段階的な部分木間の構造判定に基づく決定的係り受け解析

北川 浩太郎 †　田中 久美子 †
† 東京大学大学院情報理工学系研究科
kitagawa@cl.ci.i.u-tokyo.ac.jp, kumiko@i.u-tokyo.ac.jp

1 はじめに

統計的に単語間の係り関係を同定する手法は盛んに研究されており、膨大な構文木を生成せずに効率的な解析を実現するために主に 2 つのアプローチが用いられている。単語間の係り関係に対してスコアを近似的に定め、その合計を文全体で最大化する手法 [3, 8] と、文中の係り関係を反復的に決定していく手法 [2, 5, 7] の 2 つであり、精度と解析時間がトレードオフの関係にある中でより良いモデルであることが求められる。

最適化を行う手法においては、係り関係のスコアを決定する際に他の係り関係との独立性を仮定していた。この問題を解決するため、head(主辞) と dependent(従属語) の係り関係に、dependent の兄弟要素や左右の子要素を考慮するモデルが提案され [8, 1] 精度は大きく改善された。しかしその時間計算量は語数 n の文に対して $O(n^4)$ であり、自然言語処理の基盤技術である構文解析においてはより高速な手法が求められる。

決定的な手法は計算量は $O(n) \sim O(n^2)$ と高速であることに加え、過去に決定した係り関係を素性として利用できるといった長所がある。しかし係り関係は限定的な範囲で決定されるため、より適切な係り関係を参照する前に係り関係が決定してしまうといった問題がある。日本語係り受け解析においては複数の候補から最適な head を選ぶトーナメントモデルが提案された [4] が、これは日本語が必ず後方に係ることを利用し、後方から解析することで既に解析済みの語の中から head を決定するモデルであった。そのため、head 候補が左右に存在し、解析する順番の指針が得られない projective な言語一般への応用は難しかった。

本稿では、決定的な手法である Nivre のモデル [5] において、2 つの単語間の関係のみからでは判定困難な問題が解析に含まれていることに着目し、それを解決する手法として部分木間の構造判定に基づく解析を提案する。提案手法では、入力文を前方から参照して徐々に係り受け木を作成する手順で解析を行い、部分木間の最適な係り関係を決定することでより多くの head 候補を考慮した解析を実現した。

提案手法の時間計算量は $O(n^2)$ ではあるものの、Penn Treebank を用いた英語の解析実験では既存手法より解析時間ではそれほど劣らず、改善された精度のうち文正解率においては、McDonald らによる兄弟要素を考慮した最適化手法を上回った。

2 係り受け解析

入力された文 $x = (w_1, \ldots, w_n)$ に対して $V = \{0, 1, \ldots, n\}$(0 は ROOT) をノード、$A \subseteq V \times V$ をエッジとする係り受けグラフ $G = (V, A)$ を作成する。このとき $(i, j) \in A$ は w_i が head で w_j が dependent の関係であることを示し、解析対象として扱う文は以下の well-formed[5, 6] の条件を満たす文とする。

1. ROOT 以外の語は head を高々ひとつ持つ
2. 係り受けグラフは連結である
3. 係り受けグラフは非循環である
4. 係り関係は交差しない (projective である)

ここで、1 ～ 3 は G が根付き木であることを意味し、projective な根付き木となる G が生成される。

3 Nivre のモデル (従来手法)

Nivre のモデル [5] は入力文の語を先頭から順にスタックに移しながら解析を行う手法である。スタックのトップにある語 t と入力語のうち先頭にある語 n を参照し、以下の 4 つの解析アクションから適切なものを分類器によって選択する。距離の近い単語対から解析されていき、入力が空になったら処理を終了する。

Left-Arc (n, t) の係り関係を作成し、t をスタックから降ろす
Right-Arc (t, n) を作成し、n をスタックに積む
Reduce t をスタックから降ろす
Shift n をスタックに積む

分類器の素性は主に t や n の品詞と表層形を用いる。反復的に解析アクションを選択することで $O(n)$ で projective な係り受け木を作成できるものの、以下の点で問題があると考えられる。

解析アクションの不統一

4 つの解析アクションが存在するが、t の head が決まっていない場合は **Reduce** できず、head が決まっている場合は **Left-Arc** は選択できないため、実際には異なる三択問題を行うことになっている。

構造判定と解析アクションの不整合

Reduce と **Shift** の選択は、n との係り関係のある語を t より左から探すか、あるいは右側から探すかを判断することに他ならない。これは t の祖先に当たる語全てを素性とすれば判断可能ではある。しかし t と n の周辺のみを素性とする分類器が判定できるのは基本的に単語間の構造であり、解析アクションとの整合性が低い。

係り関係を決定する際の参照する範囲の狭さ

2 単語間の係り関係を決める際には、係り関係を持ち得る他の単語は考慮されない。そのため head としてより適切な候補があっても、先に係り関係が決定してしまう。これは一度に参照される構造が 2 つの単語の関係である以上は避けられない問題である。

4 部分木間の構造判定に基づくモデル

既存手法は単語間関係という局所的な範囲の構造判定に基づく最適な解析アクションの決定であり、その改良としては参照する範囲を拡大することが考えられる。その際には構造を適切に分類できるような範囲を定めることが望ましい。

そこで、本稿では段階的な部分木間の構造判定に基づくモデルを提案する。係り受けグラフを段階的に生成する時、その過程でのグラフは最終的に得られる係り受け木の部分木から構成される。このとき 2 つの隣接する部分木 t, n の根にあたる語を $root_t, root_n$ と表すと、t, n 間に存在する係り受け構造は、$root_t$ の head が n の中に存在する場合、$root_n$ の head が t の中に存在する場合、t と n に含まれる語の間には係り関係が存在しない場合のいずれかである。well-formed な文では全ての語は高々ひとつしか head を持たないため、部分木の中で最も head らしい語 (**最適 head 候補**) を事前に選択することで、t と n の間の構造決定は 3 値分類問題に帰着させることができる。

提案モデルは Nivre のモデルと同様に入力文をスタックに移しながら解析を行うが、スタックに積まれるのは語ではなく部分木であり、入力文は 1 語からなる部分木の列と見なす。そしてスタックのトップにある部分木 t と入力の先頭にある部分木 n を参照しながら以下の 2 つの処理を繰り返す。

1. 部分木間の最適 head 候補を選択する
2. 3 つの解析アクションを選択する

この 2 つの処理内容を順に述べる。

4.1 最適 head 候補の選択

$root_n$ の**最適 head 候補** $mphc_t(n)$ を t 内の語から選択する。選択には飯田ら [9] が提案し岩立ら [4] が日本語の係り受け解析に用いたトーナメントモデルを用いる。図 1 のように 2 つの head 候補を繰り返し比較することで、最終的に勝ち上がった語を最適 head 候補と決定する。ただし、トーナメントで選ばれる $mphc_t(n)$ はあくまで t の中で最もその head らしい語であり、head かどうかの判定は解析アクションまでは行われない。比較操作のみで最適 head 候補を決定する利点は、dependent からの相対的な距離を考慮して head 候補を選択できることであり、このトーナメントを行うためには、一回の対戦で $root_n$ と 2 つの head 候補から左右どちらがより head らしいかを決定する学習器があればよい。

$root_n$ の head となれる head 候補は t 内の最も右にある語からその head を辿りながら $root_t$ に至るまでの語であり、他の語が $root_n$ の head とすると projective な係り受け木の条件は満たされない。

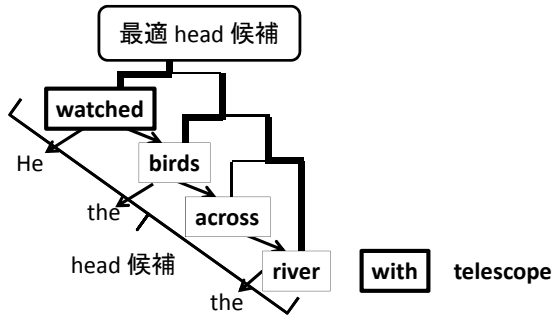

図 1: 最適係り先候補選択の例

4.2 解析アクションの決定

最適 head 候補を選択した後に解析アクションを決定する。部分木 t, n 間の構造は $root_n$ が $root_t$ の head である場合、$mphc_t(n)$ が $root_n$ の head である場合、係り関係が存在しない場合の 3 通りであり、それぞれに対して解析アクションを定める。

Left-Arc $(root_n, root_t)$ を作成し、t はスタックから降ろして n の一部とする

Right-Arc $(mphc_t(n), root_n)$ を作成し、n は t の一部とする

Shift n をスタックに積む

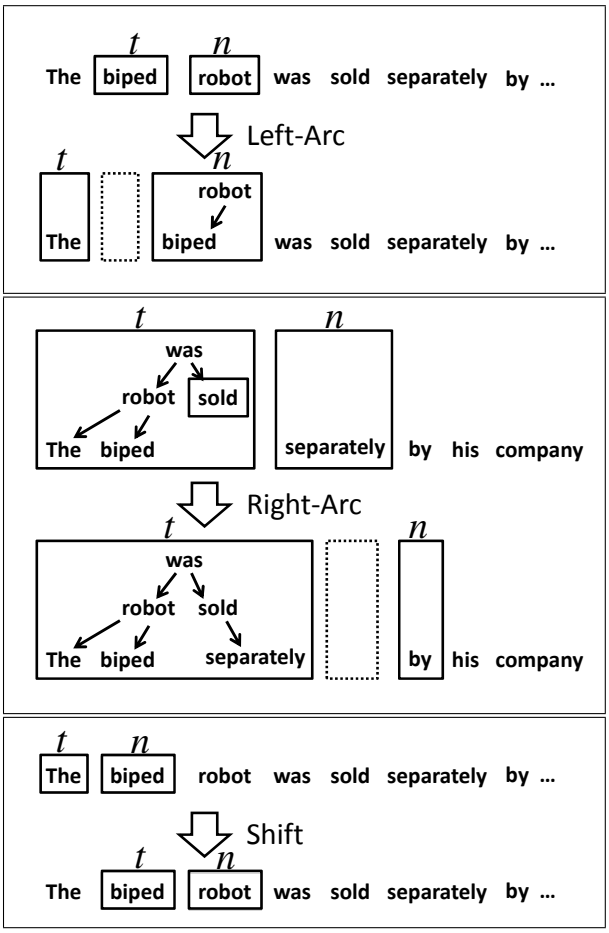

図 2: 解析アクションの例

図 2 に、枠に囲まれた 2 つの部分木 t, n に対して解析アクションを行う例を示す。**Left-Arc** の例では $root_t$ である “biped”が $root_n$ である “robot”の dependent となり、スタックのトップは “The”からなる部分木に移っている。**Right-Arc** の場合は $root_n$ の “separately”が事前に決定された最適 head 候補 “sold”の dependent となり、入力の先頭 n は “by”からなる部分木に移っている。2 つの部分木の間に係り関係がない場合は **Shift** され、例では n の位置にあった “biped”がスタックに積まれて t となり、入力の先頭は “robot”となる。

従来手法では 2 つの単語間に係り関係がなかった場合には、**Reduce** するべきか **Shift** するべきかという複数の可能性が存在したが、提案手法では左側を参照せずに必ず **Shift** できることを以下に示す。t と n の間に係り関係が存在しない場合、スタック内の語には新たな係り関係は作成されないため、$root_t$ の head は n より右側に存在するか、あるいは ROOT である。すると、仮に $root_n$ が t より左の語を head とした場合、2 つの係り関係は交差し projectivity に反することになる。

このように $root_n$ の head となり得る候補が n より左では t の中にしか存在しないため、**Right-Arc** の際には projectivity を満たす全ての head 候補を考慮した係り関係が得られる。

一方、**Left-Arc** で $root_t$ の head となるのは $root_n$ に固定されている。これは、$root_t$ 以外の候補は図 2 の例文における “The”に対する “biped”のように、一度係り関係がない **Shift** と判定された語であることによる。

このように提案アルゴリズムでは、部分木間に存在しうる 3 種の構造と解析アクションを対応させることで、**解析アクションとの整合性を高める**と同時に**解析アクションの不統一**を解消し、**projectivity を満たす全ての左側 head 候補を考慮**した解析が可能となると考えられる。

5 評価

評価は以下の 3 つのステップで行った。

1. 最適 head 候補選択のための比較操作の訓練事例を生成・学習
2. 比較操作の学習器を用い、解析アクション決定の訓練事例を生成・学習
3. 解析

訓練事例の生成は、解析アルゴリズムと同じ流れで学習データを正しく解析する中で作成する。比較操作の訓練事例の場合は、$mphc_t(n)$ を t 内から選択するとき、$root_n$ の真の head である h が t 内に存在する場合のみ作成できる。このとき、他の head 候補全てと h を比較して h が勝ったという事例を作成し、学習を行う。

$root_n$ の真の head が t 内に存在しない場合の $mphc_t(n)$ はトーナメントで得られるようになるので、次に解析アクション決定の訓練事例を作成する。解析アルゴリズムに従い、$root_t$ の真の head が $root_n$ であったなら **Left-Arc**、$root_n$ の真の head が $mphc_t(n)$ であったなら **Right-Arc**、それ以外は **Shift** と判断されたという学習事例を生成する。

5.1 実験設定

実験には Penn Treebank WSJ を、山田 [7] の主辞規則を用いて句構造から係り受け構造に変換したコーパスを利用した。学習データとして section2 から 21(39,832 文) を用い、section23(2,416 文) をテストデータとした。学習には SVM^{light}[1] とその Java native interface である JNI_SVM-light-6.01[2] を使用し、head 候補の比較は 2 次の多項式カーネル、解析アクションの決定は 3 次の多項式カーネルを用いた。解析は Core2Duo(2.53GHz) の Windows Vista 上で行った。

[1] http://svmlight.joachims.org/
[2] http://www.mpi-inf.mpg.de/ mtb/

5.2 素性

最適 head 候補選択の学習に用いた素性は、左 head 候補、右 head 候補、$root_n$ に加えて n の後続 3 語と、それぞれの語に係り関係が存在する場合はその親ノード、左にある子ノード、右にある子ノードの品詞と単語をそれぞれ素性とした。

解析アクションを選択するための学習器の素性は $root_t, mphc_t(n), root_n$ と n の後ろに続く 3 語、さらにそれらと係り関係にある語も用いた。ここで n の後続 3 語を用いるのは、例えば図 1 のように前置詞の head を決定する場合などには、後ろに続く語の情報が必要となるためである。

5.3 評価方法

単語の head(ROOT も含める) を正しく推定できた割合を示す単語正解率と、全ての head を正しく推定できた文の割合である 文正解率を用いて解析結果を評価する。ただしテストデータに含まれる 56,685 語のうち、句読点と括弧を除いた 49,892 語を評価対象とする。

5.4 結果

SVM を用いて実装した Nivre モデルの追試との比較を表 1 に示した。提案手法の時間計算量は $O(n^2)$ ではあるが、図 3 に示した文中の単語数 n ごとの平均解析時間においては $O(n)$ の既存手法から大きく劣ることはなく、精度において改善が見られた。表 2 は同じデータセットを用いた既存研究からの報告との比較である。決定的な手法のみならず、文正解率については 2 次の係り関係を考慮した McDonald らの最適化手法 [8] を上回った。

表 1: 実験結果

	単語正解率	文正解率	解析時間
Nivre モデル	91.1%	39.7%	2.46[sec/文]
提案モデル	91.3%	42.3%	2.85[sec/文]

6 まとめ

本稿では projective な言語に対する決定的な係り受け解析手法として、部分木間の構造を段階的に決定するモデルを提案した。提案モデルは部分木間の構造が 3 通りに分類できることを利用し、解析アクションをそれぞれに対応させることで、反復的な部分技間の構造判定に問題を帰着させた。語の左側については projectivity を満たす全ての head 候補を考慮することが可能であり、トーナメントモデルを用いることによって、dependent

表 2: 既存研究の比較

	単語正解率	文正解率	手法
M&P(2006)	91.5%	42.1%	2od Eisner+MIRA
Y&M(2003)	90.4%	38.4%	決定的手法+SVM
Nivre(2003)	87.1%	30.4%	決定的手法+MBL
提案モデル	91.3%	42.3%	

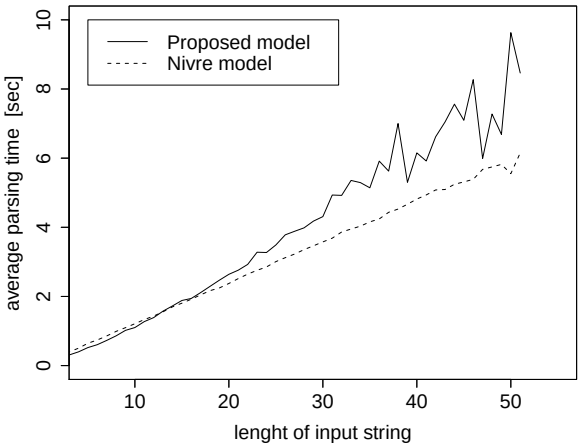

図 3: 平均解析時間

からの相対的な距離を考慮した head 候補の選択を行う。提案モデルは Nivre モデルの拡張にも相当し、計算量は $O(n^2)$ であるものの、英語の解析実験においては $O(n)$ の既存手法から解析速度を大きく落とすことなく精度を向上させた。

参考文献

[1] X. Carreras, Experiments with a higher-order projective dependency parse, *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pp.957–961, 2007.

[2] M.A. Covington, A fundamental algorithm for dependency parsing, *Proceedings of the 39th annual ACM southeast conference*, pp.95–102, 2001.

[3] J. Eisner, Three new probabilistic models for dependency parsing: An exploration, *Proceedings of the 16th International Conference on Computational Linguistics*, pp.340–345, 1996.

[4] M. Iwatate, M. Asahara, and Y. Matsumoto, Japanese dependency parsing using a tournament model, *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pp.361–368, 2008.

[5] R. McDonald and F. Pereira, Online learning of approximate dependency parsing algorithms, *Proceedings of the European Association for Computational Linguistics*, 2006.

[6] J. Nivre, An efficient algorithm for projective dependency parsing, *Proceedings of the 8th International Workshop on Parsing Technologies*, pp.149–160, 2003.

[7] J. Nivre, Algorithms for deterministic incremental dependency parsing, *Computational Linguistics*, vol.34,num.4, pp.513–553, 2008.

[8] H. Yamada and Y. Matsumoto, Statistical dependency analysis with support vector machines, *Proceedings of 8th International Workshop on Parsing Technologies*, pp.195–206, 2003.

[9] 飯田 龍, 乾 健太郎 , 高村大也, 松本 裕治, 機械学習によるゼロ代名詞同定の一方法, 情報処理学会研究報告. 自然言語処理研究会報告, 2003-NL-154, pp. 161–168, 2003.