2L-5

動的オラクルを用いた日本語の係り受け解析

前田 光貴 †　　猪口 明博 ‡
関西学院大学 理工学部 情報科学科

1　はじめに

本稿では，動的オラクルを用いた日本語の係り受け解析について報告する．Shift Reduce 法に基づいた従来の係り受け解析器では，初期状態から正解の最終状態に至る遷移経路は唯一であり，その経路から生成された訓練事例を用いて係り受け解析器の学習がなされる．そのため，学習済み係り受け解析器がこの経路から外れたとき，その動作が不安定になる課題がある．本稿では係り受け解析器が正解の経路を外れたときにも，極力正解に近い係り受け構造を出力するために，動的オラクルを用いる．動的オラクルを統合した係り受け解析器を実装し，その評価実験について報告する．

2　係り受け解析

係り受け解析は，自然言語処理の基本技術の 1 つとして認識されており，従来から多くの研究が行われてきた [3]．本節では，係り受け解析に用いられるモデルを説明し，状態遷移系に基づく解析手法を紹介する．

定義 2.1 n 文節からなる文 $x = \langle w_1, w_2, \cdots, w_n \rangle$ の係り受け構造は，有向グラフ $g = (V, E)$ で表される．ここで，$V = \{1, 2, \cdots, n\}$，及び $E \subseteq V \times V$ である．■

本稿では，日本語の文節係り受け解析を対象とする [1]．よって，係り受け構造において各文節はその後方に係り先を持つ．また，係り受け構造は頂点 n を根とする木で表される整形式であるとする．さらに，以下の定義を満たす projective な係り受け構造を対象とする．

定義 2.2 係り受け構造 $g = (V, A)$ が，全ての辺 $(i, j) \in A$ と頂点 $k \in V$ $(i < k < j)$ に対して，k から j に至る有向路が存在するときに限り，g を *projective* と呼ぶ．■

例 2.3 $x =$ ⟨ 私は, 彼女の, 真心に, 感動した. ⟩ という文の係り受け構造は図 *1* の有向グラフで表される．この図に示されるように，*projective* な係り受け構造の辺は交差なしに図示することができる．

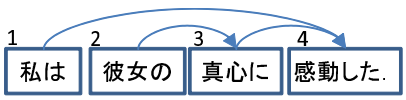

図 1: 係り受け構造の例

Dependency Analysis for Japanese Sentences using a Dynamic Oracle
†Mitsuki Maeda　‡Akihiro INOKUCHI
†‡Department of Informatics, School of Science and Technology, Kwansei Gakuin University

Parse$(x, \boldsymbol{w})$
1) $c \leftarrow c_s(x)$
2) while $c \notin C_F$
3) $t_p \leftarrow \arg\max_{t \in \{Arc, Shift\}} \boldsymbol{w}^T \phi(c, t)$
4) if $t_p \neq o(c, g_{gold})$
5) $\boldsymbol{w} \leftarrow \boldsymbol{w} + \phi(c, o(c, g_{gold})) - \phi(c, t_p)$
6) $c \leftarrow [o(c, g_{gold})](c)$
7) return $\boldsymbol{w}$

図 2: 静的オラクルによる学習アルゴリズム

Transitions	
Arc	$(N, A) \Rightarrow (N, A \cup \{(i, j)\})$
	where $\{i, j\} = roots((N, A))$
Shift	$(N, A) \Rightarrow (N \cup \{\lvert N \rvert + 1\}, A)$
Preconditions	
Arc	c is not a tree, but a forest.
Shift	$j \neq n$, where $\{i, j\} = roots((N, A))$

図 3: 遷移集合とその前提条件

次に，状態遷移系に基づく係り受け解析器を定義する [2]．ここでの入力は $x = \langle w_1, w_2, \cdots, w_n \rangle$ であり，出力は $g = (V, E)$ である．

定義 2.4 状態遷移系に基づく係り受け解析器は $S = (C, T, c_s, C_F)$ で表される．ここで

- C は状態の集合であり，各状態 $c = (N, A) \in C$ は頂点集合 $N \subseteq \{1, 2, \cdots, n\}$ と辺集合 $A \subseteq N \times N$，
- T は遷移集合であり，$t \in T$ は $t : C \rightarrow C$ を満たす部分関数，
- c_s は初期関数であり，文 $x = \langle w_1, w_2, \cdots, w_n \rangle$ に対して $c_s(x) = (\{1\}, \emptyset)$，
- $C_F \subseteq C$ は最終状態の集合であり，$c_F \in C_F$ は n を根とする木である．■

これ以降，状態 c の頂点集合を N_c，辺集合 A_c と表す．

状態遷移系に基づく係り受け解析器の学習アルゴリズムを図 2 に示す．ここで $\phi(c, t)$ は状態 c から t で遷移する際の特徴ベクトルで，$\boldsymbol{w}$ は重みベクトルである．また，g_{glod} は文 x に対する正解の係り受け構造である．さらに，o はある状態 c から次の状態へ遷移する関数 t を一意に定める関数で，静的オラクルと呼ばれる．具体的には，o は図 3 に示す Arc か Shift の遷移のいずれかを選択する関数であり，roots は森 (N, A) の根のうち，頂点 ID が大きい 2 つの頂点対 $\{i, j\}$ $(i < j)$ を返す関数である．関数 o により Arc が選択された場合，グラフ (N, A) に辺 (i, j) を加え，c から $t(c)$ に遷移する．一方，Shift が選択された場合，(N, A) に含まれない頂点の 1 つをグラフ (N, A) に加え，c から $t(c)$ に遷移する．

Parse$(x, \boldsymbol{w})$
1) $c \leftarrow c_s(x)$
2) while $c \notin C_F$
3) $t_p \leftarrow \arg\max_{t \in \{Arc, Shift\}} \boldsymbol{w}^T \phi(c, t)$
4) $t_{zero} \leftarrow t$ s.t. $cost(t; c, g_{gold}) = 0$
5) if $t_p \neq t_{zero}$
6) $\boldsymbol{w} \leftarrow \boldsymbol{w} + \phi(c, t_{zero}) - \phi(c, t_p)$
7) $c \leftarrow$ [Choose_Next (I, t_p, t_{zero})](c)
8) return $\boldsymbol{w}$

Choose_Next(I, t_p, t_{zero})
1) if $I > k$ and rand() $> p$ or $t_p = t_{zero}$
2) return t_p
3) return t_{zero}

図 4: 動的オラクルによる学習アルゴリズム

従来の係り受け解析器では，初期状態から正解の最終状態に至る遷移経路は唯一であり，その経路から生成された訓練事例を用いて係り受け解析器の学習がなされる．そのため，学習済み係り受け解析器がこの経路から外れたとき，その動作が不安定になる課題がある．本稿では係り受け解析器が正解の経路を外れたときにも，極力正解に近い係り受け構造を出力するために，動的オラクル [4] を用いる．

3 動的オラクル

正解の係り受け構造が $g_{gold} = (V, A_{gold})$ である文 x を解析し，状態 $c = (N, A)$ に至ったとする．このとき，状態 c のロスを $L(c, g_{gold}) = \left|A_{gold} \setminus A\right|$ と定義する．c から到達しうる最終状態を $c_m : c \rightsquigarrow c_m$ で表すと，c から $t(c)$ に遷移することで増加するロスは次式で表される．

$$cost(t; c, g_{gold}) = \left[\min_{c_m : t(c) \rightsquigarrow c_m} L(c_m, g_{gold})\right] - \left[\min_{c_m : c \rightsquigarrow c_m} L(c_m, g_{gold})\right]$$

このコスト関数の戻り値が 0 となる状態へ遷移するように重みベクトルを学習することで，先に述べた課題を解決することができる [4]．このコスト関数により遷移を決めることを動的オラクルと呼ぶ．

図 4 に動的オラクルを用いた学習アルゴリズムを示す．4 行目において，コストが 0 となる遷移 t_{zero} を得る．またロスが増える遷移を意図的に起こすために関数 Choose_Next を用いる．これにより上記で述べた課題に対処することができる．関数 Choose_Next の中の関数 rand は 0 から 1 の一様乱数を返す関数，I は学習の繰り返し回数である．次節の評価実験では $k = 2$ と $p = 0.1$ をデフォルト値として用いる．

次に，日本語の係り受け解析におけるコストを具体的に定義する．状態 c において，遷移 Arc のコストを

$$cost(arc; c, g_{gold}) = \left|\{(i, k) \in A_{gold} \mid roots(c) = (i, j), j < k\}\right|$$

とする．遷移 Arc を行うことで i が状態 (c, A) において根ではなくなるため，i の正しい係り先を同定できない．そのため，A_{gold} のうち係り元が i である辺の数をコストとしている．ただし，文末を除く各文節の係り先は必ず 1 つだけなので，Arc のコストの最大は 1 である．一方，状態 c において，遷移 Shift のコストを

$$cost(shift; c, g_{gold}) = \left|\{(k, j) \in A_{gold} \mid roots(c) = (i, j), k < i, \nexists h \text{ s.t. } (k, h) \in A_c\}\right|$$

とする．Shift の遷移を行うことで，それ以降 j は係り先となることはない．つまり，j を正しい係り先とする係り元に対して，正しい係り先を同定できない．そのため A_{gold} のうち，係り先が j である辺の数をコストとする．ただし，k は状態 c において根であるものとする．

表 1: 係り受け正答率の結果

記事の日付	動的オラクル	静的オラクル
1 月 9 日	88.83%	87.63%
1 月 10 日	88.68%	88.16%
1 月 11 日	88.66%	87.35%
1 月 12 日	88.36%	87.54%
1 月 13 日	88.84%	88.19%
1 月 14 日	88.78%	87.77%
1 月 15 日	88.73%	87.73%
1 月 16 日	88.67%	87.70%
1 月 17 日	89.13%	88.23%

4 評価実験

提案法を実装し，京大コーパス Version 4.0 を使用して評価実験を行った．具体的には，1 月 1 日から 1 月 8 日までの記事を訓練データとし，1 月 9 日から 1 月 17 日までを評価データとした．図 4 の k や p を変化させた実験も行ったが，紙面の都合上，表 1 において静的オラクルと動的オラクルを用いた場合の係り受け正答率のみを示す．9 日分のどの日の記事においても動的オラクルを用いた場合の係り受け正答率が高いことがわかる．改善度は小さいものもマクネマー検定を行ったところ，2 つの係り受け正答率の差は有意（有意水準 $\alpha = 0.01$）であった．

5 まとめ

本稿では係り受け解析器が正解の経路を外れたときにも，極力正解に近い係り受け構造を出力するために，動的オラクルを用いる．動的オラクルを統合した係り受け解析器を報告した．

参考文献

[1] T. Kudo and Y. Matsumoto. Japanese Dependency Analysis using Cascaded Chunking *Proc. of Conf. on Computational Natural Language Learning*, 63–69, 2002.

[2] J. Nivre. Algorithms for Deterministic Incremental Dependency Parsing. *Comp. Linguistics*, 34 (4), 513–553, 2008.

[3] S. Kubler, et. al. Dependency Parsing. *Morgan and Claypool Publishers*, 2009.

[4] Y. Goldberg and J. Nivre. A Dynamic Oracle for Arc-Eager Dependency Parsing. *Proc. of Int. Conf. on Comp. Linguistics*, 959-976, 2012.