依存構造解析におけるルール型から学習型解析器への転移

坪井 祐太　金山 博　吉川 克正　那須川 哲哉
日本アイ・ビー・エム株式会社
東京基礎研究所
{yutat, hkana, katsuy, nasukawa}@jp.ibm.com

中山 章弘　菅野 啓
日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
{nakaaki, sgnk}@jp.ibm.com

John Richardson
京都大学
大学院情報学研究科
john@nlp.ist.i.kyoto-u.ac.jp

1　はじめに

十分に整備されたルールに基づく解析手法は，機械学習に基づく解析手法と遜色のない性能を示すことがある．しかし，解析器の入力（例えば品詞タグ）への依存性やルール関係の複雑化によってルール管理が困難になり，入力の変更や解析内容の変更などに頑健ではない．一方，学習型解析手法での入力の変更は特徴量の変更で，解析内容の変更は訓練データの変更で対応可能で多くの場合は解析アルゴリズムを変更する必要がない．また，近似アルゴリズム等を適用することにより解析を高速化することも比較的容易である．

本稿では，ルール型解析器がすでにある場合に，新たに学習型解析手法を採用する状況を想定したアプローチを検討する．訓練データが少量の場合は，既存のルールやルール型解析器の出力を特徴量として参照する方法などが考えられる．一方，本研究では訓練データ作成に人手がかけない設定の下で，既存のルール型解析器を使って訓練データを生成して依存構造解析を学習する方法を採用した．本手法ではルール型解析器はブラックボックスのまま，訓練データを修正することで入出力の変更が行える．また，既存の解析器出力を人手で修正した訓練データとすることもよく行われるが，大規模に修正を行うことは困難である．

2 節で，大規模データに対応可能な遷移型依存構造解析器のオンライン学習アルゴリズムを提案し，英語依存構造解析のベンチマークデータでの性能を示す．3 節では既存のルール型解析器を使った訓練データ生成方法について述べ，4 節で英語のデータに適用した実験結果を示す．人手をかけずに大規模な訓練データを生成することでも既存のルール型解析器と同程度の性能を達成することができたことを報告する．本手法はルールを作成するために使われた文法的知識を事例に基づき転移する手法として有効であった．さらに，既存のルール型解析器は文長 T に対して計算量 $O(T^3)$ であったが，$O(T)$ の決定的な遷移型依存構造解析に置き換えることで一桁以上の高速化も実現できた．

2　遷移型依存構造解析器の学習

本節では，文献 [2] を改良した遷移型依存構造解析器のオンライン学習アルゴリズムを提案する．

遷移型依存構造解析器では中間状態（部分森）を保持しながら文の先頭のトークンから依存関係の予測と状態遷移を繰り返す．多くのアルゴリズムは文長に対して線形の計算量である．次の状態を決める行動 $a \in A$ は以前選択した行動によって決まる状態 $s \in S$ を参照して（多クラス）分類器を用いて選択する．本研究では線形分類器 $\pi(s) = \mathrm{argmax}_a \boldsymbol{w}_a^\top \boldsymbol{\phi}(s)$ を用いる．ただし，$\boldsymbol{w}_a$ は行動 a に対するパラメータベクトル，$\boldsymbol{\phi}(s)$ は s に対する特徴関数である．また，状態 s と行動 a が与えられた時の次の状態への遷移を $\tau(s,a) : S \times A \to S$ とし，次の状態は一意に求まる．

提案する遷移型依存構造解析器の学習擬似コードをアルゴリズム 1 に示す．特徴は，損失関数は a^* と現在のパラメータ $\boldsymbol{w}$ での行動 a で評価するが，状態遷移では平均化したパラメータ $\bar{\boldsymbol{w}}$ での行動 $\bar{a}$ を用いる点である．文献 [2] の学習アルゴリズムは，状態遷移するために正解行動集合 A^* の行動と現在のパラメータ $\boldsymbol{w}$ で選んだ行動を選ぶ比率の調整が必要があった．一方，提案アルゴリズムは常に $\bar{\boldsymbol{w}}$ で選んだ行動で状態遷移し，調整が必要ない．また，文献 [2] は分類器は平均化パーセプロトンだけを仮定しているが，提案法では $\gamma = 0, \eta_k = 1, \lambda = 0$ としたときパーセプロトン，$\gamma = 1, \lambda > 0$ で SVM を平均化確率的勾配法 (ASGD) で学習していることに相当する．なお，文献 [2] のオラクルのように正解行動に曖昧性がある場合は（$|A^*| > 1$），ブリッジ損失 [3] と呼ばれる凸関数差の目的関数を確率的最適化をしていることに相当する．ASGD の更新率 η_k の更新方法・高次元データでの効率的な実装については文献 [9]，平均パラメータ $\bar{\boldsymbol{w}}$ の更新方法については文献 [7] を参照して頂きたい．

続いて，Penn Treebank III（PTB） [4] をベンチマークデータとして文献 [2] と提案学習手法との比較実験の結果を示す．実験設定は文献 [8] と同じであり，

Algorithm 1 遷移型依存構造解析器の学習

入力 η_1（初期更新率）, γ（マージン）, λ (正則化)
初期化 $k = 1, \boldsymbol{w} \leftarrow \mathbf{0}, \bar{\boldsymbol{w}} \leftarrow \mathbf{0}$,
while converged **do**
  正解木付き文をランダム選択, 状態 s を初期化
  while s が終了状態でない **do**
    s での行動可能集合 A, 正解行動集合 A^*
    SVM ならば $\tilde{A} = A \setminus A^*$, それ以外は $\tilde{A} = A$
    $a^* \leftarrow \mathrm{argmax}_{a \in A^*} \boldsymbol{w}_a^\top \boldsymbol{\phi}(s)$
    $a \leftarrow \mathrm{argmax}_{a \in \tilde{A}} \boldsymbol{w}_a^\top \boldsymbol{\phi}(s)$
    $\bar{a} \leftarrow \mathrm{argmax}_{a \in A} \bar{\boldsymbol{w}}_a^\top \boldsymbol{\phi}(s)$
    if $\boldsymbol{w}_{a^*}^\top \boldsymbol{\phi}(s) - \boldsymbol{w}_a^\top \boldsymbol{\phi}(s) < \gamma$ **then**
      $\boldsymbol{w}_a \leftarrow \boldsymbol{w}_a - \eta_k \boldsymbol{\phi}(s)$
      $\boldsymbol{w}_{a^*} \leftarrow \boldsymbol{w}_{a^*} + \eta_k \boldsymbol{\phi}(s)$
    end if
    L2 正則化: $\boldsymbol{w} \leftarrow \lambda \boldsymbol{w}$
    平均 $\bar{\boldsymbol{w}}$, 更新率 η_k の更新
    状態遷移: $s \leftarrow \tau(s, \bar{a})$
    状態遷移: $k = k + 1$
  end while
end while
Return $\bar{\boldsymbol{w}}$.

文献 [10] のルールにより依存構造木に変換し，標準的な評価方法に従って精度評価した．遷移システムは Arc-eager 法 [6] を用い，正解の依存構造木から正解行動集合 A^* を求めるには文献 [2] の正解行動に曖昧性があるオラクルを使った．また，特徴テンプレートは文献 [11]（以降，Z&N）と，特徴を加えて拡張したもの（表 1，以降 Z&N +）の両方を評価した．

表 2 は，データの学習順を決める乱数のシードに異なる値を使って学習を 10 回試行した結果である．提案手法のパーセプトロンは文献 [2] と同等か優れており，正解行動と分類器の行動を選択するためのパラメータが無い点で優れた学習法であると言える．また，L2 正則化した SVM は文献 [2] の性能を有意差をもって超えた．なお，L2 正則化パラメータは 1 つの試行における学習結果を開発セットで評価し決定した．また，解析は決定的であり，ビーム探索などは用いていない．

3 ルール型解析器を用いた訓練データ生成

ルール型解析器を用いた訓練データ生成の主な課題は，a) 既存のルール型依存構造解析器と学習型依存構造解析器で前提とするトークン単位と品詞集合が異なる点と，b) 解析アルゴリズムに合った依存関係への変換である．課題 a が発生する典型的なユースケースとしては，ルール型解析器はルール型の品詞タガーを前提にしており，学習型解析器は学習型の品詞タガーを前提にする場合である．本研究では，2 つのトークン列のアラインメントとアラインメント後の依存関係の補間を行った．なお，品詞タガーも学習しなおす方法もあるがここでは品詞タガーは広く使われているため変更できないシナリオを想定する．また，課題 b に関しては遷移型依存構造解析器の解析アルゴリズムの特性に合わせて並列句の変更を行った．

最初に，学習型解析器が使う品詞タガーのトークン単位の依存関係に変換する方法を説明する．まず，編集距離が最小になるように 2 つの品詞タガーのトークン列のアラインメントを取った．図 1 は”a part-time job” が 4 トークンになるタガーと 3 トークンになるタガーのアラインメント例を示す．セルの中はトークン同士の置換コストを示す．置換コストは 2 つのトークン（文字列）を s と t としたとき次の関数 c を用いた :

$$c(s,t) = \begin{cases} \min(|s|/|t|, |t|/|s|) & \text{if } s \cap t \neq \emptyset \\ 1 & \text{otherwise,} \end{cases}$$

ただし，$|s|$ は文字列 s の長さを示し，$s \cap t$ は s と t の

Between S_0, N_0 and N_1, N_2, N_3
$S_0wpN_{123}p$; $S_0pN_{123}wp$; $N_0wpN_{123}p$; $N_0pN_{123}wp$;
S_0wpN_1p; S_0wpN_2p; S_0wpN_3p; $S_0pN_0pN_{23}p$;
Between S_1, S_2 and N_0
S_1pN_0p; S_2pN_0p; # Between S_{0h}, S_{0h2} and N_0
$S_{0h}pN_0p$; $S_{0h2}pN_0p$;

表 1: 特徴テンプレート Z&N+. 文献 [11] の特徴テンプレート に追加するテンプレート. $S_0, S_1, \ldots$: スタックの先頭からの要素; $N_0, N_1, \ldots$: キューの要素; X_{ih}: X_i の親; X_{ih2}: X_i の 2 つ上の親; X_w: X の単語; X_p: X の品詞タグ.

特徴	手法	平均	標準偏差	p 値
Z&N	文献 [2]	90.14	0.05	
Z&N	パーセプトロン	90.32	0.07	0.002
Z&N	SVM	**90.53**	0.03	0.002
Z&N+	文献 [2]	90.91	0.08	
Z&N+	パーセプトロン	90.93	0.04	0.477
Z&N+	SVM	**91.22**	0.08	0.002

表 2: PTB テストデータでのラベルなし依存関係正答率（UAS ; unlabeled attachment scores）の 10 回試行の平均. p 値は文献 [2] との差を Wilcoxon の符号順位検定した結果.

最長共通部分文字列を示す.

次に，アラインメント後に 1 対多の関係になるトークン列間に依存関係を付与する方法を説明する．アラインメントの取られたルール型解析器の部分トークン列を $S = (s_i, s_{i+1}, \ldots, s_{i+n-1})$（長さ n），学習型解析器のための品詞タガーの部分トークン列を $T = (t_j, t_{j+1}, \ldots, t_{j+m-1})$（長さ m）とする．また，ルール型解析器により予測された依存木の親トークン ID を $Y = (y_i, y_{i+1}, \ldots, y_{i+n-1})$ であらわし，学習型解析器のトークン列へ変換した親トークン ID を $Z = (y_j, y_{j+1}, \ldots, y_{j+m-1})$ とする．$n = 1, m > 1$ のとき，T の最右を主辞とし，その他は右を親とした．つまり，$j \leq k < j + m - 1$ に対して，

$$Z[k] = \begin{cases} Y[i] & \text{if } k = j + m - 1 \\ k + 1 & \text{otherwise.} \end{cases}$$

また，S の子 l にアラインメントされたトークン k があれば T の最左トークンの子供とした（$Z[k] = j$ if $Y[l] = i$）．一方，$n > 1, m = 1$ のときは，S の最右の子があるトークン（すべて葉ならば最右トークン）l を主辞と考え，l の親にアラインメントされたトークンを T の親とした．なお，本研究では多対多のアラインメント（$n > 1, m > 1$）のケースは少数のため訓練データから除外した．図 2a に変換された依存構造木を示す．なお，品詞集合の違いについては，学習型解析器が使う品詞タガーの出力を訓練データの品詞とするのが合理的である．

最後に並列句の変更について説明する．並列句は “and” などの接続詞を並列句の主辞とするのが自然であり，ルール型解析器でも採用されているとする．しかし，この構造木では依存構造解析が非常に難しくなる．そこで，本研究では CoNLL 型（文献 [1] の 3.3 節）を採用し，接続詞を並列句の主辞としている場合に接続詞の最左の子供が主辞となるように変更する（図 2b）．なお，接続詞への依存関係ラベル（”coord”），接続詞の元の子供への依存関係ラベル（”conj”）を付与しラベル付き依存構造解析をすることで並列句の範囲を同定できるようにした．本変換は非可逆変換で完全には元の並列構造を復元できないが，並列句内の一部を正解する可能性は上がるため広く使われている並列句の表現形式であり，現実の応用では有効である．

4 実験

本節では英語依存構造解析においてルール型から学習型解析器への転移を実データで検証した結果を示す.

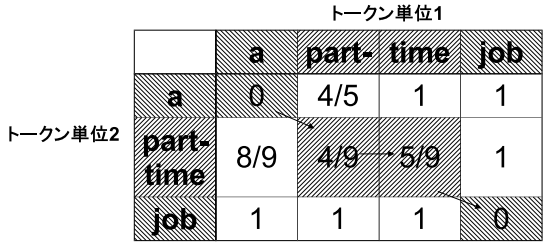
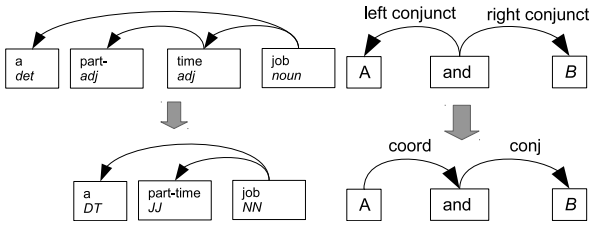


図 1: 異なるトークン列のアラインメント



(a) 異なるトークン・品詞列への変換 (b) CoNLL 型並列句へ変更

図 2: 解析出力の訓練データへの変換

実験で用いるルール型解析器はスロット文法に基づき，解析アルゴリズムは（枝狩り付き）チャート法である [5]．そのため，転移の産物として，計算量 $O(T^3)$ のチャートパーサを計算量 $O(T)$ の学習型依存構造解析に置き換えることによる依存構造解析の高速化も期待できる.

訓練データに変換する対象としては European Parliament Proceedings Parallel Corpus (Europarl) V7 の英語データ（約 5000 万語，約 200 万文）を用いた．品詞タガーを同梱するルール型解析器および学習型解析器で使う統計的品詞タガーを用いて，3 節の方法で訓練データを生成した．その際，百数文の不適切な構文木（サイクル・切断・交差）が生成されたため訓練データからは除外した．データのうち約 3600 文（約 10 万語）を開発用データとして用い，799 文（約 2 万語）に対して人手で修正を行いテスト用データとして用いた．残りを訓練データとして，学習は 2 節で提案したアルゴリズムの SVM 版を用いた．なお，ルール型・学習型の両解析アルゴリズムは非交差木のみを出力するが，テスト用データには交差木を含むため理想的な状況でも 100% の精度は望めない.

図 3 は訓練データを 80 万語から約 5000 万語まで増やしたときのテスト用データでの単語単位の精度（UAS）を示す．また，学習型解析は並列句を変更していない結果と変更した結果-学習型（並列句変更）- を示す．テスト用データでのルール型解析器の精度は 86.36% であった．学習型解析器は訓練データ量に比例して精度が上昇し，最終的には約 5000 万語の訓練

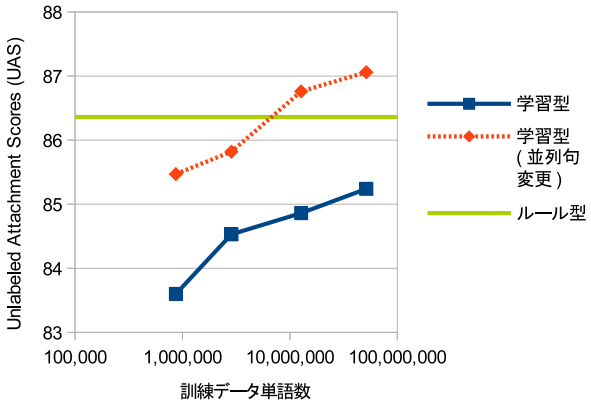


図 3: Europarl 訓練データ単語数と解析精度の関係

データで 85.24% まで上昇することが確かめられた．教師役のルール型解析器の精度を超えることは考えにくいが，この実験結果より訓練データを増やすことで既存のルール型解析器の精度に近接することが期待できる．また，予測する依存構造木が違うため直接の比較はできないが，3 節で示した並列句の変更をすることで 2 ポイント程度精度が向上することが確認された（約 5000 万語の訓練データで 87.06%）．さらに，学習型解析器はルール型解析器の 17 倍の解析速度を達成できた．

5 おわりに

既存のルール型解析器を訓練データの生成に活用し，統計的に依存構造解析を学習する方法を示した．実験により，人手をかけずに生成した大規模訓練データを学習に用いることで既存のルール型解析器と同程度の性能を達成できることが確認できた．問題設定を変えて，既存の解析器の出力の一部を人手で修正できるシナリオで有効な半教師学習手法の開発はひとつの研究の方向性である．また，3 節の手法は依存構造木コーパスをトークン単位や品詞が異なる別コーパスに変換する際にも適用可能であると考える．

参考文献

[1] Jinho D. Choi and Martha Palmer. Guidelines for the clear style constituent to dependency conversion. Technical report, University of Colorado Boulder, 2012.

[2] Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics*, pp. 959–976, 2012.

[3] Xinghua Lou and Fred A. Hamprecht. Structured learning from partial annotations. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1519–1526, New York, NY, USA, 2012.

[4] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330, 1993.

[5] Michael C. McCord. A formal system for slot grammar. Technical report, International Bussiness Machines, 2007.

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

[7] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 71–79, 2013.

[8] 坪井祐太. 模倣学習による決定的解析での誤り伝播の回避. 言語処理学会第 19 回年次大会, 2013.

[9] Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. Technical report, arXiv, 2010.

[10] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, Vol. 3, pp. 195–206, 2003.

[11] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 188–193, 2011.