Vol.2017-NL-233 No.11
2017/10/25

チャンクに基づいた逐次型統合解析

小比田 涼介[1,a)]　能地 宏[1,b)]　松本 裕治[1,c)]

概要：逐次処理は音声処理や対話システムなどの領域において、システムの素早い反応や自然な振る舞いを達成する上で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、十分な議論がなされているとは言い難い．本論ではチャンクという連続する語の塊に基づいた逐次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、本手法の他言語拡張についても議論する．

キーワード：逐次解析、チャンク、形態素解析、係り受け解析

1. はじめに

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 [19]．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 [9]．

さて、本稿で取り扱う係り受け解析に関しても、遷移型解析器によって逐次処理がすでに提案されいる [21], [26] ．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効率的な処理を可能にしている [22], [23]．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 [4], [8], [17]．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．

しかしながら、これらのシステムのほとんどが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いる点で、完全な逐次解析器とは言えない．これまで提案されてきた遷移型解析器の中で、真に逐次処理が可能なのは、中国語の単語分割・品詞タグ付け・係り受けの同時解析手法である [12], [18], [29]．この手法は、中国語の係り受け解析において元来問題となっていた単語分割のエラー伝搬について、係り受けとの同時予測によって解決しようとしたものである．そのため、あくまでも中国語処理に特化したものであり、実テキストの逐次処理ということに関しては十分に議論されているとは言い難い．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、こちらの手法における品詞推定は、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なるとより難しくなるだろう．

以上を踏まえ、本稿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．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．

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] kohita.ryosuke.kj9@is.naist.jp
[b)] noji@is.naist.jp
[c)] matsu@is.naist.jp

論文構成は以下の通りである．まず、2 章で提案手法について解説する．続く 3 章で京大コーパス [15] と KNB コーパス [32] を用いた実験を行い、非逐次型の従来の日本語解析器（CaboCha[31]、J.DepP[27]）との比較や逐次解析における困難点などを議論する．4 章では、提案手法の改善点や多言語の拡張等について紹介し、5 章にてまとめを行う．

2. 提案手法

2.1 基本構造

提案システムは基本的な遷移型解析器の拡張であり、スタック（σ）、チャンカー（δ）、バッファー（β）の 3 つのデータ構造と予測アークの集合（A）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、$i \rightarrow j$ 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．

2.2 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 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 1 遷移リスト

LA 及び RA は Arc-Standard[23] 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]

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．オラクル関数 O は、t 時刻目の状態 s_t を引数にとり、正解の遷移を返す関数である（表 2）．

	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 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．$j_{chunktag}$ はチャンクの開始文字（B）と内部文字（I）を表す（2.4 参照）．

表 3 の遷移例にあるように、本手法は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、実テキストの逐次型解析を可能としている．本稿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、単語単位の解析にもそのまま適用できる（4.1 参照）．

中国語の同時解析との大きな違いは SH と CH にある．従来手法が SH 時、つまり、単語の一文字目に対して品詞を予測する一方で、本手法はチャンカー内に文字列を一度保持し、その文字列に対して形態素解析を適用する．SH 時の品詞予測は、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ると予想され、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 [18]．それに対して、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、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（2.4 参照）．

2.3 遷移分類器

分類器には、Chen & Manning[7] と同様のニューラルネットワークを採用した（図 1）．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 を使用する．

$$\mathbf{f}_t = f(s_t) \quad (1)$$

$$\mathbf{h}_1 = \mathrm{ReLU}(\mathbf{W_1}\mathbf{f}_t + \mathbf{e_1}) \quad (2)$$

$$\mathbf{h}_2 = \mathbf{W_2}\mathbf{h}_1 + \mathbf{e_2} \quad (3)$$

$$a_t = \mathrm{softmax}(\mathbf{h}_2) \quad (4)$$

[*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 [24] 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．

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 “太郎が花子を褒めた.”に対する正解遷移系列

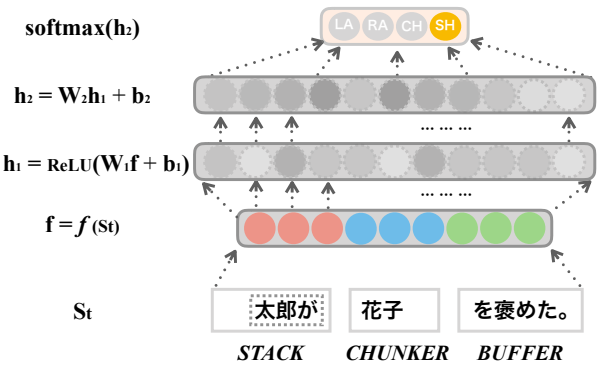

図 1 遷移分類器で用いるニューラルネットワーク

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 4 基本特徴量

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 5 追加特徴量

2.3.1 基本特徴量

表 4 が素性関数 f が抽出する基本特徴量であり、$\sigma[i]$、$\beta[i]$、$\delta[i]$ によって、各データ構造の i 番目の要素を参照する．Chen & Manning[7] 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（$\mathbf{q}$、2.4 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．

2.3.2 追加特徴量

逐次的な形態素解析を行うにあたり、Kurita ら [18] 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 5）．

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．

$$\mathbf{u_{i:j}} = \text{average}(\beta[i], ..., \beta[j]) \tag{5}$$

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]

$$\mathbf{r}_{0:k} = \text{dict}(\beta[0], ..., \beta[k]) \quad (k = 1, 2, 3) \tag{6}$$

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 $\mathbf{f}_{last}$（2.4.1 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．

$$\mathbf{v} = \mathbf{W}_{flast}\mathbf{f}_{last} + \mathbf{W}_{btop}\beta[0] + \mathbf{e}_v \tag{7}$$

[*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．

2.4 チャンカーによる形態素解析とチャンクの構築

形態素解析は、表 6 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 2 に示す．

2.4.1 BiLSTM による形態素解析

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、$\mathbf{b}_{last}$ 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．

$$\mathbf{f}_i, \mathbf{b}_i = \mathrm{BiLSTM}(n_i) \quad (8)$$

$$\mathbf{q} = \mathbf{V}[\mathbf{f}_{last}; \mathbf{b}_{last}] + \mathbf{e}_q \quad (9)$$

$$\mathbf{d_i} = \tanh(\mathbf{V}[\mathbf{f}_i; \mathbf{b}_i] + \mathbf{e}_d) \quad (10)$$

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．

$$\mathbf{o}_{i[p|d|c|s]} = \mathbf{W}_{[p|d|c|s]}\mathbf{d}_i + \mathbf{e}_{[p|d|c|s]} \quad (11)$$

$$t_{i[p|d|c|s]} = \mathrm{softmax}(\mathbf{o}_{i[p|d|c|s]}) \quad (12)$$

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．

$$t_{i[ws]} = \mathrm{softmax}(\mathbf{U}[\mathbf{d}_i; \mathbf{o}_{i[p]}; \mathbf{o}_{i[d]}; \mathbf{o}_{i[c]}; \mathbf{o}_{i[s]}] + \mathbf{e}_{ws}) \quad (13)$$

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（2.4.2 参照）．

2.4.2 チャンクベクターの構築

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ユニット数 × 5 つ（学習済みエンベディングを使わない場合は 4 つ）のエンベディグを保持しており、これらを形態素情報ごとにそれぞれ平均して、当該チャンクの形態素情報 $\mathbf{m}_{[w|t|p|d|c|s]}$ とする．例えば、n 個の単語に分割された場合の $\mathbf{m}_{[w]}$ は $\mathrm{average}(\mathbf{p}_{i[w]}, \mathbf{p}_{j[w]}, ..., \mathbf{p}_{n[w]})$ によって与えられる．

$$\mathbf{m}_{[w|t|p|d|c|s]} = \mathrm{average}(\mathbf{p}_{i[w|t|p|d|c|s]}) \quad (14)$$

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

$$\mathbf{k_j} = \mathrm{ReLU}(\mathbf{Q}[\mathbf{m}_{[w]}; \mathbf{m}_{[t]}; \mathbf{m}_{[p]}; \mathbf{m}_{[d]}; \mathbf{m}_{[c]}; \mathbf{m}_{[s]}; \mathbf{q}] + \mathbf{e}_k) \quad (15)$$

$\mathbf{k_j}$ 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．

3. 実験

3.1 データと比較手法

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）[15] と Kyoto University and NTT Blog コーパス（以下、KNB コーパス）[32] 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．

比較手法としては、CaboCha[31][*3]と J.DepP[27][*4]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．

3.2 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[20][*5]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（2.4.2 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した [6] [*6]．遷移分類器の多層パーセプトロン（2.3 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[16] 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 / 秒となった（精度は 2 から 3pt 低下）．

3.3 評価

評価では、J.DepP 付属の評価スクリプトを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・

*3 https://taku910.github.io/cabocha/
*4 http://www.tkl.iis.u-tokyo.ac.jp/ ynaga/jdepp/
*5 https://github.com/clab/dynet
*6 https://github.com/facebookresearch/fastText

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 6 形態素解析の教師データ

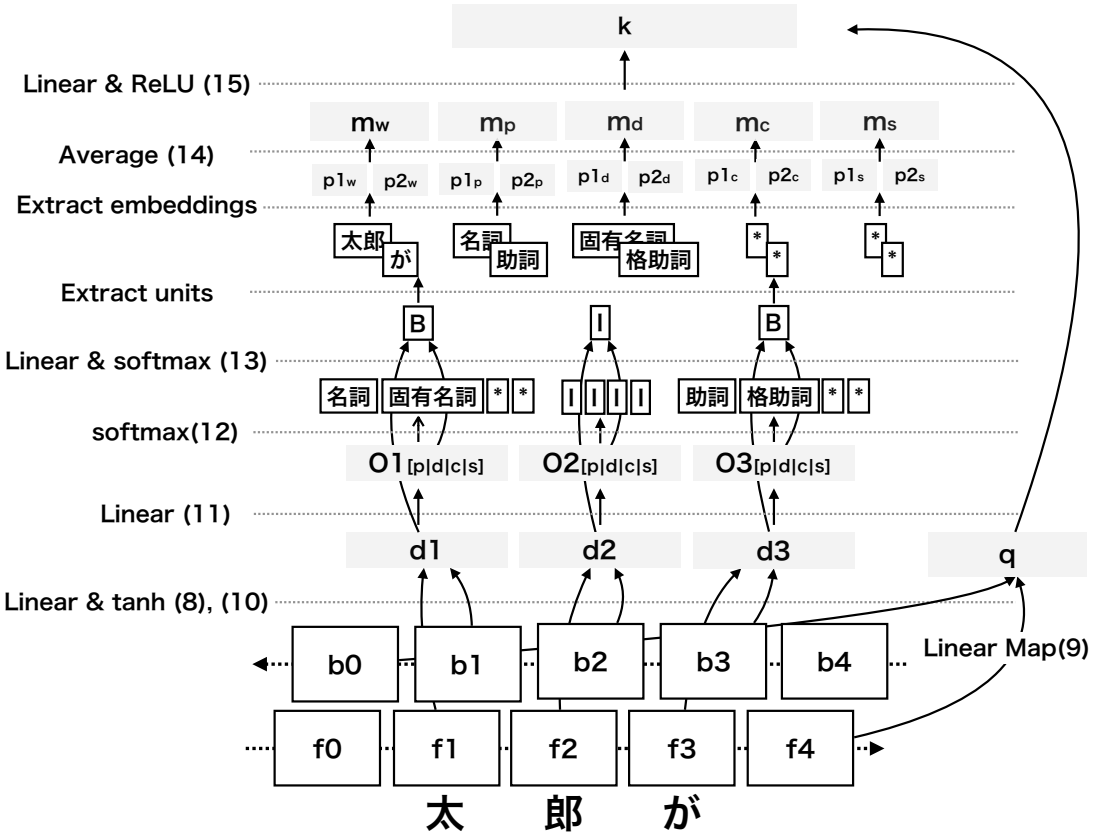

図 2 BiLSTM 形態素解析器（学習済みエンベディングなし）

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[28] 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．

3.4 結果

表 7 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 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．

表 8 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．追加素性の影響は少し異なっており、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．

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 7 結果（京大コーパス）

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 8 結果（KNB コーパス）

3.5 考察

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．

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．

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 [13]、より現実的な環境を考慮した改良や検証が望まれる．

4. 全体考察

4.1 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 3 の *Taro praises Hanako.* という英語の文に対する遷移は、表 9 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．

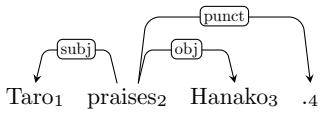

図 3 英語例文

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

表 9 図 3 の文に対する遷移系列

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 [14], [25], [30] や係り受け解析 [1], [10] 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．

4.2 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 [5], [11]．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 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 $s_t = ([\text{ 優しい}, \text{花子を }], [], [\text{ 褒}, \text{め}, \text{た}, \text{．}], A)$ で曖昧性が生じる．本稿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．

5. 結論

本稿では、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．従来の中国語の逐次解析器における品詞推定の脆弱性を補い、強力な形態素解析器を内蔵して、様々な言語に対してそのまま適用できる仕様となっている．精度としても非逐次型の解析器に劣らない性能を有しており、また、解析速度も十分に実用的な水準である．しかし、単語分割や口語解析、多言語拡張などにおいては工夫の余地が残されている．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、逐次解析に関する議論を継続する必要があるだろう．

参考文献

[1] Abney, S. P.: Parsing by Chunks, *Principle-Based Parsing: Computation and Psycholinguistics* (Berwick, R. C., Abney, S. P. and Tenny, C., eds.), Studies in Linguistics and Philosophy, Vol. 44, Kluwer (1991).
[2] Baumann, T.: Incremental Spoken Dialogue Processing: Architecture and Lower-level Components, PhD Thesis (2013).
[3] Baumann, T. and Schlangen, D.: Interactional Adequacy as a Factor in the Perception of Synthesized Speech, *Proceedings of SSW*, Barcelona, Spain (2013).
[4] Beuck, N., Köhn, A. and Menzel, W.: Incremental parsing and the evaluation of partial dependency analyses (2011).
[5] Björkelund, A. and Nivre, J.: Non-Deterministic Oracles for Unrestricted Non-Projective Transition-Based Dependency Parsing, *Proceedings of the 14th International Conference on Parsing Technologies*, Bilbao, Spain, Association for Computational Linguistics, pp. 76–86 (online), available from ⟨http://www.aclweb.org/anthology/W15-2210⟩ (2015).
[6] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T.: Enriching Word Vectors with Subword Information, *arXiv preprint arXiv:1607.04606* (2016).
[7] Chen, D. and Manning, C.: A Fast and Accurate Dependency Parser using Neural Networks, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Association for Computational Linguistics, pp. 740–750 (online), available from ⟨http://www.aclweb.org/anthology/D14-1082⟩ (2014).
[8] Demberg, V. and Keller, F.: A psycholinguistically motivated version of TAG, *Proceedings of the 9th international workshop on tree adjoining grammars and related formalisms*, pp. 25–32 (2008).

[9] Demberg-Winterfors, V.: Broad-coverage model of prediction in human sentence processing, PhD Thesis (2010).
[10] Goldberg, Y. and Elhadad, M.: An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing, *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, California, Association for Computational Linguistics, pp. 742–750 (online), available from ⟨http://www.aclweb.org/anthology/N10-1115⟩ (2010).
[11] Goldberg, Y. and Nivre, J.: A Dynamic Oracle for Arc-Eager Dependency Parsing, *Proceedings of COLING 2012*, Mumbai, India, The COLING 2012 Organizing Committee, pp. 959–976 (online), available from ⟨http://www.aclweb.org/anthology/C12-1059⟩ (2012).
[12] Hatori, J., Matsuzaki, T., Miyao, Y. and Tsujii, J.: Incremental Joint Approach to Word Segmentation, POS Tagging, and Dependency Parsing in Chinese, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea, Association for Computational Linguistics, pp. 1045–1053 (online), available from ⟨http://www.aclweb.org/anthology/P12-1110⟩ (2012).
[13] Honnibal, M. and Johnson, M.: Joint Incremental Disfluency Detection and Dependency Parsing, *Transactions of the Association for Computational Linguistics*, Vol. 2, pp. 131–142 (online), available from ⟨https://transacl.org/ojs/index.php/tacl/article/view/234⟩ (2014).
[14] Ishiwatari, S., Yao, J., Liu, S., Li, M., Zhou, M., Yoshinaga, N., Kitsuregawa, M. and Jia, W.: Chunk-based Decoder for Neural Machine Translation, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, Association for Computational Linguistics, pp. 1901–1912 (online), available from ⟨http://aclweb.org/anthology/P17-1174⟩ (2017).
[15] Kawahara, D., Kurohashi, S. and Hasida, K.: Construction of a Japanese Relevance-tagged Corpus., *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Canary Islands - Spain, European Language Resources Association (ELRA), (online), available from ⟨http://www.lrec-conf.org/proceedings/lrec2002/pdf/302.pdf⟩ (2002).
[16] Kingma, D. and Ba, J.: Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).
[17] Köhn, A. and Menzel, W.: Incremental Predictive Parsing with TurboParser, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland, Association for Computational Linguistics, pp. 803–808 (online), available from ⟨http://www.aclweb.org/anthology/P14-2130⟩ (2014).
[18] Kurita, S., Kawahara, D. and Kurohashi, S.: Neural Joint Model for Transition-based Chinese Syntactic Analysis, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, Association for Computational Linguistics, pp. 1204–1214 (online), available from ⟨http://aclweb.org/anthology/P17-1111⟩ (2017).
[19] McGraw, I. and Gruenstein, A.: Estimating Word-Stability During Incremental Speech Recognition., *the INTERSPEECH 2012 Conference* (2012).
[20] Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., Duh, K., Faruqui, M., Gan, C., Garrette, D., Ji, Y., Kong, L., Kuncoro, A., Kumar, G., Malaviya, C., Michel, P., Oda, Y., Richardson, M., Saphra, N., Swayamdipta, S. and Yin, P.: DyNet: The Dynamic Neural Network Toolkit, *arXiv preprint arXiv:1701.03980* (2017).
[21] Nivre, J.: An efficient algorithm for projective dependency parsing, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 149–160 (2003).
[22] Nivre, J.: Incrementality in Deterministic Dependency Parsing, *Incremental Parsing: Bringing Engineering and Cognition Together (ACL Workshop)*, pp. 50–57 (online), available from ⟨http://www.aclweb.org/anthology/W/W04/W04-0308.pdf⟩ (2004).
[23] Nivre, J.: Algorithms for Deterministic Incremental Dependency Parsing, *Computational Linguistics*, Vol. 34, No. 4, pp. 513–554 (online), available from ⟨http://www.aclweb.org/anthology/J08-4003⟩ (2008).
[24] Nivre, J., Kuhlmann, M. and Hall, J.: An Improved Oracle for Dependency Parsing with Online Reordering, *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, Paris, France, Association for Computational Linguistics, pp. 73–76 (online), available from ⟨http://www.aclweb.org/anthology/W09-3811⟩ (2009).
[25] Watanabe, T., Sumita, E. and Okuno, H. G.: Chunk-Based Statistical Translation, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, Association for Computational Linguistics, pp. 303–310 (online), DOI: 10.3115/1075096.1075135 (2003).
[26] Yamada, H. and Matsumoto, Y.: Statistical dependency analysis with support vector machines, *Proceedings of IWPT*, Vol. 3 (2003).
[27] Yoshinaga, N. and Kitsuregawa, M.: A Self-adaptive Classifier for Efficient Text-stream Processing., *COLING*, ACL, pp. 1091–1102 (2014).
[28] Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinkova, S., Hajic jr., J., Hlavacova, J., Kettnerová, V., Uresova, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., dePaiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, c., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonca, G., Lando, T., Nitisaroj, R. and Li, J.: CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Vancouver, Canada, Association for Computational Linguistics, pp. 1–19 (online), available from ⟨http://www.aclweb.org/anthology/K/K17/K17-3001.pdf⟩ (2017).
[29] Zhang, M., Zhang, Y., Che, W. and Liu, T.: Character-Level Chinese Dependency Parsing, *Proceedings of the 52nd Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), Baltimore, Maryland, Association for Computational Linguistics, pp. 1326–1336 (online), available from ⟨http://www.aclweb.org/anthology/P/P14/P14-1125⟩ (2014).

[30] Zhou, H., Tu, Z., Huang, S., Liu, X., Li, H. and Chen, J.: Chunk-Based Bi-Scale Decoder for Neural Machine Translation, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada, Association for Computational Linguistics, pp. 580–586 (online), available from ⟨http://aclweb.org/anthology/P17-2092⟩ (2017).

[31] 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).

[32] 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，Vol. 18, No. 2, pp. 175–201 (2011).