NAIST-IS-MT1651047

修士論文

チャンクに基づいた逐次型統合パージング

小比田 涼介

2018 年 3 月 15 日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

小比田 涼介

審査委員：

松本　裕治	教授	（主指導教員）
中村　哲	教授	（副指導教員）
新保　仁	准教授	（副指導教員）
進藤　裕之	助教	（副指導教員）
能地　宏	助教	（副指導教員）

チャンクに基づいた逐次型統合パージング*

小比田 涼介

内容梗概

逐次処理は音声処理や対話システムなど、入力を漸進的に処理する場合に必要とされ、システムの素早い反応や自然な振る舞いの達成に不可欠な技術である．係り受け解析に関しては、遷移型の解析器により逐次解析が提案されてきたが、実テキストを入力とし、形態素解析及び係り受け解析の双方を逐次的に実行する手法に関しては未だ議論が十分になされているとは言い難い．本論では連続する語の塊（チャンク）に基づいた逐次型統合解析手法を提案し、従来の非逐次型の解析器に劣らない精度を達成した．考察では、逐次型統合解析における困難点や有効素性について分析し、また、本手法の多言語拡張についても議論する．

キーワード

逐次解析, 係り受け解析, 形態素解析

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 修士論文, NAIST-IS-MT1651047, 2018 年 3 月 15 日.

Incremental Joint Chunk Parsing*

Ryosuke Kohita

Abstract

Incremental processing is necessary in the fields of speech processing or dialogue system to achieve a system's quick responses or natural behaviors. Syntactic analysis, however, has missed full incremental algorithms to take a raw text as an input and conduct tokenization, tagging and dependency parsing at the same time. In this paper, we propose a novel method of incremental joint parsing based on *chunk* and achieve high-performance as well as existing non-incremental parsers. We also give qualitative discussions about the difficulties and important features for incremental processing.

Keywords:

incremental processing, dependency parsing, morphological analysis

*Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1651047, March 15, 2018.

謝 辞

本論文の執筆も含め、２年間の研究で丁寧に指導してくださった能地宏先生に感謝します. そして、素晴らしい研究環境を提供してくださっている松本先生を始めとするスタッフ陣の皆様にも心から感謝いたします. おかげさまで充実した研究生活を送ることができ、今後の大きな糧となりました. また、研究仲間として良い先輩や同期、後輩にも恵まれました. 様々なものに対する色々な議論から学ぶことは非常に多く、とても有意義でした. ありがとうございます. 本研究は基礎解析の一部ですが、その活用によって何かの役に立つような仕組みをこれからも考えていきたいと思います.

目 次

図 目 次

表 目 次

第1章　序論

逐次解析は音声認識や対話システムなど、入力を最後まで待たずして処理を行いたい場面で重要となる技術である [23]．素早いレスポンスを可能にするだけでなく、対話におけるターンテイキングや部分的な理解からの返答などより自然な振る舞いをするシステムにも繋がり [2, 3]、対話的なシステムが台頭している今日ますます需要は高まっていくだろう．また、人間の逐次的な認知メカニズムをモデル化するといった科学的な分野に関しても貢献できる [10]．

本論文では実テキストを入力とし、単語分割・品詞タグ付け・係り受け解析の全てを逐次的に行う手法を提案する．逐次解析の性質上、局所的にならざるを得ない形態素解析については、チャンクという語の塊を用いることによって、出来る限り周辺文脈を考慮しながらの推定を可能にした．日本語を対象にした実験では、実用的な速度を保ちつつ、非逐次型の従来の解析器に劣らない精度を達成した．さらに、どのような素性が有効であるのか、どういった誤りが起きやすいのかなど定性分析も行い、今後の逐次処理の出発点としての役割も果たすものである．加えて、本手法は多言語への拡張も容易に可能であり、考察部分ではその方針についても触れる．

論文構成は以下の通りである．2 章で先行研究に触れたのち、3 章で提案手法について解説する．続く 4 章で京大コーパス [17] と KNB コーパス [36] を用いた実験を行い、非逐次型の従来の日本語解析器（CaboCha[35]、J.DepP[31]) との比較を行う．5 章では、誤り分析や提案手法の改善点、多言語への拡張について議論し、6 章にてまとめを行う．

第2章　関連研究

係り受け解析とは、与えられた入力文に対して、その文内の単語の依存関係を表す係り受け木を求める技術であるが、逐次的に行なう手法としては遷移型解析器（transition-based parsing）が提案されている [30, 25] . この解析器では、文を左から順に入力していき、係り受け木を局所的に決定していくことで逐次的な処理を可能にしている [26, 27]．さらに、文全体の全域探索を行うグラフ型の解析器 [22] に劣らない精度を保ちながら、高速に動作するアルゴリズム（$O(n)$）となっている [21]. より逐次性に焦点を当てた遷移型解析器も提案されていて、それらの手法では最終的に出力される係り受け木の形を予測しながら処理を進めていくため、生成的な一面を併せ持つ [9, 4, 19]．両者の違いは、処理途中の木の保持の仕方にあり、前者は解析対象の文に対する部分木を持つのに対し、後者は未入力のノードを予測補完して、常に一つの木を構築する．このように遷移型の解析器には、逐次処理が可能であるということのほか、高い性能や計算コストなど実用上も重要となる条件を満たしている. しかしながら、これらのシステムは基本的に単語分割や品詞といった特徴量は事前解析済みであることを想定しており、そのままでは逐次解析を行うことができない．

形態素解析と係り受け解析を同時に行う遷移型解析器としては、文字あるいは単語を入力するたびに、その部分の分割や品詞を推定するという手法が提案されている [13, 6]．品詞タグ付けと係り受け解析の同時解析では Bohnet らの [6] があり、単語分割も含めた同時解析では Hatori らに端を発する中国語解析がある [14, 33, 20]．これらの手法は、同時解析によって単語分割失敗による誤った品詞付与及び係り受けといったエラー伝搬を軽減したり、係り受けと品詞など相互的な関係によって決まる箇所の推定精度向上などを動機としている. そのため、逐次処理はあくまでも副産物であり、それ自体に関する議論は十分とは言い難い．例えば、実テキストを逐次的に解析していく上で障害となるのが、後続文脈を見ずに形態素解析（単語分割、品詞推定）を行わなければならない点であるが、中国語の同時解析手法は、左から右に順に文字列を捜査していく中で、単語の一文字目を見た時点で予測を行う．これは中国語のような単語一つあたりの文字数が少ない言語だからこそ可能であるが、文字数の多い単語に対しては難しくなるこ

とが予想され、また、格や数、活用形といったより粒度の高い形態素情報を付与しようとするとより難しくなり、ビームサーチなど計算コストのかかる推定が余儀なくされる [20].

本論文で対象とする日本語解析に関しては、これまで逐次処理が可能な解析器は提案されていない. 加えて、文節（チャンク）単位における係り受け解析が汎用性が高いという日本語解析特有の状況もあり [35]、チャンクを逐次解析においてどのように扱うと良いかという問題もある．チャンクとは連続する内容語と付属語の塊であり、形態素情報は各単語に与えられるが（表 2.1）、係り受け情報はチャンクに対して一つのヘッドが与えられる（図 2.1）．チャンクには後続の語の情報が含まれており、それらを利用することで決定的な探索においても後続の語をより広く見ながら逐次解析ができる. また、形態素解析自体に関してもより多くの情報を取り入れた解析が可能になる．次章では、提案手法であるこれら日本語文に関する解析について述べる．

単語	太郎	が	花子	を	褒めた	.
品詞	名詞	助詞	名詞	助詞	動詞	*
詳細品詞	固有名詞	格助詞	固有名詞	格助詞	*	特殊
活用形	*	*	*	*	母音動詞	*
表層形	*	*	*	*	タ形	*

表 2.1: 日本語文形態素

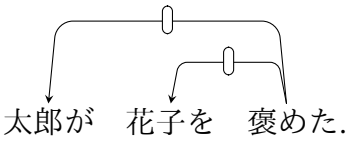

図 2.1: 日本語係り受け木

第3章　提案手法

3.1　基本構造

3.1.1　構成

一般的な遷移型解析器は、スタック（σ）、バッファー（β）、予測アークの集合（A）の３つ組によって構成される [27]. 入力単語は一旦バッファーに保持され、3.1.2 章にて後述する各種遷移を行うことで、スタックに移動させながら単語間の依存関係を決定的に推定していく. この手法の利点は、局所決定的なアルゴリズムによって高速に動作し、かつ、全域探索に劣らない高い性能を実現できる点にある [27].

提案システム s は遷移型解析器を拡張したものであり、基本構造にチャンカー（δ）を加えた４つ組で構成される（$s = (\sigma, \delta, \beta, A)$）. それぞれのデータ構造が保持するものは、スタックが構築したチャンク、チャンカーがチャンク構築中の文字列、バッファーが入力文字列である. チャンカーは形態素解析器の役割も担っており、定義されたチャンクに対して形態素解析を行う. ある状態 s における各トークンの位置はインデックス i で表現し、インデックス 0 はルートに対応する. スタックとバッファーの先頭トークンは $\sigma|i$、$j|\beta$ とし、チャンカーについては $\delta|i|...|j$ によって、先頭から順に i から j までの文字列が入っている状態を表す. また、i から j への係り受けは (j,i)、または、$i \to j$ と表記する. 例えば、文 S の解析における t 時刻目の状態が $s_t = (\sigma|0|i, \delta|j|...|k, h|\beta, A)$ であった時、スタックの先頭には i 番目に構築されたチャンクがあり、2 番目にはルートが入っている. チャンカー内には S の j から k 番目までの文字列が入っており、バッファーの先頭は h 番目の文字（$h = k+1$）である. 実際の解析においては、システムは長さ n の文を受け取り、まず初期状態 $s_0 = (\sigma|0, \delta, 1|2|...|n|\beta, A = \emptyset)$ に遷移する. その後、終了状態 $s_t = (\sigma|0, \delta, \beta, A)$ に達するまで状態遷移を繰り返す. 終了状態 s_t における予測アークの集合 A には、一つの係り受け木が保持されている.

図 3.1 に、“太郎が花子を褒めた。” という文の解析における途中状態を例示する. 左から順にスタック、チャンカー、バッファーを表しており、スタック内部に

はチャンク、チャンカーとバッファー内部には文字が挿入されている. まず、初期状態（t_0）では、スタックはルートのみ、チャンカーは空で、バッファー内に入力文字列が挿入されている. その後、適切な遷移を繰り返すことで正解の構造を構築を目指し、途中状態としては、例えば t_8 では中段のようになる. この段階では、「太郎が」というチャンクは構築済みで既にスタックに挿入されており、チャンカー内では「花子を」に対応する文字列が入力されたところで、この後にチャンク構築ための遷移が行われる. この文の場合、１７回の遷移によって正解の構造が導かれ、t_{17} のような終了状態となる. この時チャンカーとバッファーは空に、スタックの先頭はルートになっており、その下に木が作られる.

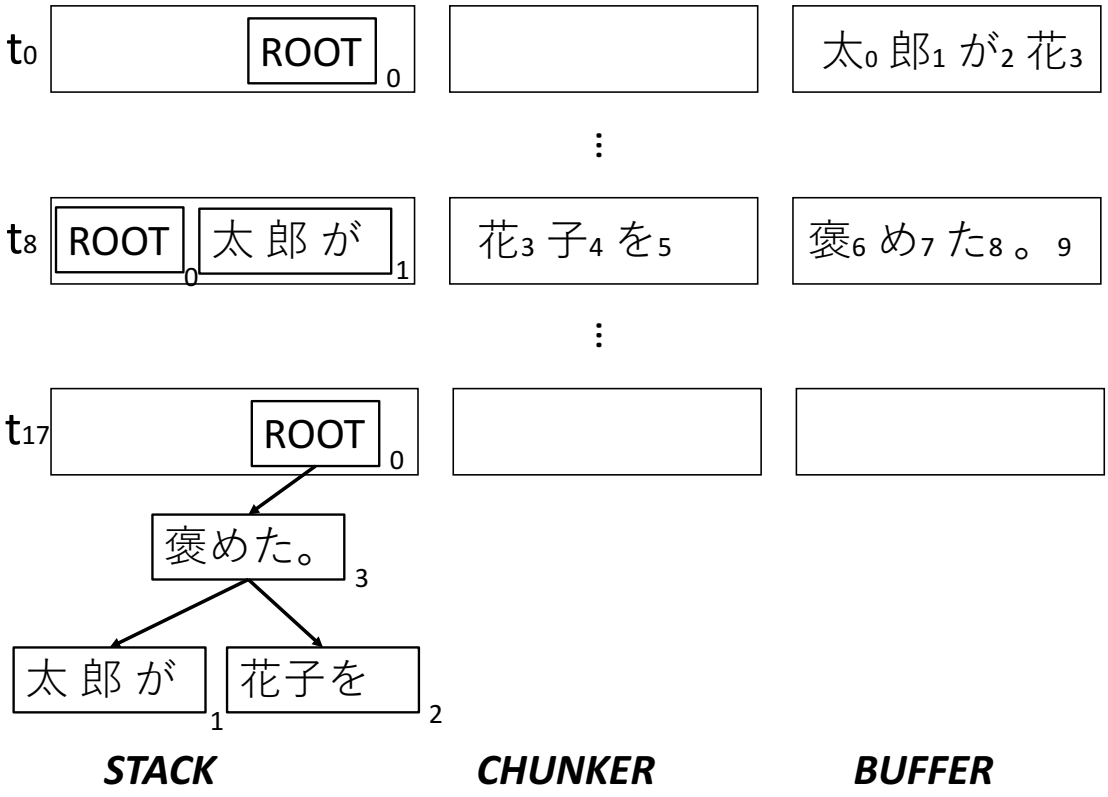

図 3.1: 遷移型解析器動作図

3.1.2 遷移

本システムが用いる遷移は LA（left-arc）、RA（right-arc）、SH（shift）、CH（chunk）の 4 つである（表 3.1）.

遷移	遷移前	遷移後	条件
LA	$(\sigma\|i\|j,\delta,\beta,A)$	$(\sigma\|j,\delta,\beta,A\cup(j,i))$	$i\neq 0$
RA	$(\sigma\|i\|j,\delta,\beta,A)$	$(\sigma\|i,\delta,\beta,A\cup(i,j))$	
SH	$(\sigma,\delta,i\|\beta,A)$	$(\sigma,\delta\|i,\beta,A)$	$\beta\neq\emptyset$
CH	$(\sigma,\delta\|i\|...\|j,\beta,A)$	$(\sigma\|k,\delta,\beta,A)$	$\delta\neq\emptyset$

表 3.1: 遷移リスト

LA 及び RA は Arc-Standard[27] と同一であり、スタックトップ 2 つのチャンクにアークを張り、かつ、子供をスタックから取り除く．ただし、日本語係り受け解析においては、ヘッドが常に右側にあるため、RA は最後にルートを決定する時にのみ使用される．SH はバッファートップの文字をチャンカーに移動させる遷移であり、CH はチャンカー内の文字列をチャンクとしてまとめ、スタックに挿入する遷移である.[1]遷移型解析器における遷移アルゴリズムは Arc-Standard の他に、Arc-Eager など様々な種類が提案されているが [27]、Arc-Standard のみがスタック上のトークンに対してアークを張れるため、今回ベースのアルゴリズムとして採用した. 例えば、Arc-Eager はスタックトップとバッファートップのトークン間でアークを張るため、次の入力単語が定まっていない逐次解析への拡張は容易ではない.

正解の係り受け木を導出する遷移系列はオラクル関数 O によって与えられる（表 3.1.2）．オラクル関数 O は、t 時刻目の状態 s_t を引数にとり、当該状態における正解の遷移を返す関数である．表 3.3 は“太郎が花子を褒めた．”という文に対する解析例である．まず、未入力文字列はバッファーに取り込まれ、SH 遷移によって順にチャンカーへと移されていく（例 $t<4$）．チャンカー内の文字列があるチャンクを構築すると CH 遷移が呼ばれ、文字列をチャンクとしてまとめ、スタックに挿入する（例 $t=4$、「太郎が」）．スタックの 1 番目と 2 番目に位置するチャンクが係り受け関係にあった場合、LA または RA によって該当アークを A に加え、子供となるチャンクをスタックから取り除く（例 $t=14$）．このように本手法は文字の入力順にチャンク同定・内部の形態素解析を貪欲に実施し、実テ

[1]この 4 つの遷移では交差なし（*projective*）の解析に限られるが、並べ替えなど遷移型解析器で提案されてきた手法 [28] を取り入れることで、容易に交差あり（*non-projective*）の解析に拡張することが可能である.

	遷移	状態 s_t	条件
$O(s_t)$	LA	$(\sigma\|i\|j,\delta,\beta,A)$	$(i,j) \in A_g$
	RA	$(\sigma\|i\|j,\delta,\beta,A)$	$(j,i) \in A_g$
	CH	$(\sigma,\delta\|i\|...\|j,k\|\beta,A)$	$k_{chunktag} = \mathrm{B}$
	SH	$(\sigma,\delta,j\|\beta,A)$	

表 3.2: オラクル関数

* 上から順に条件を確認し、条件を満たせば当該遷移を返す.
* A_g は正解の係り受け木である.
* $k_{chunktag}$ はチャンクの開始文字（B）と内部文字（I）を表す（3.2.1 参照）.

キストの逐次解析を可能としている．また、本論では文節をチャンクとして取り扱っているが、チャンクの単位は任意であり、フレーズ単位や単語単位の解析などにもそのまま適用できる.

従来の同時解析との大きな違いは SH と CH にある．従来手法ではチャンカーがなく、SH はバッファーからスタックに文字を直接移動させ、かつ、品詞を予測する役割が与えられている [14, 6]．中国語解析に関していえば、単語の一文字目に対して品詞を予測するため、文字数が多かったり、よりリッチな形態情報を付与したりする際に難しくなり、周辺部分を参照するためにビームサーチなど計算コストのかかるアルゴリズムが必要となる [20]．一方、本手法では SH 遷移によってチャンカー内に文字列を一度保持し、塊となった文字列に対して CH 遷移で形態素解析を適用する．このように、チャンカーをスタックとバッファーの中継として取り入れることで、貪欲探索においても周辺部分を参照することができ、さらに、後述するより高性能な形態素解析手法を組み込むことが可能になる.

3.2 形態素解析

3.2.1 分類器

形態素解析は、表 3.4 のような B / I タグの系列ラベリングタスクとして行い、分類器には BiLSTM を使用する．形態素解析、及び、チャンク構築までのフローを図 3.2 に示す.

BiLSTM はチャンカー内部の文字列 n_i を入力とし、前向き LSTM と後向き LSTM を走らせ、入力文字 i に対応するそれぞれの隠れ層 $\mathbf{f}_i$ と $\mathbf{b}_i$ を得る．$\mathbf{f}_{last}$、及び、

t	遷移	σ	δ	β	A
1	SH	[ROOT$_0$]	[]	[太$_0$, 郎$_1$, が$_2$,]	
2	SH	[ROOT$_0$]	[太$_0$]	[郎$_1$, が$_2$, 花$_3$,]	
3	SH	[ROOT$_0$]	[太$_0$, 郎$_1$]	[が$_2$, 花$_3$, 子$_4$,]	
4	CH	[ROOT$_0$]	[太$_0$, 郎$_1$, が$_2$]	[花$_3$, 子$_4$, を$_5$,]	
5	SH	[ROOT$_0$, 太郎が$_1$]	[]	[花$_3$, 子$_4$, を$_5$,]	
6	SH	[ROOT$_0$, 太郎が$_1$]	[花$_3$,]	[子$_4$, を$_5$, 褒$_6$,]	
7	SH	[ROOT$_0$, 太郎が$_1$]	[花$_3$, 子$_4$]	[を$_5$, 褒$_6$, め$_7$,]	
8	CH	[ROOT$_0$, 太郎が$_1$]	[花$_3$, 子$_4$, を$_5$]	[褒$_6$, め$_7$, た$_8$,]	
9	SH	[太郎が$_1$, 花 子を$_2$]	[]	[褒$_6$, め$_7$, た$_8$,]	
10	SH	[太郎が$_1$, 花 子を$_2$]	[褒$_6$]	[め$_7$, た$_8$, ．$_9$,]	
11	SH	[太郎が$_1$, 花 子を$_2$]	[褒$_6$, め$_7$]	[た$_8$, ．$_9$]	
12	SH	[太郎が$_1$, 花 子を$_2$]	[褒$_6$, め$_7$, た$_8$]	[．$_9$]	
13	CH	[太郎が$_1$, 花 子を$_2$]	[褒$_6$, め$_7$, た$_8$, ．$_9$]	[]	
14	LA	[花子を$_2$, 褒めた．$_3$]	[]	[]	(2,3)
15	LA	[太郎が$_1$, 褒めた．$_3$]	[]	[]	(1,3)
16	RA	[ROOT$_0$, 褒めた．$_3$]	[]	[]	(3,0)
17		[ROOT$_0$]	[]	[]	

表 3.3:“ 太郎が花子を褒めた．”に対する正解遷移系列

$\mathbf{b}_{last}$ はそれぞれの LSTM の最終隠れ層とする. $\mathbf{V}$ は線形写像関数であり、各最終隠れ層からチャンカーの状態ベクトル $\mathbf{q}$ を、各隠れ層 i から文字エンベディング $\mathbf{d}_i$ を得る．ここで文字エンベディングには非線形関数 tanh を適用する．

$$\mathbf{f}_i, \mathbf{b}_i = \text{BiLSTM}(n_i) \tag{3.1}$$

$$\mathbf{q} = \mathbf{V}[\mathbf{f}_{last}; \mathbf{b}_{last}] + \mathbf{e}_q \tag{3.2}$$

$$\mathbf{d_i} = \tanh(\mathbf{V}[\mathbf{f}_i; \mathbf{b}_i] + \mathbf{e}_d) \tag{3.3}$$

各文字エンベディング $\mathbf{d}_i$ に対して、品詞 、詳細品詞、活用形、表層形に対応する重み（$\mathbf{W}_{[p|d|c|s]}$）でそれぞれ線形和を取ったものが $\mathbf{o}_{i[p|d|c|s]}$ である. こちらにソフトマックス関数を適用し、予測タグ $t_{[p|d|c|s]}$ を得る．

$$\mathbf{o}_{i[p|d|c|s]} = \mathbf{W}_{[p|d|c|s]}\mathbf{d}_i + \mathbf{e}_{[p|d|c|s]} \tag{3.4}$$

$$t_{i[p|d|c|s]} = \text{softmax}(\mathbf{o}_{i[p|d|c|s]}) \tag{3.5}$$

タグ分類	太	郎	が	花	子	を	褒	め	た	.
チャンク	B	I	I	B	I	I	B	I	I	I
単語	B	I	B	B	I	B	B	I	I	B
品詞	名詞	I	助詞	名詞	I	助詞	動詞	I	I	*
詳細品詞	固有名詞	I	格助詞	固有名詞	I	格助詞	*	I	I	特殊
活用形	*	I	*	*	I	*	母音動詞	I	I	句点
表層形	*	I	*	*	I	*	タ形	I	I	*

表 3.4: 形態素解析の教師データ

単語分割タグ（$t_i[ws]$）に関しては、$\mathbf{d}_i$ と他のタグの出力層（$\mathbf{o}_{i[p|d|c|s]}$ ）を線形写像関数 $\mathbf{U}$ にかけ、同じくソフトマックス関数によって予測する．

$$t_{i[ws]} = \mathrm{softmax}(\mathbf{U}[\mathbf{d}_i; \mathbf{o}_{i[p]}; \mathbf{o}_{i[d]}; \mathbf{o}_{i[c]}; \mathbf{o}_{i[s]}] + \mathbf{e}_{ws}) \tag{3.6}$$

一連の形態素解析はチャンカーに新たな文字列が加わった際（ = SH 毎）に行われる．CH 遷移が呼ばれた時点での解析結果が最終的な予測となり、チャンクベクターの構成要素として用いられる（3.2.2 参照）．

3.2.2 チャンク構築

ここでは j 番目のチャンクベクター $\mathbf{k}_j$ の構築手順について述べる．CH 遷移が引き金となり、当該時刻において予測された単語分割タグ（$t_{i[ws]}$）に従って、チャンカー内の文字列を分割し、対応する各種エンベディング（単語 : $\mathbf{p}_{i[w]}$、学習済み単語 : $\mathbf{p}_{i[t]}$、品詞 : $\mathbf{p}_{i[p]}$、詳細品詞 $\mathbf{p}_{i[d]}$、活用形 : $\mathbf{p}_{i[c]}$、表層形 : $\mathbf{p}_{i[s]}$）を取得する．この時点で分割された単語数 $\times$ 6 つ（学習済みエンベディングを使わない場合は 5 つ）のエンベディグを保持しており、これらを形態素ごとにそれぞれ平均して、当該チャンクの形態素ベクトル $\mathbf{m}_{[w|t|p|d|c|s]}$ とする．例えば、n 個の単語に分割された場合、チャンクの単語ベクトルである $\mathbf{m}_{[w]}$ は $\mathrm{average}(\mathbf{p}_{i[w]}, \mathbf{p}_{j[w]}, ..., \mathbf{p}_{n[w]})$ によって与えられる．

$$\mathbf{m}_{[w|t|p|d|c|s]} = \mathrm{average}(\mathbf{p}_{i[w|t|p|d|c|s]}) \tag{3.7}$$

その後、線形写像関数 $\mathbf{Q}$ で形態素ベクトル（$\mathbf{m}_{[w|t|p|d|c|s]}$）とチャンカーの状態ベクトル（$\mathbf{q}$）を写像した後、非線形関数 ReLU を通して、チャンクベクトル $\mathbf{k_j}$ を得る．

$$\mathbf{k_j} = \mathrm{ReLU}(\mathbf{Q}[\mathbf{m}_{[w]}; \mathbf{m}_{[t]}; \mathbf{m}_{[p]}; \mathbf{m}_{[d]}; \mathbf{m}_{[c]}; \mathbf{m}_{[s]}; \mathbf{q}] + \mathbf{e}_k) \tag{3.8}$$

$\mathbf{k_j}$ は係り受け木のノードとしてスタックに挿入され、その後の遷移によって係り受けが決定される．

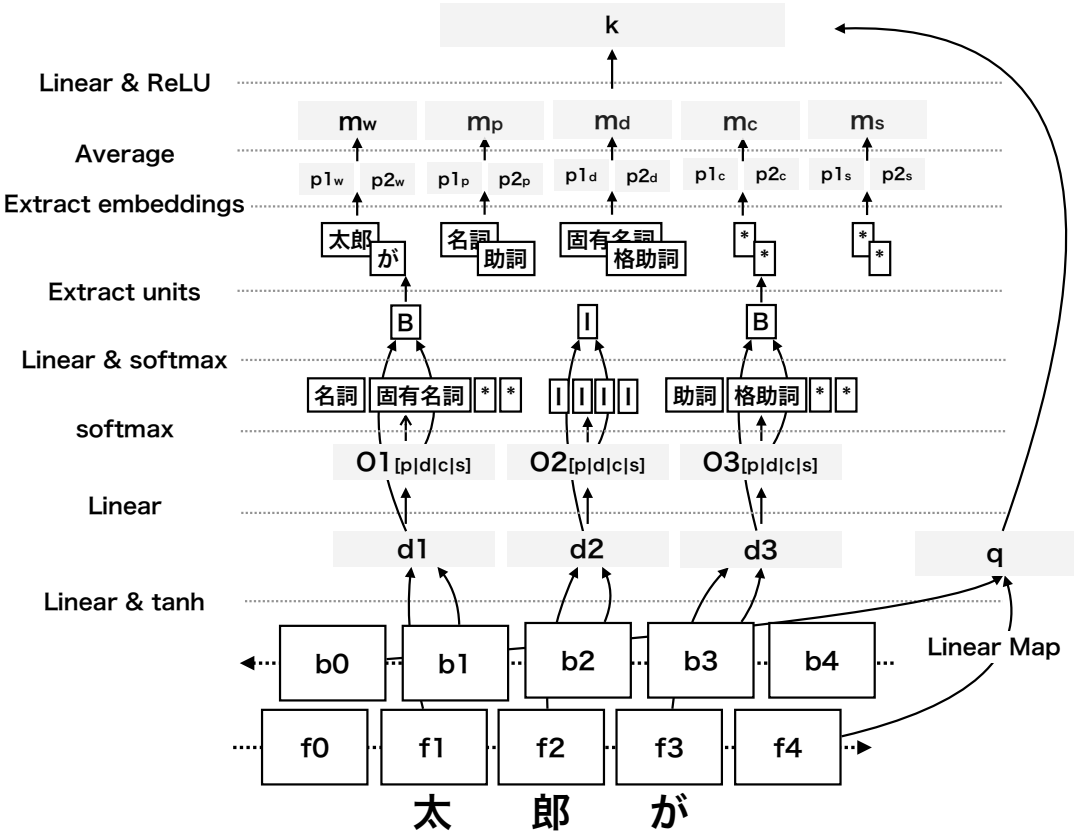

図 3.2: BiLSTM 形態素解析器（学習済みエンベディングなし）

3.3 係り受け解析

3.3.1 分類器

分類器には、Chen & Manning[8] と同様のフィードフォワードニューラルネットワークを採用した（図 3.3）．こちらでは、t 時刻目の状態 s_t から素性関数 f で素性ベクトル $\mathbf{f}_t$ を抽出し、二層の多層パーセプトロンを通して、ソフトマックス関数によって遷移 a_t を決定する．多層パーセプトロンの一層目では非線形関数 ReLU を適用する．

$$\mathbf{f}_t = f(s_t) \tag{3.9}$$

$$\mathbf{h}_1 = \mathrm{ReLU}(\mathbf{W_1}\mathbf{f}_t + \mathbf{e_1}) \tag{3.10}$$

$$\mathbf{h}_2 = \mathbf{W_2}\mathbf{h}_1 + \mathbf{e_2} \tag{3.11}$$

$$a_t = \mathrm{softmax}(\mathbf{h}_2) \tag{3.12}$$

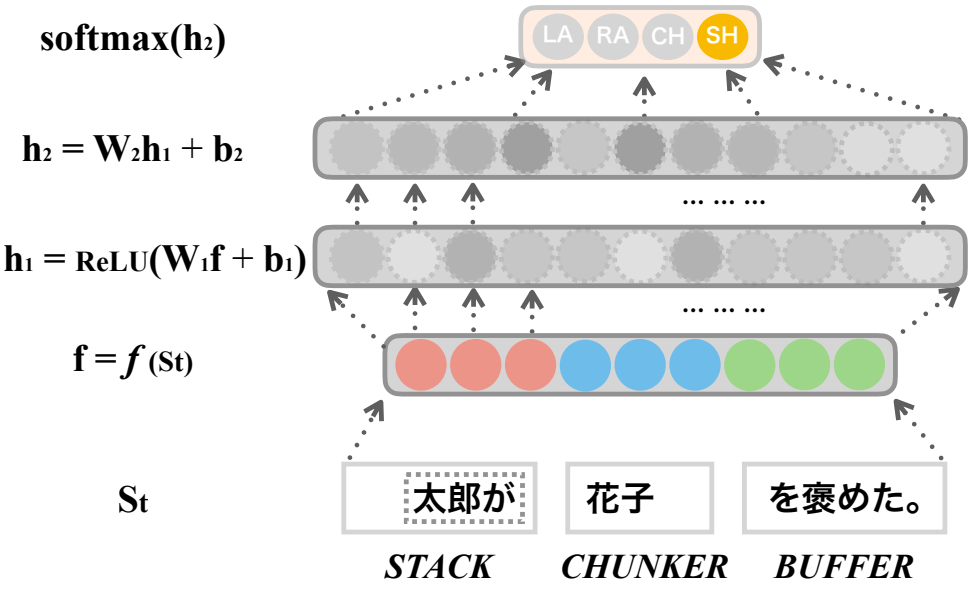

図 3.3: 遷移分類器で用いるニューラルネットワーク

3.3.2 基本特徴量

素性関数 f が抽出する基本特徴量が表 3.5 であり、$\sigma[i]$、$\beta[i]$、$\delta[i]$ は各データ構造の i 番目の要素を意味する．これらの特徴量は Chen & Manning[8] の素性テーブルを参考にし、日本語文節係り受けの特徴を踏まえて設計した．スタックからは、1・2・3 番目のチャンク（$\sigma[0]$〜$\sigma[2]$）、1 番目のチャンクの最左（$\sigma[0]_{lc[0]}$）と左から 2 番目の子供（$\sigma[0]_{lc[1]}$）、2 番目のチャンクの最左の子供（$\sigma[1]_{lc[0]}$）、そして、スタック 1・2・3 番目の単語の表層と品詞（$\sigma_{w[0|1|2]}$、$\sigma_{p[0|1|2]}$）を用いる．バッファーからは、1・2・3 番目の文字（$\beta[0|1|2]$）を、チャンカーからは状態ベクトル（$\mathbf{q}$、3.2.1 参照）と最右の単語（$\delta_{w[i]}$）と品詞（$\delta_{p[i]}$）を特徴量として抽出する．図 3.4 に、“頑張った太郎を花子が褒めた。”という文の解析における、ある状態での抽出特徴量を例示する. 空の特徴量に関しては一律ダミートークンが充てられる.

スタック	$\sigma[0], \sigma[1], \sigma[2], \sigma[0]_{lc[0]}, \sigma[0]_{lc[1]}, \sigma[1]_{lc[0]}$ $\sigma_{w[0]}, \sigma_{w[1]}, \sigma_{w[2]}, \sigma_{p[0]}, \sigma_{p[1]}, \sigma_{p[2]}$
バッファー	$\beta[0], \beta[1], \beta[2]$
チャンカー	$\mathbf{q}, \delta_{w[i]}, \delta_{p[i]}$

表 3.5: 基本特徴量

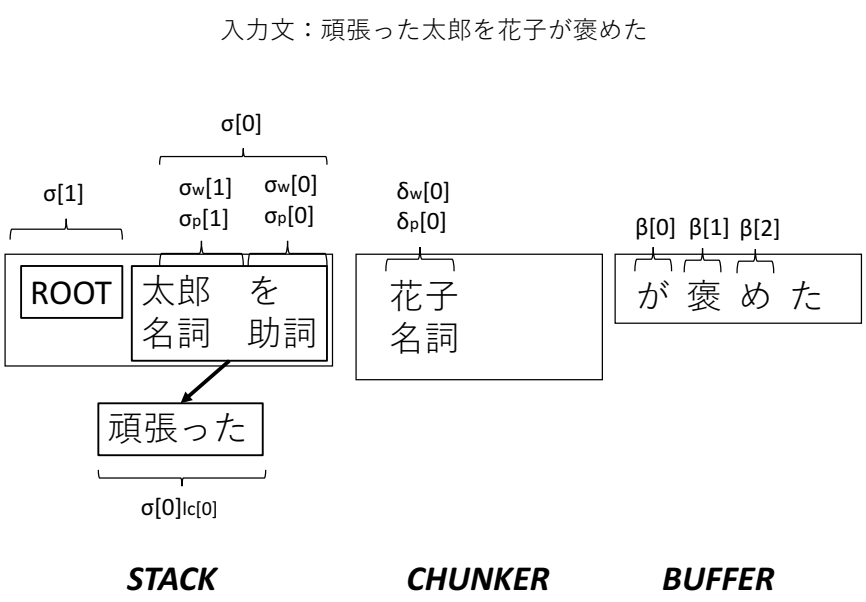

図 3.4: 特徴量抽出例

3.3.3 追加特徴量

逐次的な形態素解析を行うにあたり、Kurita ら [20] の素性テーブルを参考に、いくつかの N-gram 素性と辞書素性を追加した（表 3.6）.

バッファー内の単語を捉えるために、バッファーの i から j 番目までの文字エンベディングを平均したベクトル $\mathbf{u_{i:j}}$ を用いる.

$$\mathbf{u_{i:j}} = \text{average}(\beta[i], ..., \beta[j]) \tag{3.13}$$

バッファートップから 1・2・3 番目までの文字列に関しては、学習済みエンベ

バッファー N-gram	$\mathbf{u}_{0:1}, \mathbf{u}_{0:2}, \mathbf{u}_{0:3}$ $\mathbf{u}_{1:2}, \mathbf{u}_{1:3}, \mathbf{u}_{1:4}$ $\mathbf{u}_{2:3}, \mathbf{u}_{2:4}, \mathbf{u}_{2:5}$ $\mathbf{u}_{3:4}, \mathbf{u}_{3:5}, \mathbf{u}_{4:5}$
バッファー辞書	$\mathbf{r}_{0:1}, \mathbf{r}_{0:2}, \mathbf{r}_{0:3}$
チャンカー N-gram	$\mathbf{v}(= \mathbf{f}_{last} \times \beta[0])$

表 3.6: 追加特徴量

ディングから辞書引きを行い、対応するエンベディング $\mathbf{r}_{0:k}$ を使用した.[2]

$$\mathbf{r}_{0:k} = \mathrm{dict}(\beta[0], ..., \beta[k]) \quad (k = 1, 2, 3) \tag{3.14}$$

チャンカー最後尾とバッファートップの結合を捉えるために、形態素解析器の一部である前向き LSTM の最終層 $\mathbf{f}_{last}$（3.2.1 参照）とバッファートップの文字エンベディングを線形結合したベクトル $\mathbf{v}$ を追加した.

$$\mathbf{v} = \mathbf{W}_{flast}\mathbf{f}_{last} + \mathbf{W}_{btop}\beta[0] + \mathbf{e}_v \tag{3.15}$$

[2]dict() は辞書を引き、対応するエンベディングを返す関数である．辞書に存在しなかった場合、未知語に対応するエンベディングを返す．学習済みエンベディングが与えられていない時は、ランダム初期化のエンベディングに対して辞書引きを行う.

第4章　実験

4.1　データ

実験データには京都大学テキストコーパス v4.0（以下、京大コーパス）[17] と Kyoto University and NTT Blog コーパス（以下、KNB コーパス）[36] を用いた．京大コーパスは新聞記事であり、9501[01-11].KNP、950[1-8]ED.KNP を訓練用、95011[2-3].KNP、9509ED.KNP を開発用、95011[4-7].KNP, 951[0-2]ED.KNP を評価用に使用した．KNB コーパスは携帯電話、京都観光、スポーツ、グルメに関するブログ記事であり、各ドメインの最初 100 文を開発用、次の 100 文を評価用、残り全て 3385 文を訓練用とした．

4.2　比較手法

比較手法としては、CaboCha[35][1] と J.DepP[31][2] を取り上げる．双方ともチャンカー及び係り受けモデルは同じデータセットで再学習したものを用いたが、MeCab（v0.996）は配布されている JUMAN 辞書のモデルを使用した．ただし、両者とも逐次処理ではないという点で、提案手法とは異なる環境を想定した解析器である．

4.3　解析器

本実験での解析器は、ニューラルネットワークフレームワークの DyNet[24][3] を用いて実装した．各種ハイパーパラメーターは以下の通りである．文字、単語、品詞、詳細品詞、活用形、表層形にはそれぞれ 100 次元のエンベディング（ランダム初期化）を用い、チャンクベクター（3.2.2 参照）も 100 次元に設定した．また、学習済みの単語エンベディングとして、Wikipedia から学習したものを使用

[1] https://taku910.github.io/cabocha/
[2] http://www.tkl.iis.u-tokyo.ac.jp/ ynaga/jdepp/
[3] https://github.com/clab/dynet

した [7] [4]．遷移分類器の多層パーセプトロン（3.3.1 参照）は 1 層目（$\mathbf{h}_1$）・2 層目（$\mathbf{h}_2$）とも 200 次元とした．チャンカーの BiLSTM は双方向とも 2 層 100 次元で、ドロップアウトは 33%である．最適化には Adam[18] を用い、学習率等のパラメーターは DyNet の初期値に従った．上記の設定における解析速度は約 30 文 / 秒 であるが、パラメーターを半分ほどに下げることで約 100 文 / 秒となる（精度は 2 から 3pt 低下）．大量のテキストを取り扱うには足りない速度であるが、逐次解析場面では十分であるだろう．

4.4 評価

評価では、J.DepP 付属の評価スクリプトを一部改変したものを用い、形態素解析（単語分割・品詞・詳細品詞・活用形・表層形）の精度（accuracy）、チャンキングの F1 値、係り受け解析の F1 値を報告する．CoNLL Shared Task 2017[32] に習い、本実験でも予測された単語分割・タグ上での係り受け解析結果を評価する．チャンキング誤りを含んでいるため、係り受けの評価においては、係り元と係り先の両方のチャンクが正解データと一致する場合に正解とした．

4.5 結果

モデル	単語分割	品詞	詳細品詞	活用形	表層形	チャンキング	係り受け
基本素性	96.81	95.39	93.92	95.79	95.76	95.92	83.61
+バッファー N-gram	96.85	95.40	94.04	95.77	95.83	96.21	83.97
+バッファー辞書	96.92	95.62	93.99	95.94	95.89	96.95	84.56
+チャンカー N-gram	96.42	95.23	93.83	95.53	95.47	97.01	**85.37**
+ALL	**96.95**	**95.72**	**94.20**	**95.96**	**95.92**	**97.39**	85.24
MeCab + CaboCha	99.15	98.08	97.29	98.34	98.31	94.14	82.64
MeCab + J.DepP						98.67	90.17

表 4.1: 結果（京大コーパス）

表 4.1 は京大コーパスでの実験結果である．提案手法の形態素解析の精度は、全ての特徴量を用いた際（+ALL）、いずれも 95%前後であり、MeCab と比べるとそれぞれ 2 から 3%下がっている．追加素性に関しては、バッファー N-gram とバッファー辞書の寄与が伺えるものの大きな変化ではない．チャンキングの精度は J.DepP が最も高く、次いで提案手法、最後に CaboCha であり、提案手法と J.DepP

[4] https://github.com/facebookresearch/fastText

の差分は 1pt 強であった．チャンカー N-gram 素性の寄与が大きく、前向き LSTM の最終層とバッファートップの文字エンベディングの線形結合によって、チャンクの切れ目を認識しやすくなっている．係り受けの精度も J.DepP が最も高く、F1 値で 90 を上回るなど、非常に正確な解析を行っている．提案手法は 2 番目に良く、CaboCha よりも 3pt 高い．追加素性の中では、バッファー辞書、及び、チャンカー N-gram が役立っている．

モデル	単語分割	品詞	詳細品詞	活用形	表層形	チャンキング	係り受け
基本素性	90.38	85.54	84.09	86.79	86.53	84.37	63.43
+バッファー N-gram	91.04	86.48	84.37	87.88	87.58	85.23	63.88
+バッファー辞書	91.00	86.84	84.96	87.74	87.75	86.76	65.47
+チャンカー N-gram	90.10	85.13	83.39	86.58	86.37	85.31	64.18
+ALL	**91.09**	**86.94**	**84.94**	**88.10**	**87.84**	**87.45**	**67.69**
MeCab + Cabocha	95.22	92.45	90.60	93.11	93.70	83.06	64.79
MeCab + J.DepP						93.12	79.03

表 4.2: 結果（KNB コーパス）

表 4.2 は KNB コーパスでの実験結果である．全体の傾向として京大コーパスの結果と同様であり、形態素解析は MeCab の方が高精度で、チャンキング・係り受けに関しても、J.DepP、提案手法、CaboCha の順に良い．追加素性の影響は異なっており、バッファー N-gram とバッファー辞書の形態素解析への貢献が、全体として 1%前後と京大コーパスの時よりも大きめである．

第5章　考察

5.1　単語分割

形態素解析については、MeCab がラティス構築と CRF によって文全体を考慮して単語分割を行なっているのに対して、逐次解析を想定している本手法は限られた範囲から推定する点で基本的に難しい．特に単語分割がボトルネックとなっている様子が伺えた．現状、形態素解析で直接的に用いている情報は、チャンカー内に存在する文字列から取れるもののみであり、解析済みの部分や以降の入力部分を用いていない．追加素性によるスコアの変化が小さかったのも、それらの素性が遷移分類器でのみ使われ、形態素解析では使われていなかっためと考えられる．そういった前後文脈を考慮させることで、分割精度の向上が期待できる．例えば、(1)(2)の文において、「引き続き」「耐え難く」などの複合動詞を「引き」「続き」や「耐え」「難く」といったように細かく分割してしまうケースが見られたが、(1)のような場合、前文節の「戦後処理問題に」や後文節の「取り組む」が見えていれば、正しく分割ができると考えられる．

(1) ... 戦後処理問題に / “ 引き | 続き ” / 取り組む ...

(2) ... 不平等に / “ 耐え | 難く ” なったのと、 / 独仏 ...

品詞や活用形に関しては、単語分割の精度との比率において、MeCab との間に大きな差異はなく、単語さえ正しく推定できていれば、ローカルな情報からでもうまく推定できるようである．

5.2　チャンキング

一方で、チャンク同定は 97.39 と高い精度を保つことができている．これは日本語のチャンク（文節）が比較的わかりやすい分割単位であることもあるが、チャンクが遷移によって決定されており、遷移の決定には前後文脈や構築済みの部分

木など広範囲にわたる素性が用いられていることも関係しているだろう．チャンク誤りとしては、チャンクを短く見積もってしまう場合が多く見受けられ、これはCH遷移のタイミングが早すぎる際に生じる．例えば、「その」のような、チャンクとしての頻度の高い文字列が、「そのうちに」「その場しのぎ」といった他のチャンクの部分文字列となっている場合、解析器はチャンカーに「その」を保持した時点で、誤ってCH遷移を選択してしまう．追加素性によってチャンカーとバッファーの結合性をうまく捉えることができていたものの、バッファーN-gramに関しては単純に文字ベクトルを平均したベクトルであり、その表現に関して工夫を施すことで上記の問題を緩和する手立てとなるだろう．

5.3 係り受け解析

係り受け解析も、京大コーパスでは実テキストからの逐次解析という環境下で85.37と十分に高い精度を達成している．ただし、崩れたテキストが多く、より口語的であるKNBコーパスでの精度は67.69と低く、J.DepPよりも10pt以上スコアを下げている．原因として考えられるのは学習データの量であり、提案法の複雑なネットワークを学習するには3000文強は明らかに不足している．しかし、実用的な観点として、対話や音声認識など口語場面で逐次解析が必要となるため、こういった口語的なテキストでの解析精度が特に重要である．更に言えば、実際の場面では、認識誤りや「ええっと」「ああ」などのフィラーが存在し、こうした現象への対処も必要となり[15]、より現実的な環境を考慮した改良や検証が望まれる．

5.4 多言語への拡張

本手法でのチャンクの単位は自由であり、語をチャンクとして捉えれば、容易に多言語に拡張可能である．例えば、図5.1の*Taro praises Hanako.*という英語の文に対する遷移は、表5.1のように導出できる．ただし、チャンカー内部に存在する文字列は1つの単語に対応しており、複数の語の塊をチャンクとする場合と比べて、形態素解析器が参照できる範囲が狭くなっている．そのため、正しく解析するためには、前後文脈の素性の組み込みがより重要になると考えられる．

複数単語からなるチャンクを用いた解析に関しては、機械翻訳[29, 34, 16]や係り受け解析[1, 11]などで、昔から現在に至るまで考えられている枠組みであり、意味のある塊というのは言語に依らず想定できる．しかしながら、日本語の文節

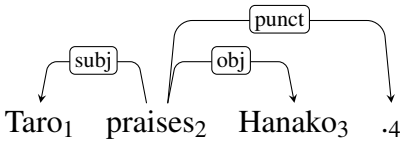

図 5.1: 英語文係り受け木

t	遷移	σ	δ	β
1	SH $\times$ 4	[ROOT]	[]	[T, a,]
2	CH	[ROOT]	[T, a, r, o]	[p, r,]
3	SH $\times$ 7	[ROOT, Taro]	[]	[p, r,]
4	CH	[ROOT, Taro]	[p, r, a, i, s, e, s]	[H, a,]
5	LA	[Taro, praises]	[]	[H, a,]
6	SH $\times$ 6	[ROOT, praises]	[]	[H, a,]
7	CH	[ROOT, praises]	[H, a, n,a ,k,o]	[.]
8	RA	[praises, Hanako]	[]	[.]
9	SH	[ROOT, praises]	[]	[.]
10	CH	[ROOT, praises]	[.]	[]
11	RA	[praises, .]	[]	[]
12	RA	[ROOT, praises]	[]	[]
13		[ROOT]	[]	[]

表 5.1: 図 5.1 の文に対する遷移系列

のように比較的明確な単位を持つ言語は珍しく、どのようにチャンクを定義すべきなのか、どのような定義が効率的な解析へと繋がるのかということに関しては明らかではない．多言語におけるチャンクの定義に関しては、多言語ツリーバンクの Universal Dependencies（UD）が適した分析対象となるだろう．UD では単語単位の係り受けを基本としているものの、全言語が可能な限り同じアノテーション基準に則っている．その基準からチャンクを構築するルールを導き出すことができれば、言語普遍的にチャンクの活用性について議論できるようになるだろう．

5.5 遷移の多様性

遷移型解析器では、ある特定の係り受け木を得られる遷移系列が複数あり、そうした曖昧性をうまく学習することで精度が向上することが知られている [12, 5].

本手法も同様であり、LA と SH のタイミングに選択の余地がある．今回のオラクル関数（表 3.1.2 参照）は、LA は常に SH より優先されるが、実際には、LA のタイミングは、次のチャンクの構成文字列がバッファーを満たす前であれば、いつでも良い．例えば、“ 太郎は優しい花子を褒めた．” という文の解析において、状態 $s_t = ([\text{優しい}, \text{花子を}], [], [\text{褒}, \text{め}, \text{た}, .], A)$ で曖昧性が生じる．本論でのオラクル関数はここで LA を返すが、一旦 SH を選択して、先の入力をある程度見た後に “ 花子を ” → “ 優しい ” のアークを張ることも可能である．先行研究のように遷移の非決定性を解析器にうまく学習させることで、精度が向上すると期待される．

第6章　結論

本論では、形態素解析・係り受け解析の双方を逐次的に行う解析手法を提案した．逐次解析の一つの困難点である後続入力が未定の状態での形態素解析に関しては、チャンクという語の塊を定義することで、効率的、かつ、高精度の解析が可能であることを示唆した. また、チャンクを用いることによって、Bi-LSTM をアルゴリズムで使用するデータ構造内に自然な形で埋め込むことができるようになっており、形態素解析の精度向上に寄与したと考えられる. これらの結果として、非逐次型の解析器に劣らない性能を有しつつ、解析速度も十分に実用的な水準を満たすことができた.

しかしながら、誤り分析を行う中で、単語分割のための特徴量設計、口語解析における諸問題への対応、遷移系列の多様性など工夫の余地が多くの点で残されていることも明らかになった. また、多言語拡張も今後の課題の一つであり、日本語で今回用いたようなチャンクをどのように言語普遍的に定義できるかという問いがあった. この点に関しては、文節という比較的明確なチャンクを持つ日本語での知見が他言語に活かせる部分であり、日本語での研究がより広い価値を持ちうると考えている. 今後もインタラクティブなシステムが増えていくと予想され、これらの課題に取り組み、逐次解析に関する議論を深めていきたい.

参考文献

[1] Steven P. Abney. Parsing by chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, Vol. 44 of *Studies in Linguistics and Philosophy*. Kluwer, 1991.

[2] Timo Baumann. *Incremental Spoken Dialogue Processing: Architecture and Lower-level Components*. PhD thesis, 2013.

[3] Timo Baumann and David Schlangen. Interactional adequacy as a factor in the perception of synthesized speech. In *Proceedings of SSW*, Barcelona, Spain, 2013.

[4] Niels Beuck, Arne Köhn, and Wolfgang Menzel. Incremental parsing and the evaluation of partial dependency analyses. 2011.

[5] Anders Björkelund and Joakim Nivre. Non-deterministic oracles for unrestricted non-projective transition-based dependency parsing. In *Proceedings of the 14th International Conference on Parsing Technologies*, pp. 76–86, Bilbao, Spain, July 2015. Association for Computational Linguistics.

[6] Bernd Bohnet and Joakim Nivre. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1455–1465, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[8] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pp. 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

[9] Vera Demberg and Frank Keller. A psycholinguistically motivated version of tag. In *Proceedings of the 9th international workshop on tree adjoining grammars and related formalisms*, pp. 25–32, 2008.

[10] Vera Demberg-Winterfors. *Broad-coverage model of prediction in human sentence processing*. PhD thesis, 2010.

[11] Yoav Goldberg and Michael Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 742–750, Los Angeles, California, June 2010. Association for Computational Linguistics.

[12] Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pp. 959–976, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[13] Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pp. 1216–1224, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

[14] Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1045–1053, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[15] Matthew Honnibal and Mark Johnson. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, Vol. 2, pp. 131–142, 2014.

[16] Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa, and Weijia Jia. Chunk-based decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pp. 1901–1912, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[17] Daisuke Kawahara, Sadao Kurohashi, and Koiti Hasida. Construction of a japanese relevance-tagged corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Canary Islands - Spain, May 2002. European Language Resources Association (ELRA).

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Arne Köhn and Wolfgang Menzel. Incremental predictive parsing with turboparser. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 803–808, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[20] Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. Neural joint model for transition-based chinese syntactic analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1204–1214, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[21] Ryan McDonald and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 122–131, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[22] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 523–530, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[23] Ian McGraw and Alexander Gruenstein. Estimating word-stability during incremental speech recognition. In *the INTERSPEECH 2012 Conference*, 2012.

[24] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017.

[25] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 149–160, 2003.

[26] Joakim Nivre. Incrementality in deterministic dependency parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together (ACL Workshop)*, pp. 50–57, 2004.

[27] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, Vol. 34, No. 4, pp. 513–554, 2008.

[28] Joakim Nivre, Marco Kuhlmann, and Johan Hall. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pp. 73–76, Paris, France, October 2009. Association for Computational Linguistics.

[29] Taro Watanabe, Eiichiro Sumita, and Hiroshi G. Okuno. Chunk-based statistical translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 303–310, Sapporo, Japan, July 2003. Association for Computational Linguistics.

[30] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, Vol. 3, 2003.

[31] Naoki Yoshinaga and Masaru Kitsuregawa. A self-adaptive classifier for efficient text-stream processing. In *COLING*, pp. 1091–1102. ACL, 2014.

[32] Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan

Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 1–19, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[33] Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1326–1336, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[34] Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. Chunk-based bi-scale decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 580–586, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[35] 工藤拓, 松本裕治. チャンキングの段階適用による日本語係り受け解析. 情報処理学会論文誌, Vol. 43, No. 6, pp. 1834–1842, 2002.

[36] 橋本力, 黒橋禎夫, 河原大輔, 新里圭司, 永田昌明. 構文・照応・評判情報つきブログコーパスの構築. 自然言語処理, 第 18 巻, pp. 175–201, 2011.