# 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
IBM Research - Tokyo

中山 章弘, 菅野 啓
IBM Software Group

John Richardson
Kyoto University, Graduate School of Informatics

# 機械学習セッションですが…

# よく整備されたルール型解析器は学習型より良い性能を示すこともある

- スロット文法に基づくルール型構文解析器(ESG)
  - Charniak-Johnson parserより高精度かつ100倍高速

**Table 1** Parser comparison.

| *Parser* | *Test set A (Jeopardy!)* | *Test set B (Wikipedia)* |
|---|---|---|
| ESG (in IBM Watson) | 92.0% | 88.7% |
| ESGbase | 84.2% | 84.4% |
| Charniak parser | 83.6% | 81.1% |

Charniak-Johnson reranking parser（句構造解析器）は依存構造ネイティブ解析器より依存構造解析も高精度 (Cer+, LREC2010)

•表は以下論文より引用

•Michael C. McCord, J. William Murdock, and Branimir Boguraev. “Deep parsing in Watson”. IBM Journal of Research and Development 56(3):3 (2012)

# ルール型解析器の課題

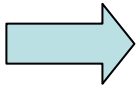
- 外部入力とルールの密結合:
  例:前処理である品詞タガーの置き換えが困難
- 解析アルゴリズムとルールの密結合:
  例:O($N^3$)のチャート法が前提
- ルール管理が属人的:
  例:ルール間依存関係が複雑になると全体像を把握することが困難

# ルール型解析器の課題と学習型解析器による置き換え

- 外部入力とルールの密結合:
  例:前処理である品詞タガーの置き換えが困難
  →[学習型] 再学習により入力の変化に対応可能
  別の品詞タガー出力を入力として再学習
- 解析アルゴリズムとルールの密結合:
  例:O($N^3$)のチャート法が前提
  → [学習型] 訓練データと解析アルゴリズムが独立
  解析アルゴリズムを変更することで高速化が可能
- ルール管理が属人的:
  例:ルール間依存関係が複雑になると全体像を把握することが困難
  → [学習型] 訓練データを確認・修正することで管理可能

# 機械学習システムへの移行方法

1. ルール型解析器は参照せずに移行
   - シンプル
   - ルール型解析器でできていた事ができなくなる可能性
2. ルール型解析器を参照して転移
   - ルールやルール型解析器出力を特徴量として入力
     - 訓練データを使ってルール間の強さを調整
     - ルールシステムへの理解、ルール型解析器のメンテナンスが必要
   - ルール型解析器の出力を学習
     - ルール型解析器の出力を模倣（訓練データ作成不要）
     - ルール型解析器は超えられない

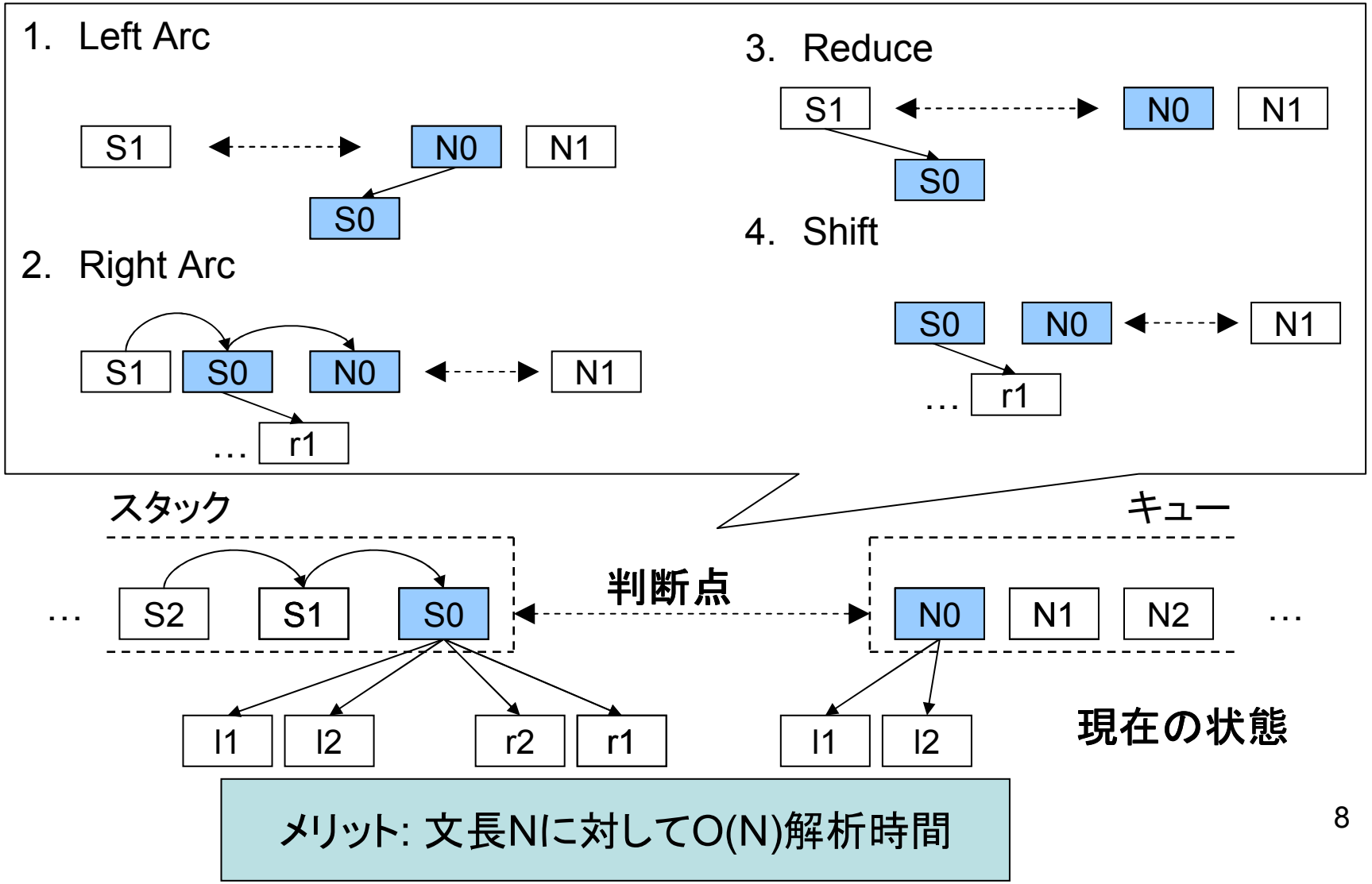


本研究のターゲット
ルール型解析器を学習型の遷移型依存構造解析器へ転移

# 発表の流れ

- 背景
- 遷移型依存構造解析器のオンライン学習アルゴリズムと特徴量の提案
  - ベンチマークデータでの比較結果
- ルール型から学習型への転移
  - ルール型解析器を用いた訓練データ生成
  - 異なるトークン列のアラインメント・依存関係付与
- 実験
  - 大規模データを使用した転移
  - 並列句の変更

# 遷移型依存構造解析器
# (Arc-eager法, Nivre 2003)



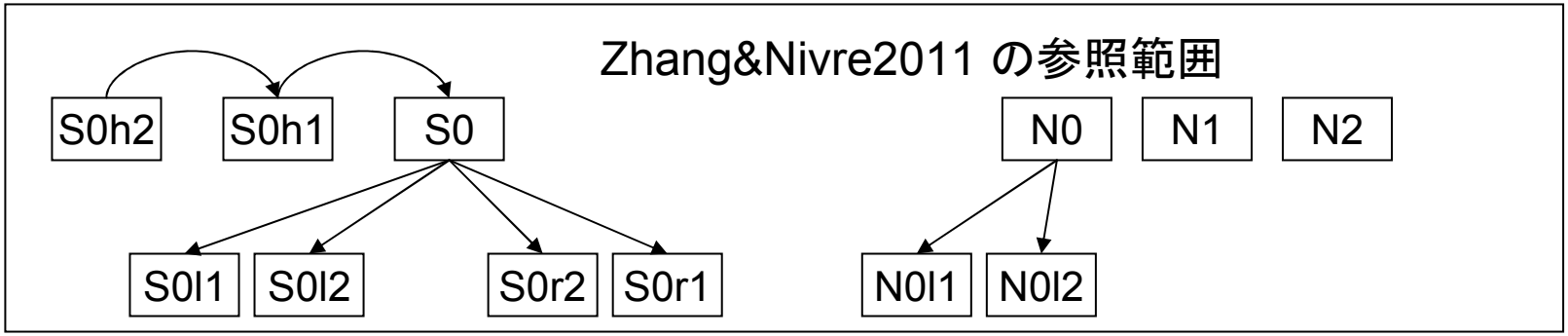
# 遷移型依存構造解析器の学習アルゴリズム
# Goldberg & Nivre 2012の改良版

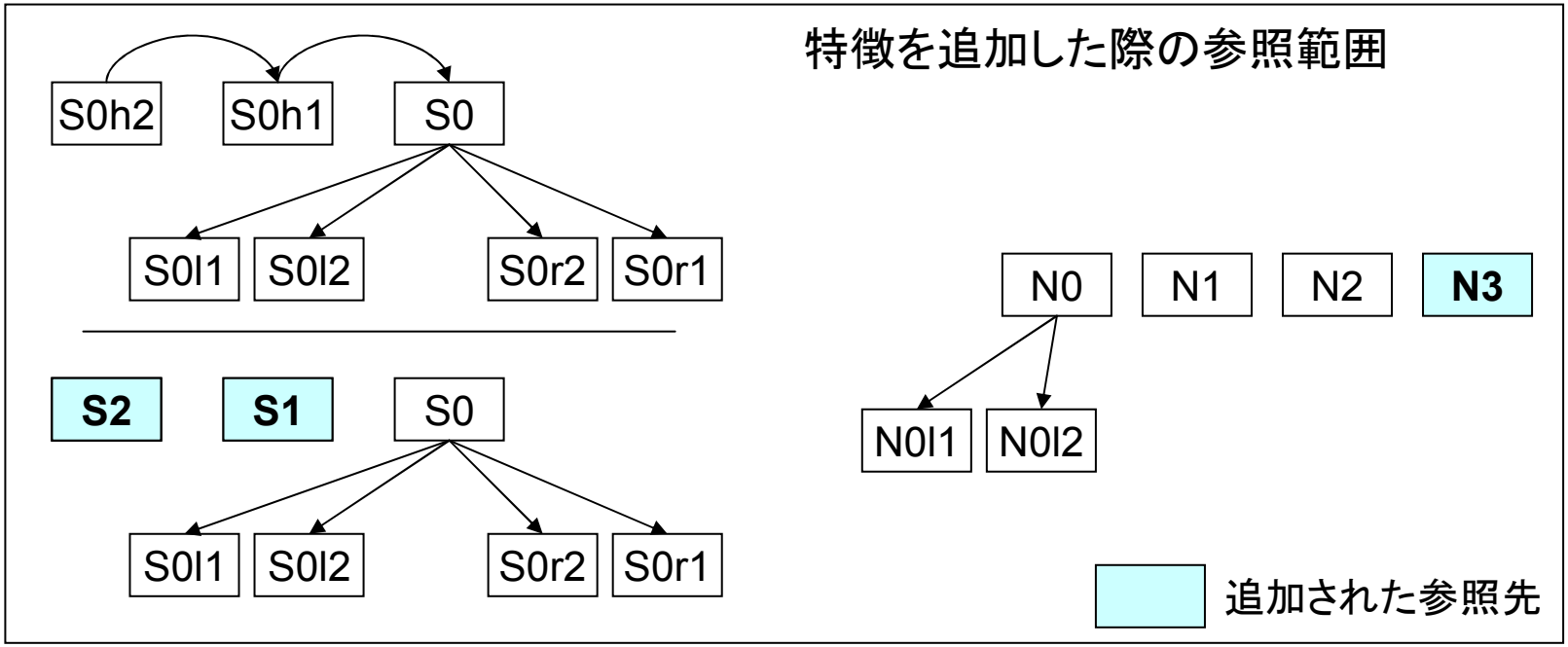
- オンライン学習アルゴリズム
  - 各行動ごとにパラメータ更新し最後に平均パラメータを出力
  - 特徴：平均パラメータで状態遷移
    - 平均パラメータで遷移にしたときの損失を最小化
  - 本研究ではbridge loss (複数の正解に対応したSVM)を使用(任意の損失関数が利用可能)

- Goldberg & Nivre 2012
  - 平均化パーセプトロン
  - 初期X回は常に正しい行動(オラクル行動)で遷移し、それ以降も確率pで正しい行動で遷移する
    - 提案手法は超パラメータ(X, p)が少ない

入力 $\eta_1$（初期更新率），$\gamma$（マージン），$\lambda$ (正則化)
初期化 $k = 1, w \leftarrow 0, \bar{w} \leftarrow 0,$
**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 $\tilde{A} = A$
　　$a^* \leftarrow \mathrm{argmax}_{a \in A^*} w_a^\top \phi(s)$
　　$a \leftarrow \mathrm{argmax}_{a \in \tilde{A}} w_a^\top \phi(s)$
　　$\bar{a} \leftarrow \mathrm{argmax}_{a \in A} \bar{w}_a^\top \phi(s)$
　　**if** $w_{a^*}^\top \phi(s) - w_a^\top \phi(s) < \gamma$ **then**
　　　$w_a \leftarrow w_a - \eta_k \phi(s)$
　　　$w_{a^*} \leftarrow w_{a^*} + \eta_k \phi(s)$
　　**end if**
　　L2 正則化: $w \leftarrow \lambda w$
　　平均 $\bar{w}$, 更新率 $\eta_k$ 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 $\bar{w}$.

# Zhang&Nivre2011特徴テンプレート(Z&N)の拡張(Z&N+)



# ベンチマークデータでのテスト性能

（Penn Treebank, Yamada&Matsumoto2003 head rule)

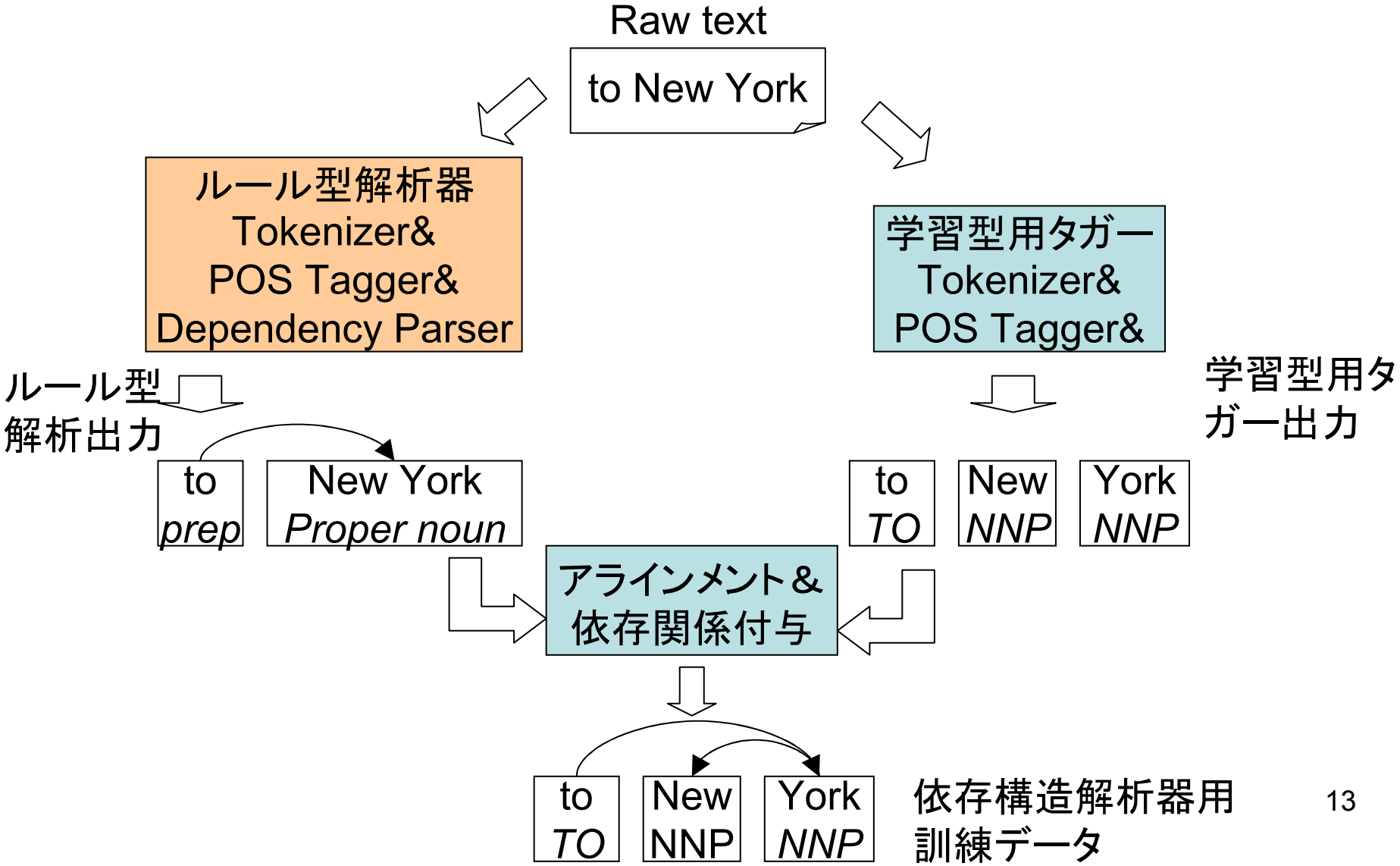
- 比較手法: Goldberg & Nivre 2012
- 依存ラベルあり決定的解析（ビーム探索なし）
- ラベルなし依存関係正答率（UAS；unlabeled attachment scores）の10 回試行平均

| 特徴 | 学習手法 | 平均 UAS(%) | σ | P値 |
|---|---|---|---|---|
| Z&N | Goldberg & Nivre 2012 | 90.14 | 0.05 | - |
| Z&N | 提案手法 | **90.53** | 0.03 | 0.002 |
| Z&N+ | Goldberg & Nivre 2012 | 90.91 | 0.08 | - |
| Z&N+ | 提案手法 | **91.22** | 0.08 | 0.002 |

# 発表の流れ

- 背景
- 遷移型依存構造解析器のオンライン学習アルゴリズムと特徴量の提案
  - ベンチマークデータでの比較結果
- ルール型から学習型への転移
  - ルール型解析器を用いた訓練データ生成
  - 異なるトークン列のアラインメント・依存関係付与
- 実験
  - 大規模データを使用した転移
  - 並列句の変更

# ルール型解析器を用いた訓練データ生成



# 異なるトークン列のアラインメント

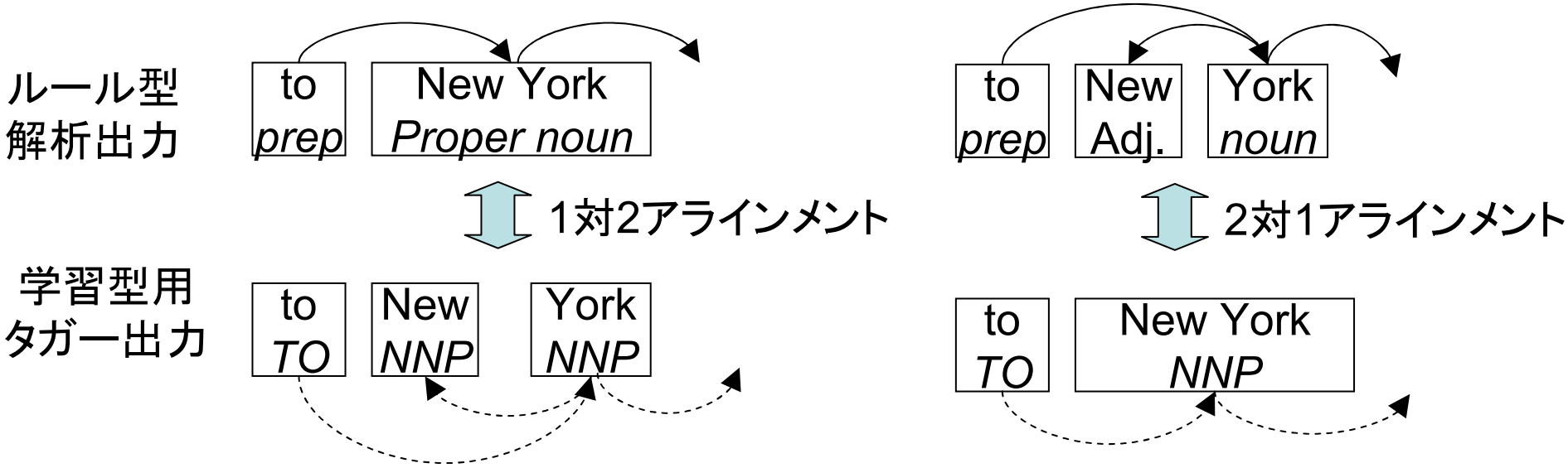
- 動的計画法によって編集距離を最小化するアラインメントを求める
  - 共通部分文字列がある場合のトークンの置換コストは文字列長の比
  - それ以外のコストは1

トークン単位1

| トークン単位2 | a | part- | time | job |
|---|---|---|---|---|
| a | 0 | 4/5 | 1 | 1 |
| part-time | 8/9 | 4/9 → 5/9 | | 1 |
| job | 1 | 1 | 1 | 0 |

# 1対N対応トークンの依存関係付与

- 1:N対応
  - 最右を主辞
  - それ以外は最右を親とする
- N:1対応
  - 最右の親を句の親



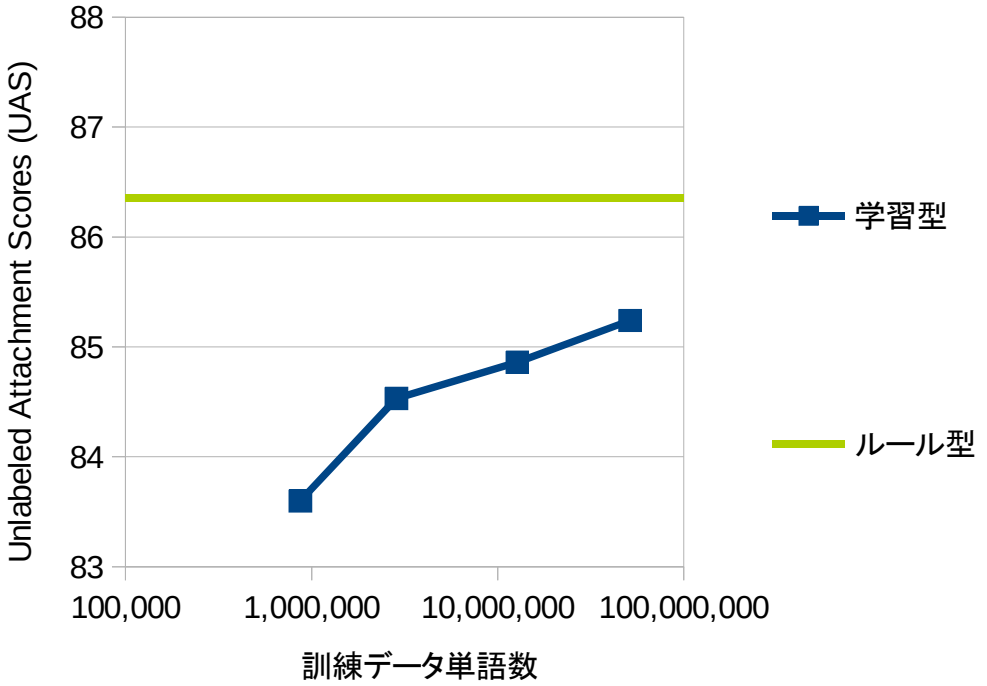

多対多のアラインメントは少数であるため訓練データから除外

# 発表の流れ

- 背景
- 遷移型依存構造解析器のオンライン学習アルゴリズムと特徴量の提案
  - ベンチマークデータでの比較結果
- ルール型から学習型への転移
  - ルール型解析器を用いた訓練データ生成
  - 異なるトークン列のアラインメント・依存関係付与
- 実験
  - 大規模データを使用した転移
  - 並列句の変更

# ルール型→学習型転移の実験結果

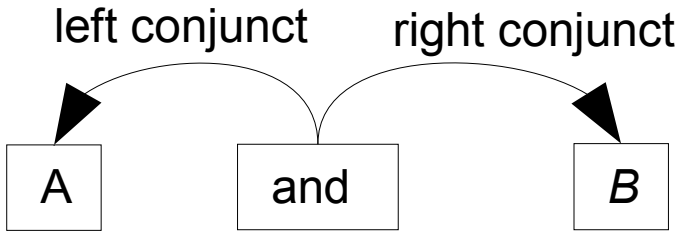
- 実験データ: European Parliament Proceedings Parallel Corpus (Europarl)：約5000 万語，約200 万文
- スロット文法に基づくルール型構文解析器(ESG)
- 評価用データ：799 文（約2万語）を人手で修正（交差依存構造木を含む）
- 訓練データを増やせば増やすほど（80万→5000万語）ルール型解析器の性能に近づく
- ルール型解析器,O(N$^3$),に比べて17倍高速（O(N)）



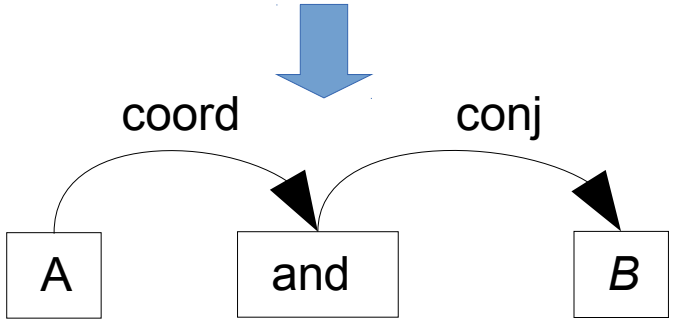
# 並列句の変更

- 特に並列句の解析誤りが多い
- 左から右に解析する遷移型依存構造解析が解析しやすい形に並列句を(非可逆)変換

ルール型解析器による
並列句の解析結果

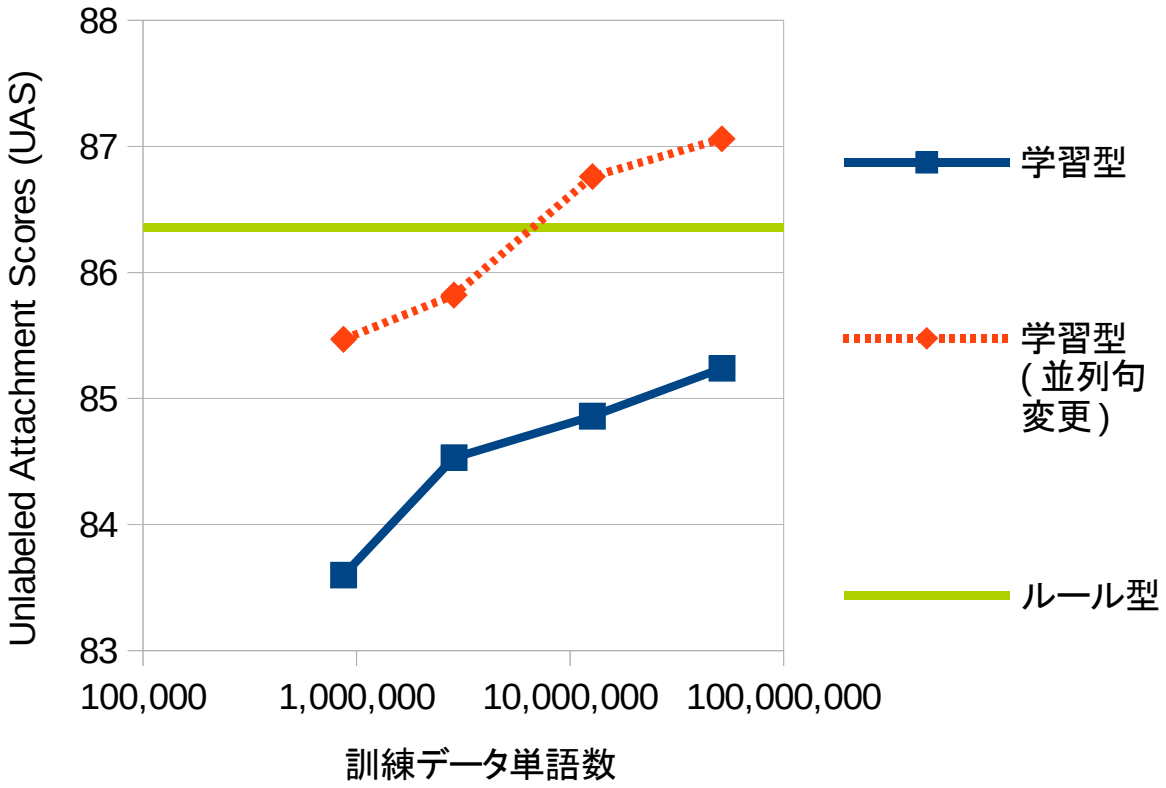



CoNLL型に並列句を変更



# 訓練データの並列句変更による解析精度向上

- 注：解いている問題が違うため直接の比較はできない
- 並列句の解析精度も改善



# まとめ

- 既存のルール型解析器を訓練データの生成に利用し、依存構造解析器を学習
  - 依存構造解析器のオンライン学習アルゴリズムと特徴テンプレートを提案
  - 訓練データを増やせば増やすほどルール型解析器の性能に近づく
  - 解析アルゴリズムに合った問題に変換すると性能が大きく向上
  - ルール型解析器に比べて一桁高速化($O(N^3)$→$O(N)$)