依存構造解析のための内容語と機能語の多言語可逆変換

小比田涼介　　能地宏　　松本裕治
奈良先端科学技術大学院大学情報科学研究科
{kohita.ryosuke.kj9, noji, matsu}@is.naist.jp

1 はじめに

依存構造木のアノテーションにはいくつかの種類があり、図 1 に見られるような前置詞句のヘッド選択がその代表である。これまで様々なアノテーションスキーマが提案されてきたが [1, 2, 4]、近年 Universal Dependencies (UD) [6] が注目を集め、多言語間での統一的な標準となりつつある。

しかし解析難易度に関しては、UD のスキーマは必ずしも最適とは言えない。UD は統一的に**内容語ヘッド** (内容語が機能語の親、図 1 上部) でアノテートされているが、一方で、**機能語ヘッド** (機能語が内容語の親、図 1 下部) の方が解析しやすいことが知られている [3, 11]。

本稿では、訓練時に内容語ヘッドを解析しやすい機能語ヘッドに変換し、そして、機能語ヘッドで学習した解析器の出力を内容語ヘッドに再変換する。この可逆変換によって、UD の解析精度が向上することを示す。

構造の可逆変換によって依存構造解析の精度が向上することは、Nilsson ら [8] によって報告されている。しかし、彼らは Prague Dependency Treebank を使用しており、変換対象も並列句や助動詞句であった。Silveira & Mannning [12] は UD のスキーマでアノテーションされた英語コーパスを用いて機能語周辺の可逆変換を検証したが、コピュラ周りを再変換する際に生じるエラー伝搬によって、解析精度が下がることを報告している。Rosa [10] は、接置詞において機能語ヘッドの利点を報告しているものの、使用したツリーバンクは HamleDT (Prague Dependency ベース) [13] であり、変換対象も接置詞のみに限られていて、機能語全体の効果とは言い難い。このように UD における内容語・機能語ヘッドの変換については、その効果が未だ明らかではない。

これらの先行研究を踏まえ、本稿の主な貢献は以下のようにまとめられる。まず、提案する可逆変換が UD の多くの言語において解析精度を向上させた。これは、これまで得られなかった結果であり、要因としては、

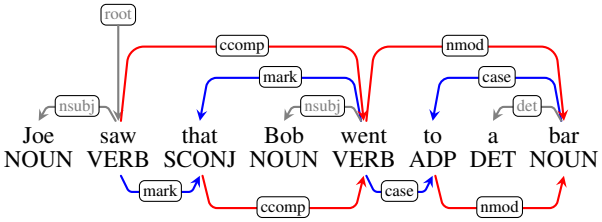

図 1: 内容語ヘッド (上)、及び、機能語ヘッド (下)

POS	Label	Example
ADP	case	... *a post* **about** *fault* ...
	dep	(ja) *Taro ni* **ha** ...
	mark	... *opinions* **on** *how it* ...
SCONJ	mark	*I think* **that** ...
ADV	mark	... *feet* **when** *you* ...
PART	case	*Elena* **'s** *motor cycle* ...
	mark	... *Sharon* **to** *make* ...

表 1: 変換対象

我々の可逆変換が多様な言語現象をカバーしつつも、対象をシンプルな構造に制限することで再変換時のエラーを抑えたという点が挙げられる。また、最新の解析器を使用して上記の結果を得た点も主な貢献の一つである。先行研究が使用している MSTParser だけではなく、最新の解析器においても可逆変換の有効性が確認できたことは、このアプローチが依存構造解析においてアルゴリズムや素性設計と同様に重要であることを示唆している。最後に、経験的にのみ知られていた機能語ヘッドの解析性について、ヘッド語彙のエントロピーの減少という観点から詳細な分析を行い、理論的・定量的な説明を試みた。

2 提案手法：多言語可逆変換

まずは表記について定義する。文中の i 番目の単語 w_i に対して、p_i はその品詞タグ、h_i はヘッドのインデックスを指す。l_i は w_i の依存関係ラベル、$left_i$ ($right_i$) は左 (右) の子のインデックスの集合である。図 1 上部の木を例に挙げると、w_5 = *went*、 p_5 = VERB、 h_5 = 2、 l_5 = ccomp、 $left_5$ = [3, 4] となる。

内容語ヘッドから機能語ヘッドへの変換アルゴリズムは、UD の元々の木を入力として、その h_i を修正す

Algorithm 1 内容語ヘッド ⇒ 機能語ヘッド変換

入力: 係り受け木 y と変換対象の集合 T
出力: CONV($root(y)$) による変換後の y

```
1:  procedure CONV(i)
2:      for j in left_i do
3:          CONV(j)
4:      CHANGEDEP(SEARCH(left_i), i)
5:      for j in right_i do
6:          CONV(j)
7:      CHANGEDEP(SEARCH(reverse(right_i)), i)
8:  procedure SEARCH(children)
9:      for j in children do
10:         if (p_j, l_j) ∈ T then          ▷ T:変換対象
11:             return j                    ▷ 最端のみを変換
12: procedure CHANGEDEP(j, i)
13:     if l_i ≠ root then                  ▷ root を飛ばす
14:         h_j = h_i, h_i = j
```

ることで機能語ヘッドの木に変換する。表 1 が変換対象である。アルゴリズム 1 が擬似コードで、T が変換対象の集合、root (y) は木 y のルートのインデックスを返す関数である。

このアルゴリズムはルートを初期入力として、依存構造木を後順で走査する。CONV は親 (i) を引数として、その子供 (j) を探索する再帰関数である。(p_j, l_j) のペアが変換対象であるような子 w_j (図 1 *that*, *to*) を発見した際に、ヘッドの修正 (図 1 *went* から *that* への mark のアークなど) が行われ、$h_j = h_i$、$h_i = j$ という形でヘッドを付け替える (CHANGEDEP)。

この時変更するのはヘッド (h_i) だけであり、ラベル (l_i) はそのままである。例外処理として、root ラベルが中間ノードに現れるのを避けるため、ルートの子供は変換しない。また、複数の変換対象が子供の中に見つかった場合、最も端の子だけを対象とし、その他は処理しない。機能語ヘッドから内容語ヘッドへの再変換アルゴリズムも同様の手順で行うが、子ではなく親の品詞タグとラベルを確認する。[1]

変換対象は同じものを全言語に適用した。しかし表 1 の対象は、英語と日本語における再変換の精度を考慮して選定したため、これが必ずしも最適解とは言えない。より良い対象の模索も今後の課題である。

提案手法の一つの欠点として non-projective 構造の増加が挙げられ、訓練データにおいて平均 10%増加していた。しかしながら、UD は元来 non-projective 構造を含んでおり、解析器はこれらを扱えるようになる必要があるため、このことによって可逆変換の有用性が著しく損なわれることはないだろう。

[1]UD の元の木において、mwe の単語が機能語をヘッドに持つ場合がある。複単語表現の構成要素の関係を反転させないために、再変換時に子が mwe ラベルを持つ場合、CHANGEDEP 処理を飛ばす。

3 実験

3.1 実験設定

各ツリーバンクと解析器において 2 つのモデルを学習する。一つは UD 元来の木 (UD、内容語ヘッド) で、もう一つが変換木 (CONV、機能語ヘッド) である。CONV の出力を UD のスキーマに再変換することで、UD のテストデータで平等に両モデルを評価できる。

解析器には non-projective 構造を扱える MSTParser [7] (MST、second-order) と RBGParser [5] (RBG、third-order) を採用した。

ツリーバンクには、サイズと多様性を考慮して 19 の言語を UD から選択した [2]。訓練データにおいて、平均 6.3% (2.3%-15.6%) の単語を変換し、再変換の失敗率は高々0.01% (平均で 0.002%) であった。また、ゴールドの品詞タグを使用し、句読点は評価から除いている。

3.2 結果

Attachment scores 表 2 が示すように、特に labeled attachment score (LAS) において顕著な精度向上が見られた。MST では 11 の言語で 1.0 pt 以上向上しており、RBG においても 10 の言語で 0.5 pt 以上向上した。一方、unlabeled attachment score (UAS) における差異は小さかった。これは本稿での変換が、単なるヘッドの推定以上に、正解ラベルの推定について効果があることを示唆している。

しかし、ヒンディー語の LAS は RBG の結果において低下している。一つの可能性として、ヒンディー語のように元々高いスコア (91.74) が出るような場合、変換によって解析しにくくなるのかもしれない。

とはいえ、このような全体的な精度の向上は、先行研究において報告されていない。一つの要因として、ここでの変換が単純な構造と処理に制限されていたことが考えられる。今回変換しなかったコピュラや助動詞周辺は、構造上複雑な変更を伴うため、再変換時にエラー伝搬が起きやすい。我々の変換も一部エラー伝搬の影響を受けているが (後述)、その程度は小さく、解析のしやすさと変換の複雑さについて良いバランスが取れたのだろう。

全体傾向は両解析器で同様であるので、以降の解析では RBG の結果に焦点を当てる。

Label F1-scores さて変換によってどういった誤りが減少したのだろうか。それを検証するために、ここではラベル毎の F 値を比較していく。表 3 は高頻度ラベルの結果を要約したものであり、dobj (+.008)、nmod

[2]アラビア語とフランス語は、RBG での訓練中にエラーが生じたため除外した。

L.	UAS				LAS				CNC	
	MST		RBG		MST		RBG		RBG	
	UD	CONV	UD	CONV	UD	CONV	UD	CONV	UD	CONV
bg	88.39	**88.85**	90.33	**90.73**	81.63	**82.63**	84.85	**85.64**	80.74	**81.92**
cs	86.65	**87.20**	91.40	**91.67**	79.85	**80.65**	87.25	87.22	85.23	85.21
da	82.03	**83.46**	86.08	**86.49**	76.81	**78.52**	82.13	**82.63**	78.42	**79.57**
de	84.69	84.65	**87.19**	86.68	75.47	**77.69**	79.39	**80.63**	72.03	**74.10**
en	85.97	**86.30**	89.69	89.65	80.67	**81.89**	86.32	**86.50**	82.30	**82.83**
es	85.98	**86.47**	89.02	**89.21**	80.13	**81.96**	84.98	**85.75**	77.33	**79.00**
et	**81.04**	80.81	87.67	87.60	71.28	**71.55**	83.84	**84.07**	82.58	**82.99**
fa	83.26	**84.20**	82.83	**84.35**	78.43	**80.02**	78.64	**80.53**	74.53	**77.47**
fi	**76.76**	76.42	85.57	**85.80**	68.24	**68.55**	81.69	**82.46**	80.46	**81.22**
hi	89.80	**92.13**	**95.10**	94.99	84.11	**87.19**	**91.74**	90.76	**87.96**	87.22
hu	79.31	**79.94**	**84.53**	84.15	66.47	**67.26**	79.53	**79.94**	77.19	**78.06**
it	88.82	**89.47**	92.14	**92.83**	83.90	**85.93**	89.22	**90.16**	83.31	**85.27**
ja	87.67	**90.19**	91.58	**92.24**	79.96	**85.41**	87.70	87.62	81.09	81.14
no	89.14	**89.44**	91.57	91.57	84.06	**85.23**	88.31	88.32	84.81	**85.14**
pl	**88.10**	87.71	92.25	**92.47**	80.20	**80.73**	87.51	**87.70**	85.08	**85.64**
pt	**85.82**	85.34	90.51	**91.04**	80.16	**80.53**	86.79	**87.47**	80.30	**81.90**
ru	81.46	**81.91**	86.76	**87.13**	74.79	**75.86**	83.15	**83.92**	81.01	**82.04**
tr	**79.02**	78.90	85.10	85.13	62.56	**62.66**	75.33	**75.57**	73.70	**74.19**
zh	**79.28**	79.05	**85.75**	85.46	73.44	**74.70**	80.91	**81.67**	79.43	**80.45**
Avg.	84.38	**84.87**	88.69	**88.90**	76.96	**78.37**	84.17	**84.67**	80.40	**81.33**

表 2: UAS 及び LAS の比較。CNC については本文を参照。太字は 0.1 ポイント以上の差を意図している。

Type	Label	Ratio	UD	CONV
core	advmod	4.9%	0.792	0.791
	amod	6.3%	0.924	0.926
	conj	4.4%	0.666	**0.678**
	dobj	5.7%	0.819	**0.827**
	nmod	14.6%	0.765	**0.790**
	nsubj	7.3%	0.802	**0.818**
func	case	11.4%	**0.955**	0.945
	cc	3.3%	0.795	**0.800**
	det	6.6%	0.949	0.950
	mark	2.9%	**0.874**	0.849

表 3: RBG における UD と CONV それぞれの高頻度ラベル F1 値とテストデータにおける割合

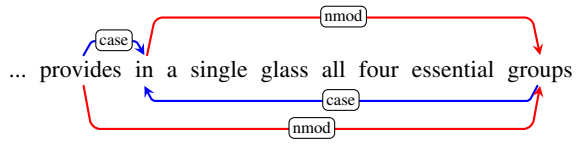

図 2: CONV モデルの初期出力 (上) と case に関する後変換時のエラー伝搬 (下)

(+.025)、nsubj (+.016) というように意味的に重要な核ラベル [3] のスコアが上昇している。図 1 を見ると、UD の木 (図上) において 2 つの内容語 (*went* と *bar*) は nmod のアークによって直接繋がれているが、CONV の木 (図下) においては、それらは機能語 (*to*) を介している。内容語間の繋がりはヘッド語彙がスパースになりやすく、推定も難しくなりがちである。その結果として後者の構造の方が解析しやすいという説明が考えられ、この仮説については定量的な検証を後に行う。

一方で、mark や case といった機能ラベルの F 値は若干悪化しており、エラー分析の結果、再変換時のエラー伝搬が原因であることがわかった。これらのラベルは、CONV モデルの出力において正しく推定されていたとしても、対になるラベル (例 nmod) を誤っていた場合、再変換の過程でそのエラーが伝搬するのである。図 2 を参照しながら、より具体的な説明を行う。CONV モデルの最初の出力 (上部) において、*in* へのアーク (case) は正しいが、*groups* へのアーク (nmod) は *in* から伸びており、これは誤りである (正しくは *provides* から)。この場合、再変換時に、case のアークは *groups* から *in* という誤った形へ変換され (正しくは *glass* から)、それがスコアを低下させる要因となっている。機能ラベルの誤りのうち、CONV モデルのみで観察されたものの多くはこのパターンであり、機能ラベルの推定自体はいずれのアノテーションにおいても容易であることが伺える。しかし、提案する可逆変換は再変換を正しく行うために、2 つのアークを正確に解析する必要があり、その点で難しい局面がある。

この付随的な複雑性は欠陥のように見えるが、全体としてのスコアは上昇している。これは両方のアークを正しく推定できる場合の方が大半であることを示唆しており、提案手法は機能ラベルのスコアを僅かに落としてしまうが、それ以上に、**一般に推定の難しい核ラベル**の正確な解析に寄与していると捉えられる。

CNC　これまでの直感をより明確にするために、CNC と呼ばれる別の尺度を用いた結果についても報告する。これは Nivre [9] が UD の評価を目的として提案したものであり、機能的アーク [4] を除いて LAS を計算する。表 2 の右端の列が結果であり、LAS よりも明確にスコアの向上が伺え、平均で 0.9 ポイント上がってい

[3] 本稿では、表 3 の ”core ” という行にあるラベルを**核ラベル**と呼び、”func ”という行にあるラベルを**機能ラベル**と呼ぶ

[4] 以下のラベルを持つアーク: aux, auxpass, case, cc, cop, det, mark, neg.

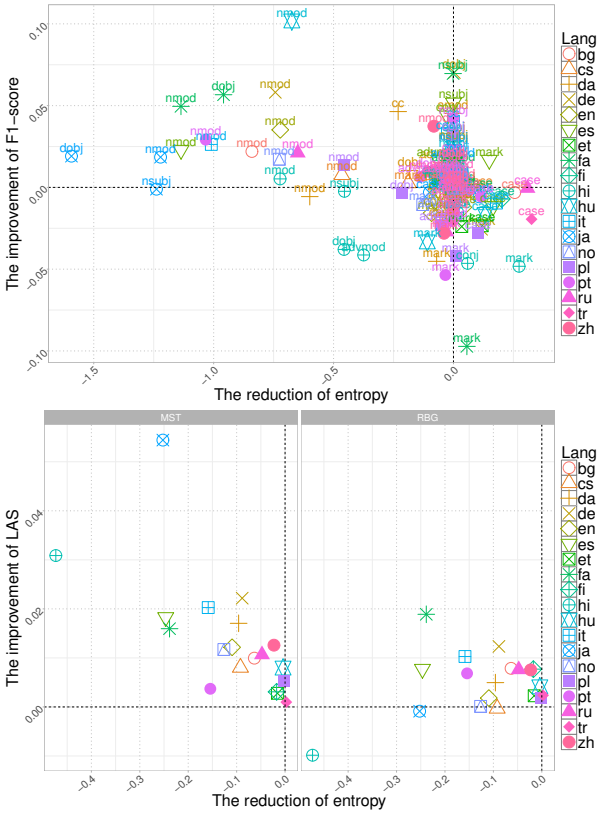

図 3: エントロピーの低下とラベル別 F1 値 (上、RBG) 及び LAS (下、MST 及び RBG)

る。この結果も、可逆変換が機能的アークのスコアを僅かに落とすものの、意味的に重要なアークの解析に成功することを示唆している。

Head word vocabulary entropy　最後になぜ核ラベルのスコアが向上したのかについて詳細な解析を行う。上述した通り、これは機能語の仲介による内容語間の語彙的なスパース性の緩和と関連していると考えられる。そのため、ヘッド語彙のエントロピーの減少という観点から定量的に検証していく。Schwartz [11] は、エントロピーと依存構造解析の難易度に関して言及しているが定量的には未検証であった。

まず、訓練データにおいて、ラベル l を持つ h から w への各アーク $h \overset{l}{\curvearrowright} w$ について、ペア $((p, l, w), h)$ を取り出す (p は w の品詞タグ)。次に、頻度 5 未満のタプル (p, l, w) を除き、ヘッド単語のエントロピー $H_l(h)$ を $P(h|p, l, w)$ の条件付き確率から計算する。この処理を UD と CONV の両方に行った後、$H_l^{orig}(h) - H_l^{conv}(h)$ で各ラベルにおけるエントロピーの差を計算する。

図 3 上図では、nmod の多くが左上付近に位置しており、言語横断的にエントロピーの減少と精度向上の関連性が伺える。また、付近には日本語とペルシャ語の dobj もあり、両言語とも目的格の表現においてケースを明示するため、機能語の仲介が生じる言語である。

さらに、各言語の一単語辺りのエントロピーの減少と LAS の相関も検証した。図 3 下図において負相関が観察でき、これもまた変換による全体としてのエントロピーの減少と精度向上の関連性を示唆している。特に MST で強い負の相関があった ($r = -.75; p < .01$)。RBG では、外れ値に見えるヒンディー語を除外すると、有意ではないが弱い負相関があった ($r = -.35; p = .14$)。

4　結論

機能語周辺の可逆変換がヘッド単語の語彙を制限し、解析精度の向上へと繋がることを示した。この手法はモジュール的であり、前後処理段階としていかなる解析器とも組み合わせて用いることができる。本稿はデータの形を工夫することで、計算量や複雑さを増やすこと無く、精度改善が可能であることを示すものであり、新たな方向性の一つとして重要となるだろう。今後の依存構造解析における中間表現を用いた解析精度の向上に関する研究の発展に繋がることを期待する。

参考文献

[1] Marie-Catherine de Marneffe and Christopher D Manning. The stanford typed dependencies representation. In *COLING*, pp. 1–8, 2008.

[2] Jan Hajic, Barbora Vidová-Hladká, and Petr Pajas. The prague dependency treebank: Annotation structure and support. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pp. 105–114, 2001.

[3] Angelina Ivanova, Stephan Oepen, and Lilja Øvrelid. Survey on parsing three dependency representations for english. In *ACL*, pp. 31–37, 2013.

[4] Richard Johansson and Pierre Nugues. Extended constituent-to-dependency conversion for english. In *Nordic Conference of Computational Linguistics*, pp. 105–112, 2007.

[5] Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Low-rank tensors for scoring dependency structures. In *ACL*, pp. 1381–1391, 2014.

[6] Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. Universal dependency annotation for multilingual parsing. In *ACL*, pp. 92–97. Citeseer, 2013.

[7] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *EMNLP*, pp. 523–530, 2005.

[8] Jens Nilsson, Joakim Nivre, and Johan Hall. Graph transformations in data-driven dependency parsing. In *ACL*, pp. 257–264, 2006.

[9] Joakim Nivre. Universal depndency evaluation. In *http://stp.lingfil.uu.se/ nivre/docs/udeval-cl.pdf*, 2016.

[10] Rudolf Rosa. Multi-source cross-lingual delexicalized parser transfer: Prague or stanford? In *Depling*, pp. 281–290, 2015.

[11] Roy Schwartz, Omri Abend, and Ari Rappoport. Learnability-based syntactic annotation design. In *COLING*, pp. 2405–2422, 2012.

[12] Natalia Silveira and Christopher Manning. Does universal dependencies need a parsing representation? an investigation of english. In *Depling*, pp. 310–319, 2015.

[13] Daniel Zeman, David Mareek, Martin Popel, Loganathan Ramasamy, Jan tpnek, Zdenk abokrtsk, and Jan Haji. Hamledt: To parse or not to parse? In *LREC*, pp. 2735–2741, 2012.