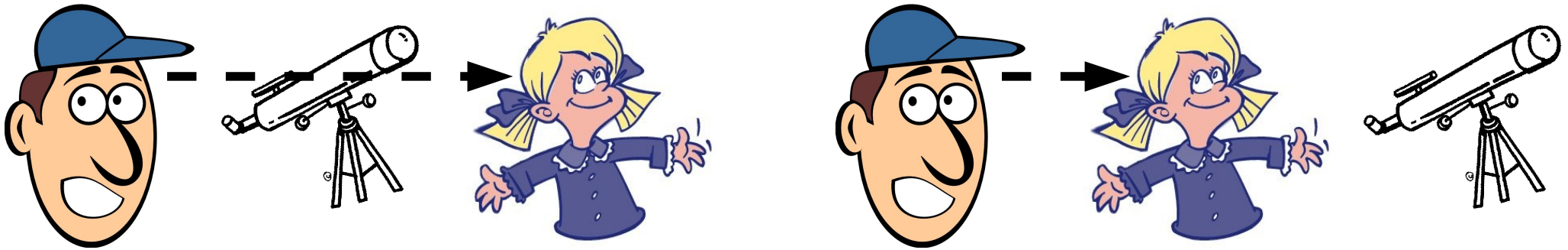
自然言語処理プログラミング勉強会 12
係り受け解析

Graham Neubig
奈良先端科学技術大学院大学 (NAIST)

自然言語は曖昧性だらけ！

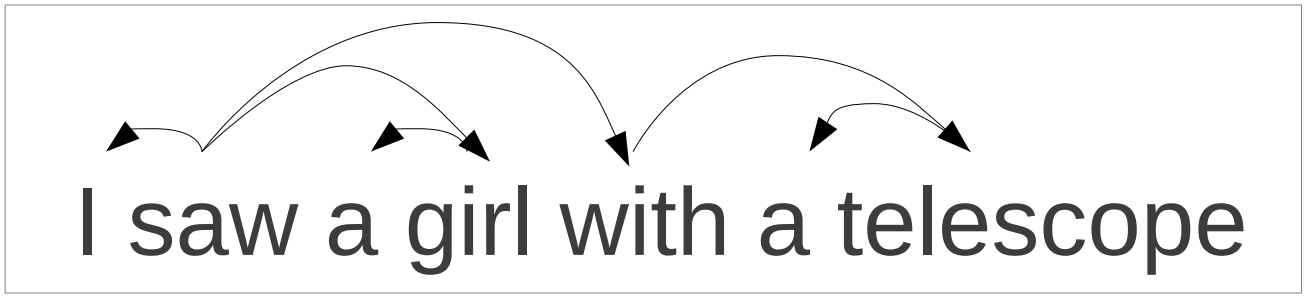
I saw a girl with a telescope



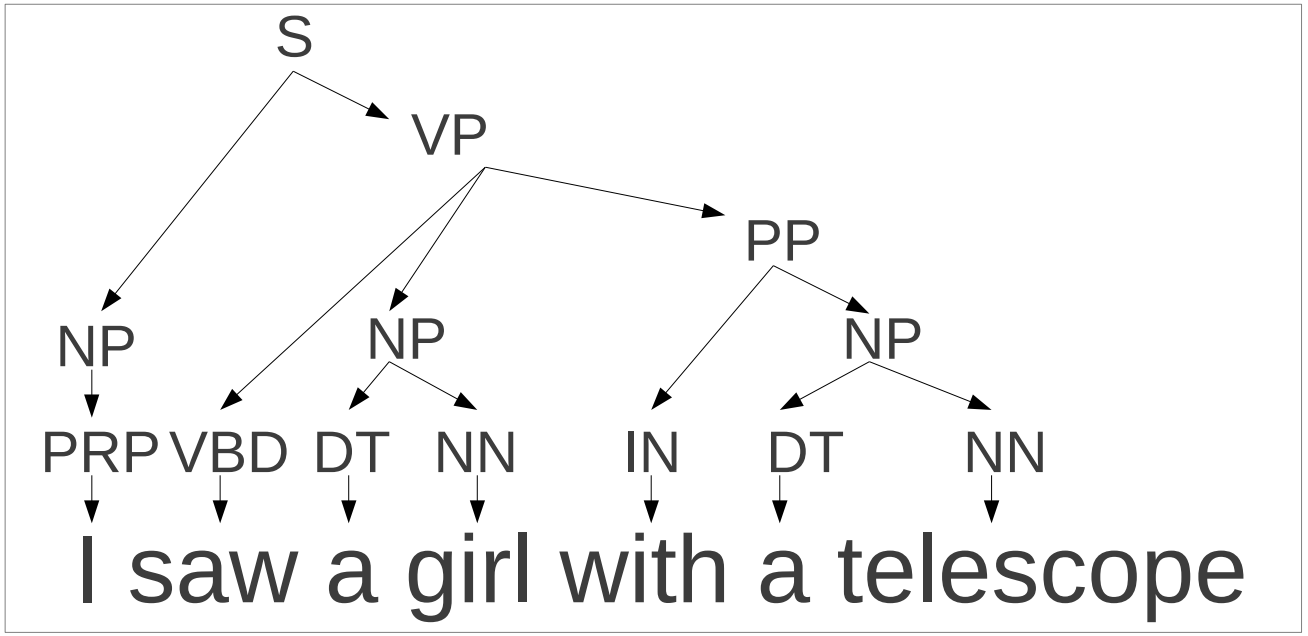
- 構文解析（パージング）は構造的な曖昧性を解消

構文解析の種類

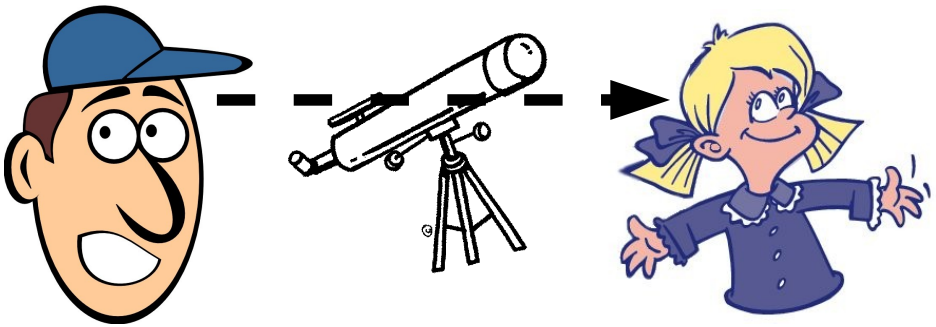
- 係り受け解析 : 単語と単語のつながりを重視

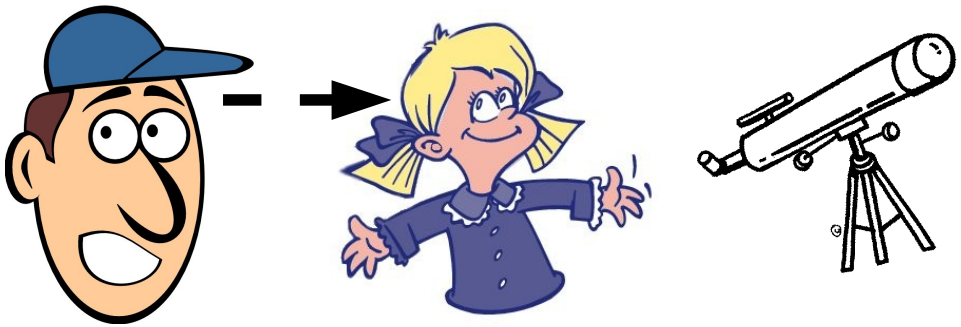


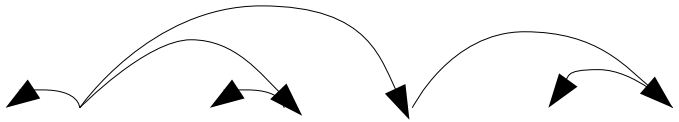
- 句構造解析 : 句とその再帰的な構造を重視



係り受けを使った曖昧性解消





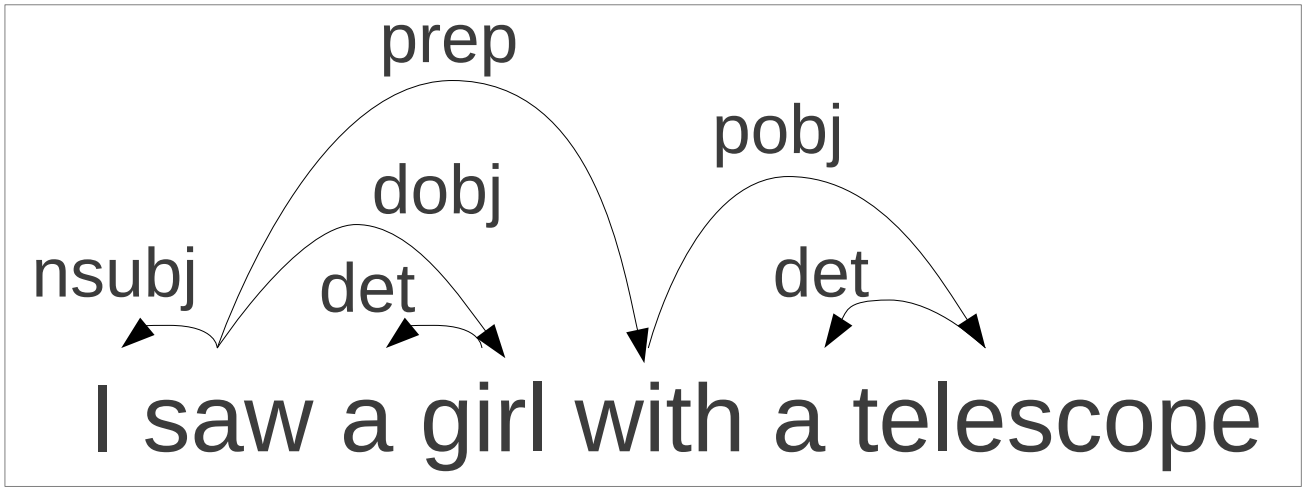


I saw a girl with a telescope

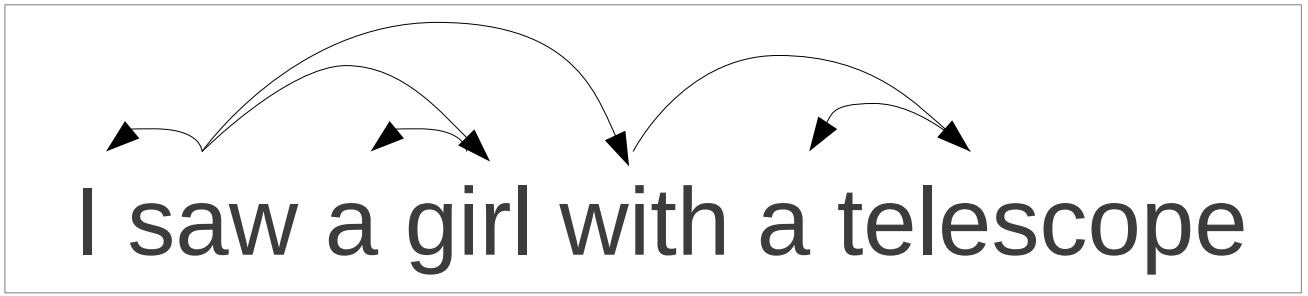
I saw a girl with a telescope

係り受けの種類

- ラベルあり： 単語間の関係の種類を記述



- ラベルなし： 関係の種類を問わず、親と子のみ

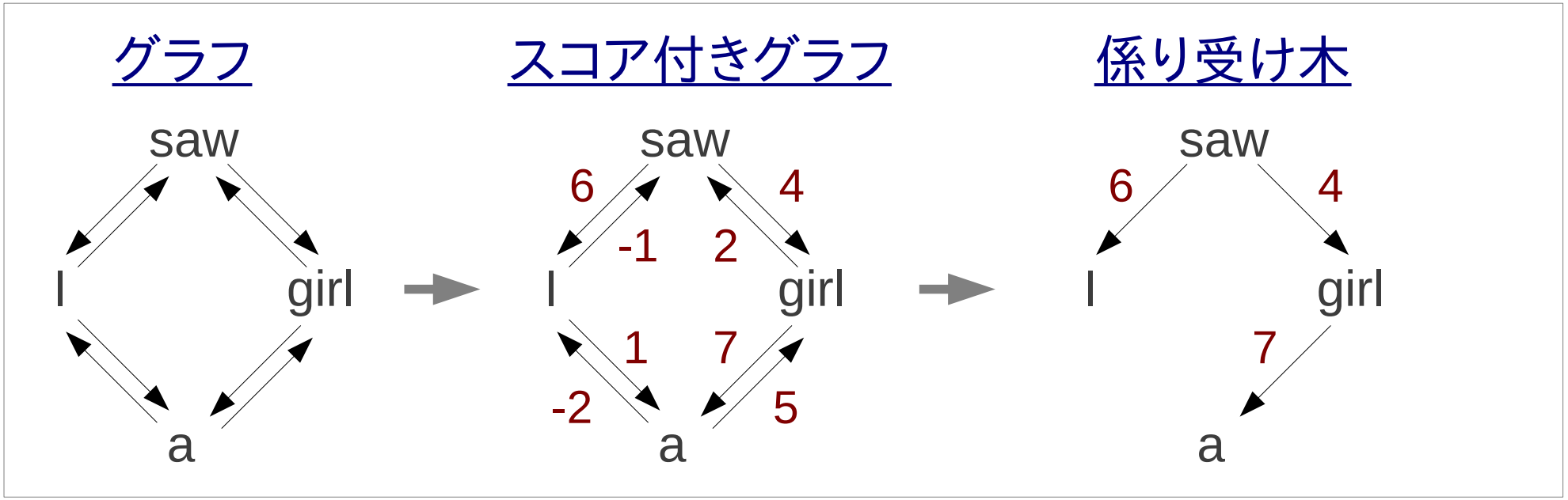


係り受け解析手法

- Shift-reduce
  - 左から右へ貪欲的に解析
  - 高速（線形時間）、少し精度が低い
  - ソフト： MaltParser
- 全域木
  - 文全体の係り受けを最適化
  - 精度が少し高く、スピードが少し落ちる
  - ソフト： MSTParser, Eda
- チャンク同定の段階適用
  - 1) 単語を句にチャンキング、 2) 主辞（親）を発見の繰り返し
  - ソフト： CaboCha

最大全域木

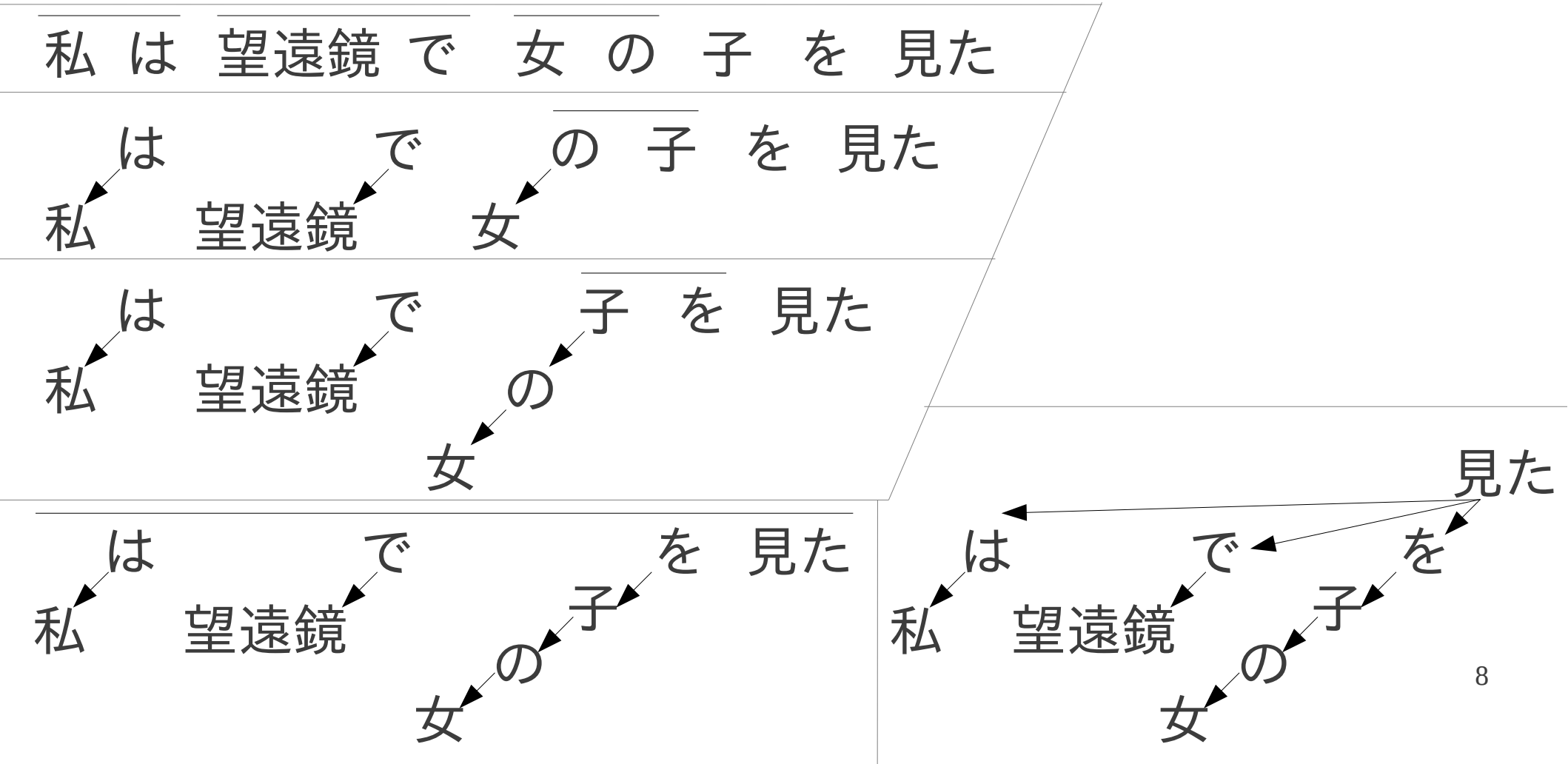
- 各係り受けは有向グラフの辺
- 機械学習を用いて辺に対してスコアを付与
- スコア最大の木を返す



(Chu-Liu-Edmonds アルゴリズム)

チャンク同定の段階適用

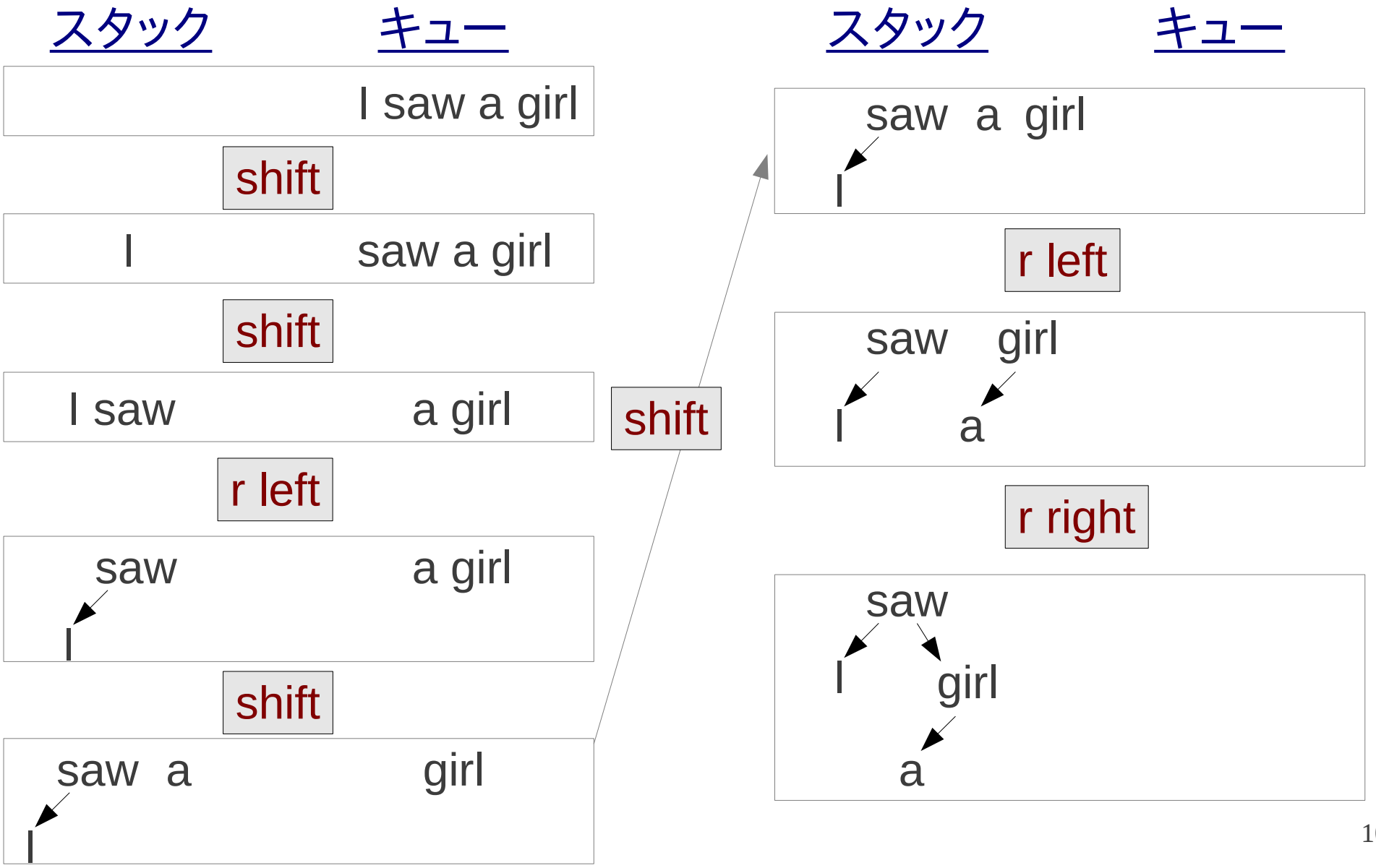
- 親が必ず最後にくる日本語に対して有効
- 文をチャンクに分割、親を右の単語にする



Shift-Reduce

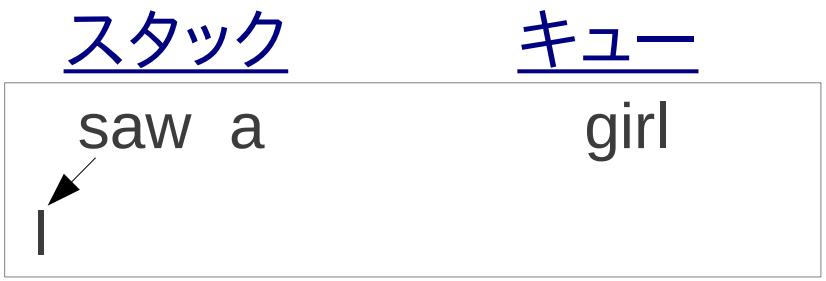
- 左から右へ単語を 1 個ずつ処理
- 2 つのデータ構造
  - キュー: 未処理の単語を格納
  - スタック: 処理中の単語を格納
- 各時点で 1 つの動作を選択
  - shift: 1 単語をキューからスタックへ移動
  - reduce 左: スタックの 1 単語目は 2 単語目の親
  - reduce 右: スタックの 1 単語目は 2 単語目の親
- 分類器を使ってどの動作を取るかを学習

Shift-Reduce の例



Shift-Reduce のための分類

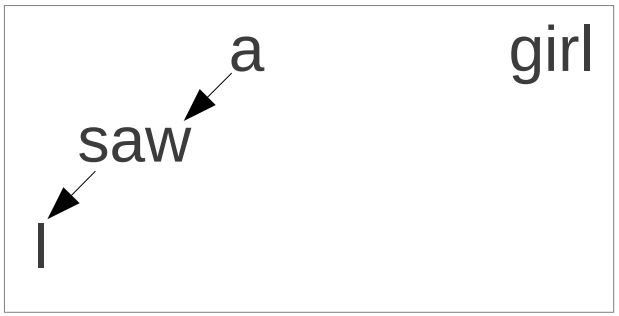
- 状態が与えられ：



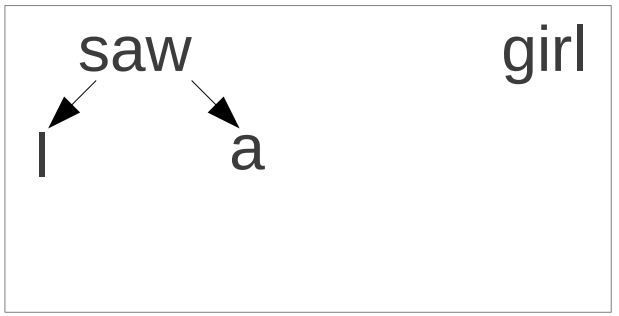
- どの動作を選択？

shift ?

r left ?



r right ?



- 正しい選択 → 正しい係り受け木

Shift-Reduce のための分類

- 「shift」「reduce 左」「reduce 右」の重みベクトル

  w_s w_l w_r

- キューとスタックに基づいて素性関数を計算

  $\varphi(queue, stack)$

- 素性関数と重みを掛け合わせてスコアを計算

  $s_s = w_s * \varphi(queue,stack)$

- スコアが最大の動作を利用

  $s_s > s_l \;\&\&\; s_s > s_r$ → do shift

Shift-Reduce の素性

- 素性は少なくともスタックの最後の２項目、キューの最初の項目をカバー

	stack[-2]	*stack*[-1]	*queue*[0]
単語 :	saw	a	girl
品詞 :	VBD	DET	NN

(-2 → 最後から2番)
(-1 → 最後)
(0 → 最初)

$\varphi_{W\text{-}2saw,W\text{-}1a} = 1$

$\varphi_{W\text{-}2saw,P\text{-}1DET} = 1$

$\varphi_{P\text{-}2VBD,W\text{-}1a} = 1$

$\varphi_{P\text{-}2VBD,P\text{-}1DET} = 1$

$\varphi_{W\text{-}1a,W0girl} = 1$

$\varphi_{W\text{-}1a,P0NN} = 1$

$\varphi_{P\text{-}1DET,W0girl} = 1$

$\varphi_{P\text{-}1DET,P0NN} = 1$

アルゴリズムの定義

- アルゴリズム **ShiftReduce** の入力:
  - 重み w_s w_l w_r
  - *queue*=[(1, $word_1$, POS_1), (2, $word_2$, POS_2), …]
- スタックは特別な ROOT 記号のみを格納:
  - *stack* = [(0, “ROOT”, “ROOT”)]
- 戻り値は各単語の親の ID を格納した配列:
  - *heads* = [-1, $head_1$, $head_2$, …]

Shift-Reduce アルゴリズム

```
ShiftReduce(queue)
    make list heads
    stack = [ (0, “ROOT”, “ROOT”) ]
    while |queue| > 0 or |stack| > 1:
        feats = MakeFeats(stack, queue)
        s_s = w_s * feats              # “shift” のスコア
        s_l = w_l * feats              # “reduce left” のスコア
        s_r = w_r * feats              # “reduce right” のスコア
        if s_s >= s_l and s_s >= s_r and |queue| > 0:
            stack.push( queue.popleft() )    # shift を実行
        elif s_l >= s_r:                     # reduce 左を実行
            heads[ stack[-2].id ] = stack[-1].id
            stack.remove(-2)
        else:                                # reduce 右を実行
            heads[ stack[-1].id ] = stack[-2].id
            stack.remove(-1)
```

Shift-Reduce の学習

- パーセプトロンで学習可能
- 解析を行い、正解 *corr* が分類器の戻り値 *ans* と異なる際、重みを更新
- 例： if *ans* = SHIFT and *corr* = LEFT

$$w_s \mathrel{-}= \varphi(queue, stack)$$

$$w_l \mathrel{+}= \varphi(queue, stack)$$

「正解」の計算 (1)

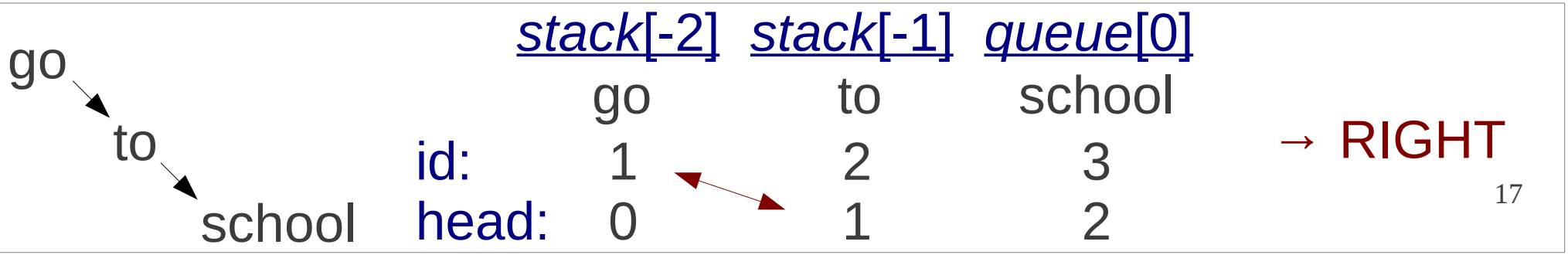
- 各単語の親 *head* を知っている場合、とりあえず：

stack[-1].*head* == *stack*[-2].*id*　　(左が右の親)
→ *corr* = RIGHT

stack[-2].*head* == *stack*[-1].*id*　　(右が左の親)
→ corr = LEFT

else
→ corr = SHIFT

- 問題 : reduce 右を速すぎる段階で実行！



「正解」の計算 (2)

- 各単語の未処理の子供の数 *unproc* を考慮 :

stack[-1].*head* == *stack*[-2].*id*　　(右が左の親)
stack[-1].*unproc* == 0　　(左に未処理の子供がない)
→ *corr* = RIGHT

stack[-2].*head* == *stack*[-1].*id*　　(左が右の親)
stack[-2].*unproc* == 0　　(右に未処理の子供がない)
→ *corr* = LEFT

else
→ *corr* = SHIFT

- 木を読み込みながらに *unproc* の初期値を計算
reduce を行うたびに *unproc* から 1 を引く
corr == RIGHT → *stack*[-1].*unproc* -= 1

Shift-Reduce の学習アルゴリズム

```
ShiftReduceTrain(queue)
    make list heads
    stack = [ (0, “ROOT”, “ROOT”) ]
    while |queue| > 0 or |stack| > 1:
        feats = MakeFeats(stack, queue)
        calculate ans                    # ShiftReduce と同様
        calculate corr                   # 前のスライドと同様
        if ans != corr:
            w_ans -= feats
            w_corr += feats
        perform action according to corr
```

CoNLL ファイル形式 :

- 係り受けの標準的な形式
- タブで区切られた行、空の 1 行で区切られた文

ID	単語	原型	品詞	品詞 2	拡張	親	ラベル
1	ms.	ms.	NNP	NNP	_	2	DEP
2	haag	haag	NNP	NNP	_	3	NP-SBJ
3	plays	plays	VBZ	VBZ	_	0	ROOT
4	elianti	elianti	NNP	NNP	_	3	NP-OBJ
5	.	.	.	.	_	3	DEP

演習課題

演習課題

- 実装　train-sr.py test-sr.py
- 学習　`data/mstparser-en-train.dep`
- 実行　`data/mstparser-en-test.dep`
- 評価　精度を測る
  `script/grade-dep.py`
- チャレンジ
  - 素性の拡張
  - 識別学習アルゴリズムの改良
  - よくある誤りの分析

Thank You!