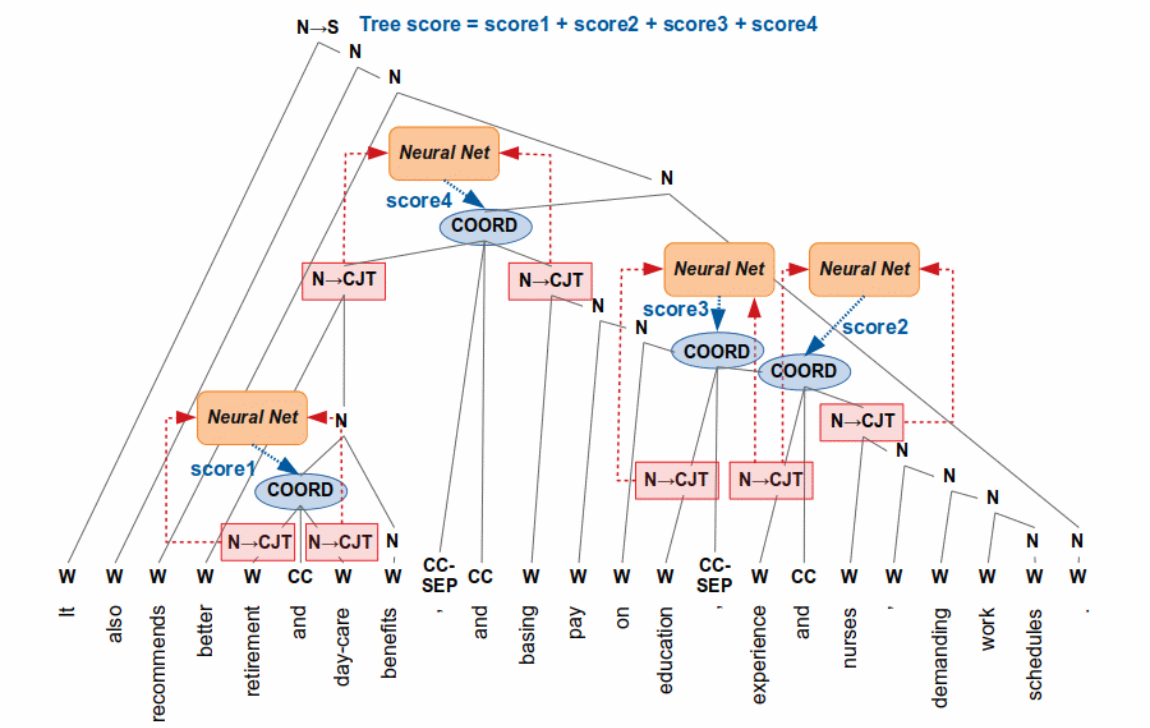

図 3　並列構造を表す構文木へのスコア付与の例

3.4　ニューラルネットワークによるスコア計算

本研究では，ニューラルネットワークを用いて句のペアに対してスコアを計算する．スコア付与に用いるモデルは二つの句が並列関係になりやすいほど高いスコアを割り当てるように設計・学習される．本研究で使用するモデルは寺西ら [10] が提案したモデルと同一である．寺西らのモデルは並列句の類似性・可換性に基づいて特徴ベクトルの計算を行う．

単語・品詞ベクトルの系列を双方向型 LSTM に入力し，得られた隠れ状態ベクトルの系列を $\{\mathbf{h}_t\}_{t=1}^{N}$（$N$ は単語数）とする．類似性の特徴ベクトルは，スパン (i,j) と (l,m)（ただし，$j<l$）に対して次のとおりに計算される．

$$\begin{aligned}\mathbf{v}_{i,j} &= \mathrm{average}(\mathbf{h}_{i:j})\\ \mathbf{v}_{l,m} &= \mathrm{average}(\mathbf{h}_{l:m})\\ f_{sim}(\mathbf{v}_{i,j},\mathbf{v}_{l,m}) &= \left[|\mathbf{v}_{i,j}-\mathbf{v}_{l,m}|;\mathbf{v}_{i,j}\odot\mathbf{v}_{l,m}\right]\end{aligned} \tag{2}$$

すなわち，隠れ状態ベクトルの系列のうちスパン範囲のベクトルの平均をスパンベクトルとし，スパンベクトル同士の要素間の差の絶対値と要素積を類似性の特徴ベクトルとしている．

一方で可換性の特徴ベクトルは次の式で計算される．

$$\begin{aligned}f_{repl}(\mathbf{h}_{1:N},i,j,l,m) = \\ \big[|\mathbf{h}_{i-1}\odot\mathbf{h}_i-\mathbf{h}_{i-1}\odot\mathbf{h}_l|;\\ |\mathbf{h}_m\odot\mathbf{h}_{m+1}-\mathbf{h}_j\odot\mathbf{h}_{m+1}|\big]\end{aligned} \tag{3}$$

つまり，前方の句の直前の語とそれぞれの句の先頭の語とのベクトルの要素積をとり，二つの要素積の差の絶対値を求め，後方の句の直後の語においても同様の計算を行って可換性の特徴ベクトルとしている．

句のスパンのペア (i,j) と (l,m) のスコアは式 (2) および (3) で得られた特徴ベクトルから多層パーセプトロンを用いた演算によって求める．

$$\begin{aligned}\mathrm{PairScore}(\mathbf{h}_{1:N},i,j,l,m) = \\ \mathrm{MLP}\big(\big[f_{sim}(\mathbf{v}_{i,j},\mathbf{v}_{l,m});\\ f_{repl}(\mathbf{h}_{1:N},i,j,l,m)\big]\big)\end{aligned} \tag{4}$$

よって関数 Score は入力文 x の単語系列 $w_{1:N}$・品詞系列 $p_{1:N}$ と COORD ノード v が持つ並列句のスパンのペア $(i,j),(l,m)$ を用いて以下のように定義する．

$$\begin{aligned}\mathrm{Score}(v) &= \mathrm{Score}'(w_{1:N},p_{1:N},(i,j),(l,m))\\ &= \mathrm{PairScore}(\mathrm{BiLSTM}(w_{1:N},p_{1:N}),i,j,l,m)\end{aligned} \tag{5}$$

ただし，BiLSTM は単語・品詞の系列を単語・品詞の連結ベクトルの系列に変換をし，双方向 LSTM により隠れ状態ベクトルの系列を返す計算ユニットとする．

本研究では構文木は CKY アルゴリズムによってボトムアップに解析されるため，アルゴリズムの適用に先立って計算が必要な COORD ノードを数え上げることはできない．しかしながら，COORD ノードの出現の都度ニューラルネットワークによるスコアを行った場合，計算が並列化できないことから効率が悪く，解析に時間を要する．そこで複数の COORD ノードの計算を並列化する実装上の工

夫について付録 A.1 で説明する．

3.5 学習

3.5.1 1-best 学習

提案手法による並列構造木の予測 $\hat{y}$ と正解の並列構造木 y を用いて損失関数 $\ell_\theta(\hat{y}, y)$ を次のとおり定義する．

$$\ell_\theta(\hat{y}, y) = \begin{cases} 0 & (\hat{y} = y) \\ \max(0, \alpha - s(y) + s(\hat{y})) & (otherwise) \end{cases} \tag{6}$$

ただし θ はモデルのパラメータの集合であり，α は $s(y)$ と $s(\hat{y})$ のスコア差（マージン）が α 以上となるよう学習に用いられるハイパーパラメータである．

ニューラルネットワークの学習は式 6 を用いて，次の関数を最小化することにより行うことができる．

$$J(\theta) = \sum_{(x^{(d)}, y^{(d)}) \in D} \ell_\theta(\underset{t \in \mathcal{T}_G(x^{(d)})}{\arg\max}\ s(t), y^{(d)}) + \frac{\lambda}{2}\|\theta\|^2 \tag{7}$$

ここで D はトレーニングのデータセットにおける文と並列構造木の対の集合であり，λ は L2 正則化の強度を調整するハイパーパラメータである．

3.5.2 N-best 学習

式 6 を用いてニューラルネットワークを学習した場合，パラメータ θ の 1 回の更新において学習データ D の 1 事例 $(x^{(d)}, y^{(d)})$ につき一つの予測並列構造木 $\hat{y}$ についてしかパラメータ θ の勾配が計算がされず，学習が収束するまでに多くの更新回数を必要とする[*2]．学習の収束を向上させるために，1 事例 $(x^{(d)}, y^{(d)})$ につき予測並列構造木を N 個導出し，それぞれの木について正解の並列構造木 y との損失を計算する．

$$J'(\theta) = \sum_{(x^{(d)}, y^{(d)}) \in D} \sum_{\substack{\hat{y} \in \arg\mathrm{N}\max \\ t \in \mathcal{T}_G(x^{(d)})}} \ell_\theta(\hat{y}, y^{(d)}) + \frac{\lambda}{2}\|\theta\|^2 \tag{8}$$

本研究では式 8 を確率的勾配降下法によって最小化することによってモデルのパラメータ θ を最適化する．

3.5.3 事前学習

本研究では，寺西ら [10] が用いた学習手法をニューラルネットワークの事前学習に用いる．学習データに含まれる全ての並列キーに対して可能な並列句ペアの組み合わせ全てについてスコア計算し，クロスエントロピー損失を最小化することによってパラメータ θ を学習させる．

$$\begin{aligned} \mathcal{S}^{pre}_{x_{1:N},k} &= \{(1, k-1), (2, k-1), \ldots, (k-1, k-1)\} \\ \mathcal{S}^{post}_{x_{1:N},k} &= \{(k+1, k+1), (k+1, k+2), \ldots, (k+1, N)\} \\ \mathcal{S}_{x_{1:N},k} &= \mathcal{S}^{pre}_{x_{1:N},k} \times \mathcal{S}^{post}_{x_{1:N},k} \cup \{\textsc{None}\} \end{aligned} \tag{9}$$

[*2] Adam のような学習率を減衰させる最適化手法を用いて学習を行った場合に学習が早期に停滞した．

表 2 Penn Treebank コーパスにおける並列キーの出現数

	# 並列キーの出現数	# 並列キーの出現文数
コーパス全体	27,903 (24,450)	21,314 (19,095)
訓練データ	22,670 (17,893)	17,282 (15,481)
開発データ	953 (848)	742 (673)
評価データ	1,282 (1,099)	985 (873)

括弧内は実際に並列構造が存在する場合のみの集計．

$$\textsc{Score}''(x, k, s) = \begin{cases} \mathbf{w}_{none} \cdot \mathrm{BiLSTM}(\mathrm{w}_{1:\mathrm{N}}, \mathrm{p}_{1:\mathrm{N}})_{\mathrm{k}} + \mathbf{b}_{\mathrm{none}} & (s = \textsc{None}) \\ \textsc{Score}'(w_{1:N}, p_{1:N}, (i, j), (l, m)) & (otherwise) \end{cases} \tag{10}$$

$$\hat{p}_\theta(x, s) = \frac{\exp(\textsc{Score}''(x, k, s))}{\sum_{s' \in \mathcal{S}} \exp(\textsc{Score}''(x, k, s'))} \tag{11}$$

$$\begin{aligned} &J'_{pretrain}(\theta) = \\ &- \sum_{(x^{(d)}, y^{(d)}) \in D} \sum_{(k,s) \in y^{(d)}} \log \hat{p}_\theta(x^{(d)}, (i, j), (l, m)) \\ &+ \frac{\lambda_{pretrain}}{2}\|\theta\|^2 \end{aligned} \tag{12}$$

ただし，$(k, s) \in y^{(d)}$ の k は入力文 $x^{(d)}$ における並列キーの位置を表し，s は並列キー k に隣接する並列句ペア（*false* 並列キーの場合は None）とする．式 12 を用いて並列句ペアにスコアを付与するニューラルネットワークのパラメータ θ を事前に学習させ，提案手法の CKY 構文解析に用いるニューラルネットワークのパラメータ θ の初期値とする．

4. 実験

提案手法の性能評価の実験を並列構造のアノテーションが付与された Penn Treebank コーパス [1] を用いて行った．評価対象の並列キーは “and”，“or”，“but”，“nor”，“and/or” とした．並列キーの出現数と文数を**表 2** にまとめる．

4.1 実験設定

Penn Treebank コーパスの Wall Street Journal のパートのうち，セクション 2 から 21 を訓練データ，セクション 22 を開発データ，セクション 23 を評価データとして用いた．ニューラルネットワークのアーキテクチャに関するハイパーパラメータは寺西ら [10] のものと同様とした．単語の分散表現は English Gigaword コーパス第 5 版 [7] の New York Times パートを Word2Vec[*3]のデフォルトのパラメータで事前に学習した 200 次元のベクトル表現を用いた．品詞は Stanford Parser [11] を用いて訓練データに対

[*3] https://code.google.com/archive/p/word2vec/

表 3　並列構造単位での各評価基準による評価結果

（単位：%）

		Development						Test					
		全並列構造			名詞句並列			全並列構造			名詞句並列		
		P	R	F	P	R	F	P	R	F	P	R	F
Ours	whole	77.39	70.63	73.85	82.32	76.88	79.51	76.69	67.97	72.06	81.31	74.87	77.96
	outer	76.48	69.81	72.99	81.56	76.17	78.78	74.74	66.24	70.23	80.21	73.86	76.90
	inner	**78.42**	71.58	**74.84**	**82.82**	**77.35**	**80.00**	**76.89**	68.21	72.29	**82.05**	75.54	**78.66**
	exact	76.39	69.70	72.89	81.20	75.87	78.45	74.74	66.24	70.23	80.21	73.86	76.90
Teranishi+ 17 :re-impl+ext	whole	75.53	74.29	74.91	78.84	77.35	78.09	74.23	75.04	74.63	80.23	80.10	80.16
	outer	71.82	70.63	71.22	76.68	75.23	75.95	68.94	69.69	69.32	75.84	75.71	75.78
	inner	73.02	71.81	72.41	77.88	76.41	77.14	71.10	71.88	71.49	77.87	**77.74**	77.80
	exact	71.46	70.28	70.86	76.44	75.00	75.71	68.13	68.88	68.50	75.67	75.54	75.61
Teranishi+ 17	whole	75.92	72.87	74.36	77.90	75.05	76.45	-	-	-	-	-	-
	outer	72.48	69.57	70.99	76.24	73.45	74.82	-	-	-	-	-	-
	inner	74.07	71.10	72.56	77.43	74.59	75.99	73.46	72.16	**72.81**	75.87	74.76	75.31
	exact	72.11	69.22	70.63	75.77	72.99	74.35	-	-	-	-	-	-
Ficler+ 16	inner	72.34	**72.25**	72.29	75.17	74.82	74.99	72.81	**72.61**	72.7	76.91	75.31	76.1

Teranishi+ 17 および Ficler+ 16 らの手法の結果はそれぞれ寺西ら [10]，Ficler ら [2] が報告している結果より抜粋．Teranishi+ 17:re-impl+ext は 4.2 節で説明したベースラインの手法である．

する 10 分割ジャックナイフ法にて付与し，品詞の分散表現として区間 $[-1, 1]$ の一様分布でランダムに初期化をした 50 次元のベクトルを用いた．モデルのパラメータはバッチサイズ 20 のミニバッチ確率的勾配降下法により最適化をし，学習率の調整は Adam [5] をデフォルトの設定値で用いて行った．Dropout や L2 正則化を用いた場合に未学習（underfitting）が生じるため，ニューラルネットワークの学習および事前学習のいずれにおいても適用しなかった．N-best 学習における予測の並行構造木の導出数 N は {4, 8, 16} から，損失関数のマージン α は {0.5, 1.0, 2.0, 3.0, 4.0, 5.0} から選択した．学習のエポック数は 30 とし，開発データにおける exact の評価指標（後述）における F1 のスコアが最も高いモデルとハイパーパラメータの設定を評価データでの評価に用いるモデルとして採用した．最終的なハイパーパラメータを付録 A.2 に示す．

4.2　評価指標

並列構造に属する並列句の始点・終点の一致について，寺西ら [10] と同様に四種類の一致基準で評価を行う．

- **whole**：最初の並列句の始点と最後の並列句の終点の一致
- **inner**：並列キーに隣接する並列句の始点・終点の一致
- **outer**：最初と最後の並列句の始点・終点の一致
- **exact**：全ての並列句の始点・終点の一致

それぞれの一致基準について，適合率 (P)・再現率 (R) およびそれらの調和平均である F 値によって手法の評価を行う．また，提案手法の有効性を検証するため，文中の複数の並列構造の完全一致についても評価を行う．提案手法と比較するベースラインのモデルとして寺西ら [10] のモデルを用いる．寺西らのモデルの評価においては，文の個々の並列構造のスパンのスコアを独立に計算し，複数の並列構造のスパンが競合しない組み合わせのなかで最もスコアの高い並列構造の組み合わせをモデルの予測として用いる．

4.3　実験結果・考察

4.3.1　並列構造単位での評価

並列構造単位での実験の評価結果を**表 3** に示す．並列構造単位での評価において，提案手法は寺西ら [10] のモデルを拡張したベースラインのモデルをいずれの基準においても上回った．inner 基準による評価では，提案手法は寺西ら [10]，Ficler ら [2] の手法と同程度の性能を達成した．また，寺西らの手法においては補助並列キーを全て *true* 補助並列キーとみなして三つ以上の並列句へと分割して評価していたのに対し，提案手法は三つ以上の並列句から成る並列構造を導出できる文法規則を用いることで，outer 基準・exact 基準において寺西らの手法より高い F 値を達成している．

4.3.2　文単位での並列構造の一致の評価

文単位での並列構造の一致についての評価結果を**表 4** に示す．なお，評価対象となる文を下記に分類して評価を行っている．

- **All**：並列構造を持つ全ての文
- **Simple**：二つの並列句から成る並列構造を一つだけ持つ文
- **Not Simple**：三つ以上の並列句から成る並列構造を持つか，複数の並列構造を持つ文
- **Consecutive**：三つ以上の並列句から成る並列構造を持つ文
- **Multiple**：複数の並列構造を持つ文

なお Consecutive と Multiple は，「三つ以上の並列句から

表 4　文単位での並列構造の完全一致率

（単位：%）

モデル	対象	Development	Test
Ours	All	437 / 673 = **64.93**	532 / 873 = 60.93
	- Simple	332 / 481 = 69.02	403 / 609 = 66.17
	- Not Simple	105 / 192 = **54.68**	129 / 264 = 48.86
	- Consecutive	41 / 66 = **62.12**	48 / 96 = 50.00
	- Multiple	74 / 146 = **50.68**	88 / 197 = 44.67
Teranishi+ 17 :re-impl+ext	All	437 / 673 = **64.93**	549 / 873 = **62.88**
	- Simple	338 / 481 = **70.27**	416 / 609 = **68.30**
	- Not Simple	99 / 192 = 51.56	133 / 264 = **50.37**
	- Consecutive	37 / 66 = 56.06	53 / 96 = **55.20**
	- Multiple	70 / 146 = 47.94	89 / 197 = **45.17**

成る並列構造を少なくとも一つ持ち，かつ複数の並列構造から成る文」が重複している．

文単位での並列構造の完全一致率の評価において，提案手法はベースラインのモデルと同程度の性能となった．ベースラインのモデルは寺西ら [10] のモデルを改良し，文の複数の並列構造を同時にデコーディングを行っており，複数の並列構造を持つ文（Multiple）において文法規則を用いた提案手法と同程度の解析性能を得ている．三つ以上の並列句から成る並列構造を持つ文（Consecutive）において，提案手法は *false* 補助並列キーに対してより頑健に解析できることが期待されていたが，ベースラインの手法と同程度の性能に留まった．今後は異なるドメインのコーパスを用いて実験を行い，補助並列キーの出現数や真偽による提案手法の解析への影響を定性的に調べる必要がある．また，提案手法では文法規則を用いて構文木を導出しているものの，ニューラルネットワークのモデルはベースラインと同一であり，今後は CKY アルゴリズムの適用により適したアーキテクチャを用いることが課題である．

5. 考察

先行研究と比較をして提案手法の再現率が低いことから，提案手法によって導出できない並列構造が少なくないことが考えられる．そこで Penn Treebank コーパスにアノテーションされた並列構造を含む文のうち，提案手法で用いた表 1 の文法規則によって並列構造木を導出可能な文の数を調べた（**表 5**）．本研究で提案した表 1 の文法規則から正しい並列構造木が導出不可能な文が少なくない数存在することがわかった．評価データにおいて並列構造木が導出できない例を調べたところ，**図 4** のような文が検出された．図 4.1 の例では，並列キー *and* の直後の副詞が並列句に含まれていない．並列キー前後の副詞句を並列句に含むかどうかは明確な決まりはなく，アノテーションの方針によって異なる．副詞句に限らず並列キーの前後の句を並列句に含まないような並列構造を導出できるよう，提案手法の文法規則を拡張することを検討する．図 4.2 の例では，並列キー *but* の直後に出現する副詞 *then* が並列句に含まれていない．提案手法では並列キーの対象となる語は一つの単語から成るもののみとしているが，“and then” や “but then” のような複数の語から成る表現が等位接続の働きをする場合は，それら一連の語を複数の語から成る並列キーとしてみなすよう手法を拡張することについても検討する．図 4.3 の例は，二重引用符の始まりが並列句に含まれているが二重引用符の終わりが並列句に含まれておらず，アノテーションの誤りと考えられる．

表 5　文法規則により並列構造が導出可能な文の数・割合

（単位：%）

	# 並列構造が導出可能な文の数・割合
コーパス全体	17739 / 19095 = 92.89
訓練データ	14382 / 15481 = 92.90
開発データ	643 / 673 = 95.54
評価データ	793 / 873 = 90.83

6. 関連研究

6.1 並列句の類似性に基づく手法

日本語における並列句の範囲同定のタスクにおいて，黒橋ら [6] は並列句の類似度計算にチャートを用い，動的計画法によって並列構造の検出と範囲の同定を行った．新保ら [9] は英語の並列構造解析において，系列アラインメントと単語・品詞・形態情報に基づく素性により，並列句の内部の複数単語間の類似度を動的計画法を用いて計算した．新保らのモデルは入れ子となる並列構造を扱うことができなかったが，原ら [4] は新保らの手法を拡張し，並列構造を木として導出するためのルールを設けることで，複数の並列構造・三つ以上の並列句について取り扱った．原らの手法では個々の並列構造に対して並列句の類似度を計算し，複数の並列構造のスコアの総和が最も高くなるような並列構造の範囲の組み合わせを文全体の並列構造の木として解析した．花元ら [3] は HPSG の構文解析器を拡張し，原ら [4] のモデルと組み合わせて使用し，並列句の範囲を双対分解を用いて同定する手法を提案した．本研究の提案

1 One obvious place to attach [a capital-gains tax cut] , and perhaps [other popular items stripped from the deficit-reduction bill] , is the legislation to raise the federal borrowing limit .
2 Dealers “ would [give you a quote] , but then [refuse to make the trade] . ”
3 He [thinks another crash is “ unlikely] , ” and [says he was “ nibbling at ” selected stocks during Friday ’s plunge] .

図 4 並列構造が導出不可能な文

手法は原ら [4] が提案する並列構造を木として導出する方法を拡張したものである．

6.2 ニューラルネットワークによる手法

Ficler ら [2] は英語の並列構造解析におけるニューラルネットワークを用いた手法を提案した．Ficler らの手法は並列キーに対して外部の構文解析器（Berkeley Parser [8]）を利用して並列句ペアの候補をあらかじめ絞っており，残った候補に対してスコア付けをして最もスコアの高い並列句ペアを同定している．並列句の類似性のみならず可換性についてもニューラルネットワークを利用した特徴ベクトルの計算を行っており，非ニューラルネットワークによる手法と比較して解析精度が向上した．しかしながら，並列キーの前後の並列句のみを解析対象としている点や文の複数の並列構造を独立に決定している点，構文解析の結果に強く依存している点で課題が残る．寺西ら [10] は構文解析の結果を用いずにニューラルネットワークのみから並列句の類似性・可換性の特徴ベクトルの計算を行う手法を提案した．また，寺西らの手法は並列句ペアの特徴ベクトルの計算が簡易・高速であるため，並列句ペアの候補をあらかじめ限定せずに解析を行っている．寺西らは三つ以上の並列句から成る並列構造についても実験による解析器の性能評価を行っているものの，三つ以上の並列句に対するデコーディングは簡易なルールに基づくヒューリスティックな方法であるため，実際の解析器の利用において問題がある．また，Ficler らと同様に複数の並列構造を同時に解析することができていない点でも課題がある．本研究では，寺西ら [10] の提案したモデルを原ら [4] による手法と組み合わせることで，三つ以上の並列句や複数の並列構造を解析可能にしている．

7. おわりに

本研究では文の並列構造を木として導出できる文脈自由文法の規則を定義し，CKY アルゴリズムによる構文解析とニューラルネットワークによる並列構造のスコア付けを組み合わせて用いる方法を提案した．提案手法は三つ以上の並列句を含む並列構造や複数の並列構造を解析できる点において，先行研究の課題を一部解決している．提案手法は先行研究と同等以上の解析性能を達成したものの，提案した文法規則で導出可能な並列構造が限られている．

今後の課題として，第一に取り扱える並列キー・並列構造を増加させるために文法規則を拡張することが挙げられる．導出可能な並列構造を増やすことでさらなる解析精度の向上が見込まれる．また，提案手法のニューラルネットワークのモデルは先行研究と同一のモデルを用いているため，CKY アルゴリズムによる構文解析との併用により適したアーキテクチャを構築することも必要である．これらの課題を解決するとともに，異なるドメインにおける提案手法の有効性の検証・性能評価や並列構造のアノテーションの方針策定についても取り組む．

謝辞 本研究は，一部 JST CREST(課題番号: JPMJCR1513) の支援を受けて行った．

参考文献

[1] Ficler, J. and Goldberg, Y.: Coordination Annotation Extension in the Penn Tree Bank, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, Association for Computational Linguistics, pp. 834–842 (online), available from ⟨http://www.aclweb.org/anthology/P16-1079⟩ (2016).
[2] Ficler, J. and Goldberg, Y.: A Neural Network for Coordination Boundary Prediction, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, Association for Computational Linguistics, pp. 23–32 (online), available from ⟨https://aclweb.org/anthology/D16-1003⟩ (2016).
[3] Hanamoto, A., Matsuzaki, T. and Tsujii, J.: Coordination Structure Analysis using Dual Decomposition, *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, Association for Computational Linguistics, pp. 430–438 (online), available from ⟨http://www.aclweb.org/anthology/E12-1044⟩ (2012).
[4] Hara, K., Shimbo, M., Okuma, H. and Matsumoto, Y.: Coordinate Structure Analysis with Global Structural Constraints and Alignment-Based Local Features, *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore, Association for Computational Linguistics, pp. 967–975 (online), available from ⟨http://www.aclweb.org/anthology/P/P09/P09-1109⟩ (2009).
[5] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization (2014).
[6] Kurohashi, S. and Nagao, M.: A Syntactic Analysis Method of Long Japanese Sentences Based on the Detection of Conjunctive Structures, *Computational Linguistics*, Vol. 20, No. 4, pp. 507–534 (online), available from ⟨http://www.aclweb.org/anthology/J94-4001⟩ (1994).
[7] Parker, R., Graff, D., Kong, J., Chen, K. and Maeda, K.: English Gigaword Fifth Edition (2011).

[8] Petrov, S., Barrett, L., Thibaux, R. and Klein, D.: Learning Accurate, Compact, and Interpretable Tree Annotation, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, Association for Computational Linguistics, pp. 433–440 (online), DOI: 10.3115/1220175.1220230 (2006).

[9] Shimbo, M. and Hara, K.: A Discriminative Learning Model for Coordinate Conjunctions, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, Association for Computational Linguistics, pp. 610–619 (online), available from ⟨http://www.aclweb.org/anthology/D/D07/D07-1064⟩ (2007).

[10] Teranishi, H., Shindo, H. and Matsumoto, Y.: Coordination Boundary Identification with Similarity and Replaceability, *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Taipei, Taiwan, Asian Federation of Natural Language Processing, pp. 264–272 (online), available from ⟨http://www.aclweb.org/anthology/I17-1027⟩ (2017).

[11] Toutanova, K., Klein, D., Manning, C. D. and Singer, Y.: Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network, *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, Stroudsburg, PA, USA, Association for Computational Linguistics, pp. 173–180 (online), available from ⟨http://www.aclweb.org/anthology/N03-1033⟩ (2003).

付　　録

A.1 句ペアのスコア計算の並列化

CKY アルゴリズムによる確率文脈自由文法の構文解析では，動的計画法によりスコアを計算した後により上位のノードのスコアを計算するため，動的計画法の適用に先立って計算が必要な COORD ノードのみを列挙することができない．しかしながら，あらかじめ全ての導出可能な COORD ノードの計算を行うことは 2.3 節で示したとおり現実的ではない（計算可能であるならば CKY アルゴリズムと動的計画法を適用する動機がない）．ニューラルネットワークによるスコア計算にナイーブに CKY アルゴリズムと動的計画法を適用した場合，並列構造のスコア計算は表 1 に示した規則により COORD ノードを導出する都度行われるため，スコア計算の並列化ができず解析の速度が低下する（Algorithm 1）．さらに，文によって構文木のノードの生成過程が異なるため異なる CKY テーブルを用いることになるが，文ごと・COORD ノードごとにスコア計算を行った場合はミニバッチ化による複数文の学習・解析において文の数に比して速度が低下する．本節では COORD ノードごと・文ごとのスコア計算の問題を解消するためのスコア計算の並列化について説明をする．

Algorithm 1 Naive Binary Production

Input: $table, back, n\_words, level \geq 2, grammar$
Output: NONE

for $begin \leftarrow 0$ **to** $n\_words - level$ **do**
  $end \leftarrow begin + level$
  for $mid \leftarrow begin + 1$ **to** $end - 1$ **do**
    for each $\{R_A \mid R_A \rightarrow R_B R_C \in grammar,$
      $table[begin, mid, R_B] > \text{FLOAT\_MIN},$
      $table[mid, end, R_C] > \text{FLOAT\_MIN}\}$ **do**
      if R_A *is* COORD **then**
        $score \leftarrow \text{ScoreFunc}(\dots)$
      else
        $score \leftarrow 0$
      end if
      $score \leftarrow score + table[begin, mid, R_B] + table[mid, end, R_C]$
      if $score > table[begin, end, R_A]$ **then**
        $table[begin, end, R_A] \leftarrow score$
        $back[begin, end, R_A] \leftarrow \{mid, R_B, R_C\}$
      end if
    end for
  end for
end for

A.1.1 同一スパン長の並列構造のスコア計算並列化

本小節では CKY アルゴリズムによる句（ノード）の導出の順番は，スパンの長さの小さい句から導出していくことを前提とする．スパンの長さの小さい句から導出する場合，図 **A·1** のような CKY テーブルにおいて構文解析は左下から右上へと進んでいく．スパンの長さが i となる句を導出する CKY アルゴリズムの段階を $level = i$ と呼ぶと，同一レベルにあるノードはその導出において互いに依存していないことから，並列してスコアを算出することができる．図 A·1 の例では，緑の同一線上に並ぶ句のスコアは並列に計算ができ，赤い枠で囲まれた並列構造の句が実際にスコア計算を要するノードで並列化が行われる部分を示している．スコア計算の並列化によって，図 A·1 の例で COORD のノードに対するスコア計算の関数の適用回数が 12 回から 5 回に減っている．スコア計算に並列化に対応した疑似コードを Algorithm 2 に示す．

A.1.2 ミニバッチ確率的勾配法による学習時の複数文の計算並列化

前小節で示したとおり，CKY テーブルにおいて同一レベルのノードに対するスコア計算は並列化が可能である．複数文でのスコア計算の並列化は，スコア計算が必要になるノードが出現するレベルまで各文の構文解析を進め，構文解析が終了していない文の全てがスコア計算が必要なレベルに達した場合に一括してスコア計算を行うことで達成される．疑似コードを Algorithm 3 に示す．

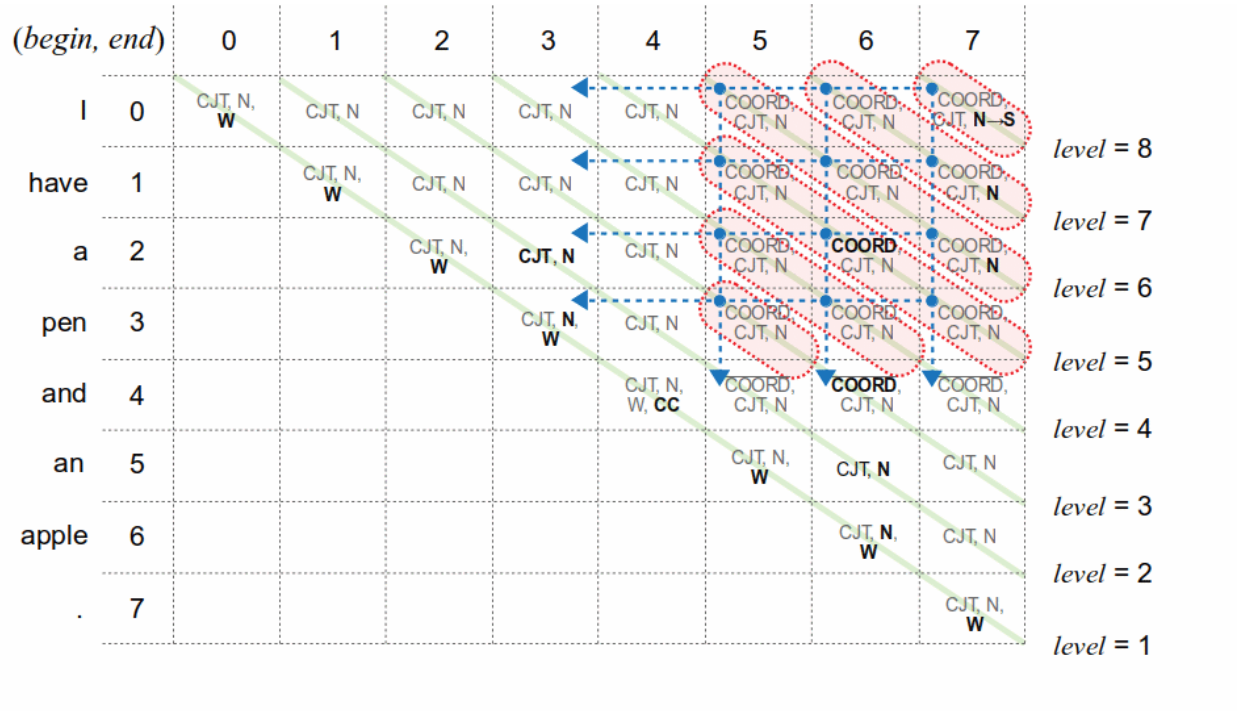

図 **A·1**　並列計算可能な COORD ノード

A.2　実験に用いたハイパーパラメータ

4 節の実験で用いたハイパーパラメータの設定を表 **A·1** に示す.

表 **A·1**　Penn Treebank の実験に使用した最終的なハイパーパラメータの設定

パラメータ	値
LSTM の隠れ状態のベクトルの次元数	600
多層パーセプトロンの隠れ層のユニット数	2,400
Dropout の適用の比率	0.0
L2 正則化の強度 λ	0.0
損失関数のマージン α	3.0
N-best 学習で用いる予測構造木数 N	8
事前学習のエポック数	4

Algorithm 2 Binary Production with Parallel Computation

Input: $table, back, n\_words, level \geq 2, grammar$
Output: NONE

$\mathcal{C} \leftarrow \emptyset$
for $begin \leftarrow 0$ **to** $n\_words - level$ **do**
  $end \leftarrow begin + level$
  for $mid \leftarrow begin + 1$ **to** $end - 1$ **do**
    for each $\{R_A \mid R_A \rightarrow R_B R_C \in grammar,$
      $table[begin, mid, R_B] > \text{FLOAT\_MIN},$
      $table[mid, end, R_C] > \text{FLOAT\_MIN}\}$ **do**
      if R_A *is* COORD **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(begin, mid, end, R_B, R_C)\}$
      end if
    end for
  end for
end for
$scores \leftarrow \text{BatchedScoreFunc}(\mathcal{C}, \dots)$
for $begin \leftarrow 0$ **to** $n\_words - level$ **do**
  $end \leftarrow begin + level$
  for $mid \leftarrow begin + 1$ **to** $end - 1$ **do**
    for each $\{R_A \mid R_A \rightarrow R_B R_C \in grammar,$
      $table[begin, mid, R_B] > \text{FLOAT\_MIN},$
      $table[mid, end, R_C] > \text{FLOAT\_MIN}\}$ **do**
      if R_A *is* COORD **then**
        $score \leftarrow scores[begin, mid, end, R_B, R_C]$
      else
        $score \leftarrow 0$
      end if
      $score \leftarrow score + table[begin, mid, R_B] + table[mid, end, R_C]$
      if $score > table[begin, end, R_A]$ **then**
        $table[begin, end, R_A] \leftarrow score$
        $back[begin, end, R_A] \leftarrow \{mid, R_B, R_C\}$
      end if
    end for
  end for
end for

Algorithm 3 Batched Binary Production

Input: $S = \{\{table_i, back_i, n\_words_i, level_i\} \mid 0 < i \leq N\}$, $grammar$
Output: NONE

$\mathcal{T} \leftarrow \mathcal{S}$
$\mathcal{C} \leftarrow \emptyset$
while $\mathcal{T} \neq \emptyset$ **do**
  $scores \leftarrow \text{BatchedScoreFunc}(\mathcal{C}, \dots)$
  for each $\{table_i, back_i, n\_words_i, level_i\} \in \mathcal{T}$ **do**
    if $level_i = 2$ **then**
      $\mathcal{C}_i \leftarrow \emptyset$
    else
      $\mathcal{C}_i \in \mathcal{C}$
    end if
    while true **do**
      ApplyBinaryRules($table_i$, $back_i, n\_words_i, level_i, scores, grammar$)
      if $level_i = n\_words_i$ **then**
        $\mathcal{T} = \mathcal{T} \setminus \{table_i, back_i, n\_words_i, level_i\}$
        break
      end if
      $level_i \leftarrow level_i + 1$
      $\mathcal{C}_i \leftarrow \text{EnumerateNextCoord}($ $table_i, n\_words_i, lavel_i, grammar)$
      if $\mathcal{C}_i \neq \emptyset$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathcal{C}_i\}$
        break
      end if
    end while
  end for
end while