自然言語処理勉強会@東京

統計的係り受け解析入門

日本アイ･ビー･エム（株）東京基礎研究所
海野　裕也（@unnonouno）

自己紹介 (1/2)

- 海野　裕也(うんの　ゆうや)
  - twitter: @unnonouno
  - blog: http://unnonouno.blogspot.com
  - NLPかプログラミングか写真の話題
- 日本アイ･ビー･エム　東京基礎研究所所属
  - 今はTRLという略称は使われていません:)
  - テキストマイニング，自然言語処理の研究開発
  - 主に，動的計画法と木構造と戯れている

自己紹介 (2/2)

@unnonouno

- 読めない
- unno / no / uno
  - ≠unnounno, unonono
  - たまにリプライ先を間違えられる
- 海野（うんの） / の / UNO
  - この文自体に特に意味はない
- かな漢字変換と分かち書きという日本語自然言語処理特有の問題を表したID

今日のお話

- 係り受け解析とは何か知ろう
- いくつかの重要な用語，記法を知ろう
- 典型的なアルゴリズムを知ろう

目次

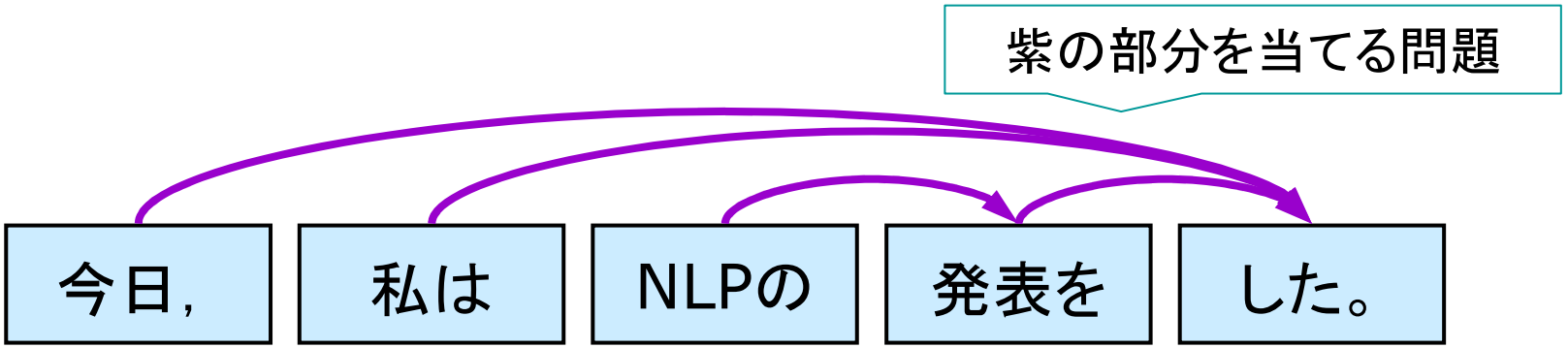
- 係り受け解析とは何か
  - 用語の説明
  - タスクの説明
- 手法の紹介
  - Shift-reduce法
  - Eisner法
  - その他の手法
- まとめ

目次

- 係り受け解析とは何か
  - 用語の説明
  - タスクの説明
- 手法の紹介
  - Shift-reduce法
  - Eisner法
  - その他の手法
- まとめ

係り受け解析とは何か？

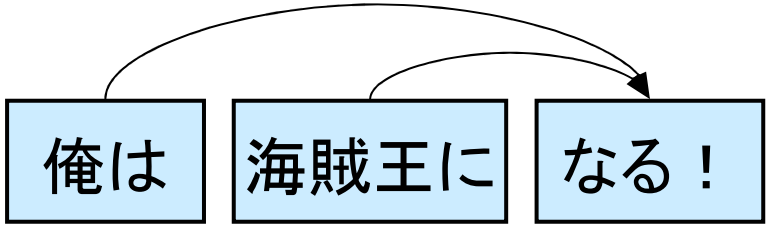
- 各単語（or文節）の係り先を決定する
- 係り先とは何ぞや？という問題は割愛します :)

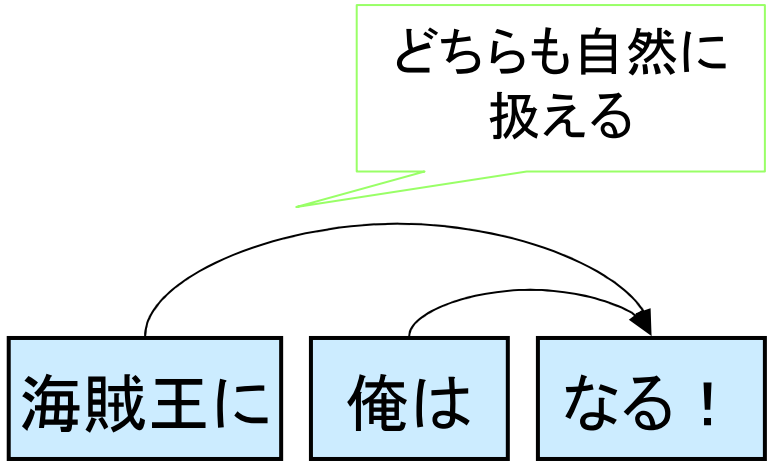


※日本語だと係り元から係り先を指すことが多い

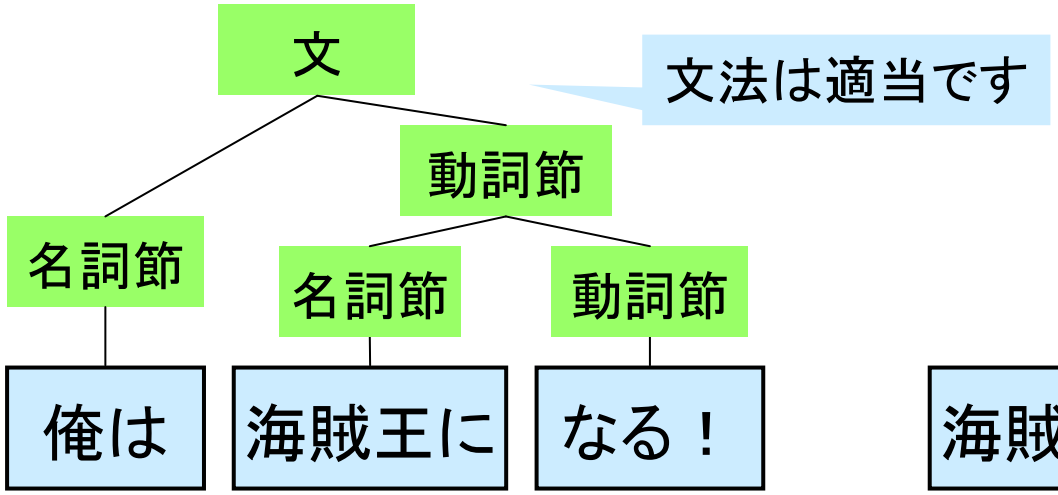
海賊王に俺はなる！

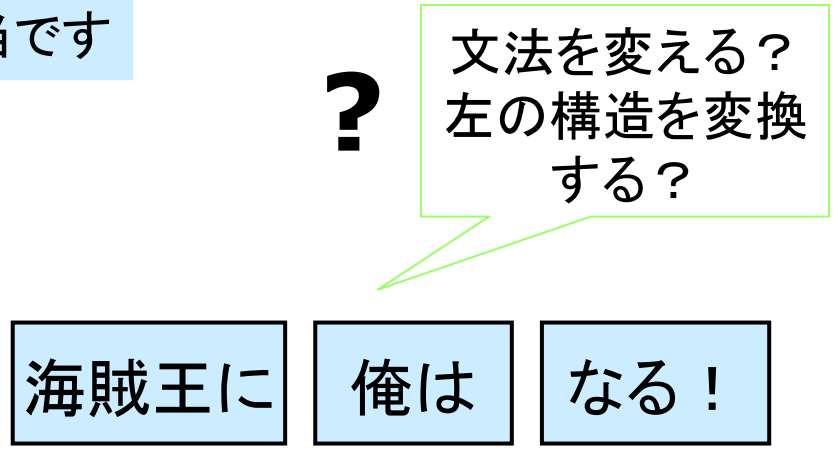
係り受け（依存文法）





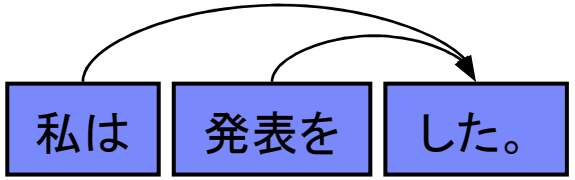
句構造文法

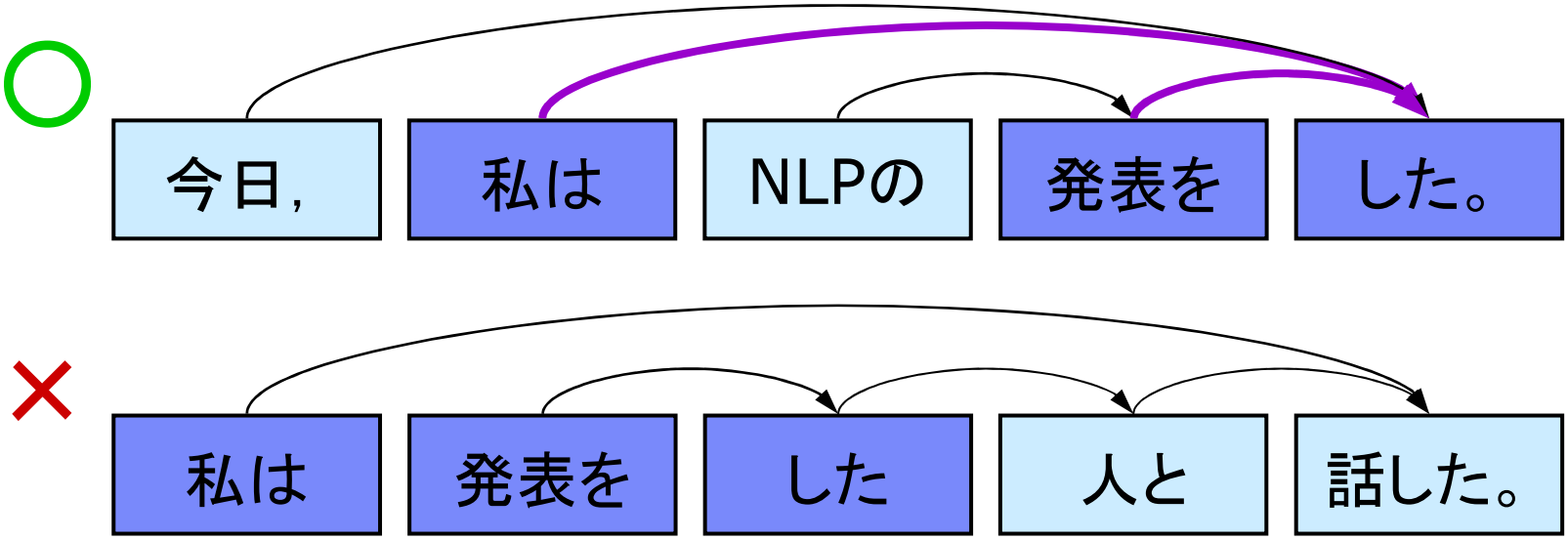




係り受け解析の応用例

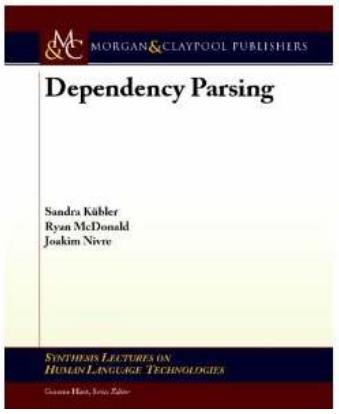
- 情報抽出に使う
  - –「私は・・・発表を・・・した」を探す





単なる単語共起ではゴミが多いときに有効

教科書の紹介

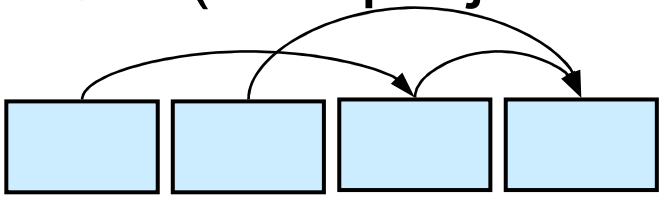


- Dependency Parsing (Synthesis Lectures on Human Language Technologies)
  - S. Kubler, R. McDonald, J. Nivre
  - MSTパーザーのR. McDonaldとShift-ReduceパーザーのJ. Nivreの本
  - 読みやすいのでお勧め

- 言語と計算 (4) 確率的言語モデル
  - 北 研二, 辻井 潤一
  - この本にもEisner法と同等の手法が載ってる
  - 最近の本を知っている人は教えてください

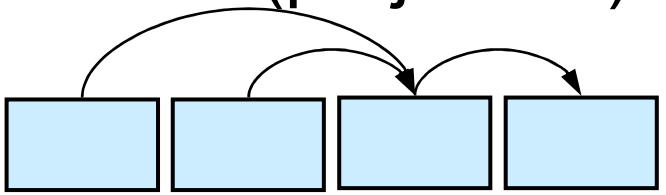
係り受け解析の用語

交差あり (non-projective)



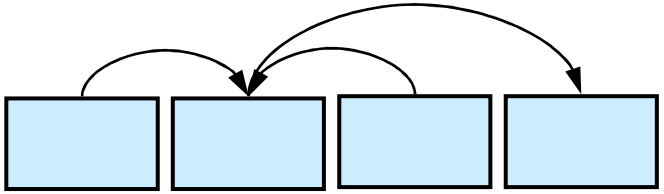
最近は交差を許す問題設定が多い

交差なし (projective)



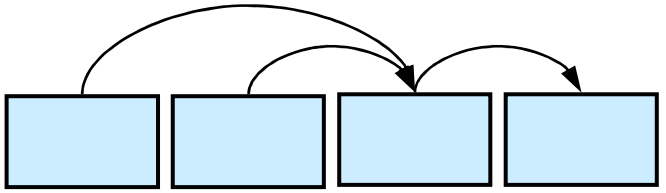
ほとんどの場合，交差しない

双方向



英語など一般的には双方向

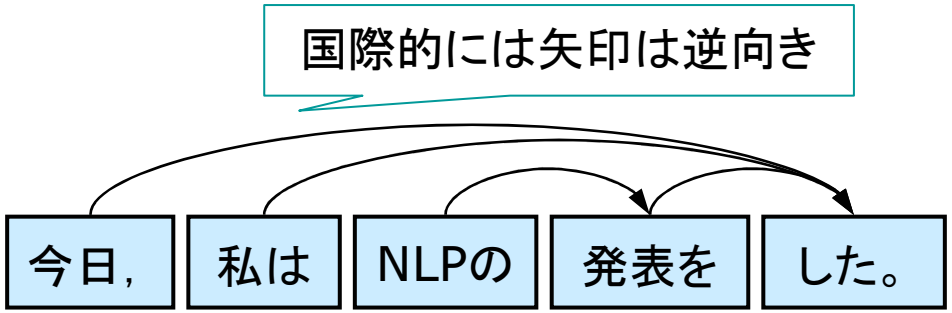
単方向



日本語書き言葉など一部の言語

今日は，交差なし・双方向を中心にはなします

問題の定式化



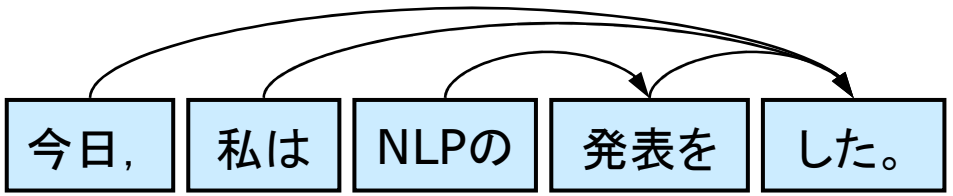
- 入力
  - 単語列 $S = \{w_1, \ldots, w_n\}$
- 出力
  - 係り先 $D = \{d_1, \ldots, d_n\}$
  - ただし，連結・非サイクルである必要アリ

日本語の場合，文節単位のことも多い

- 長さnの単語列の各単語の係り先を出力
- ラベル付きなどもあるが今回は無視

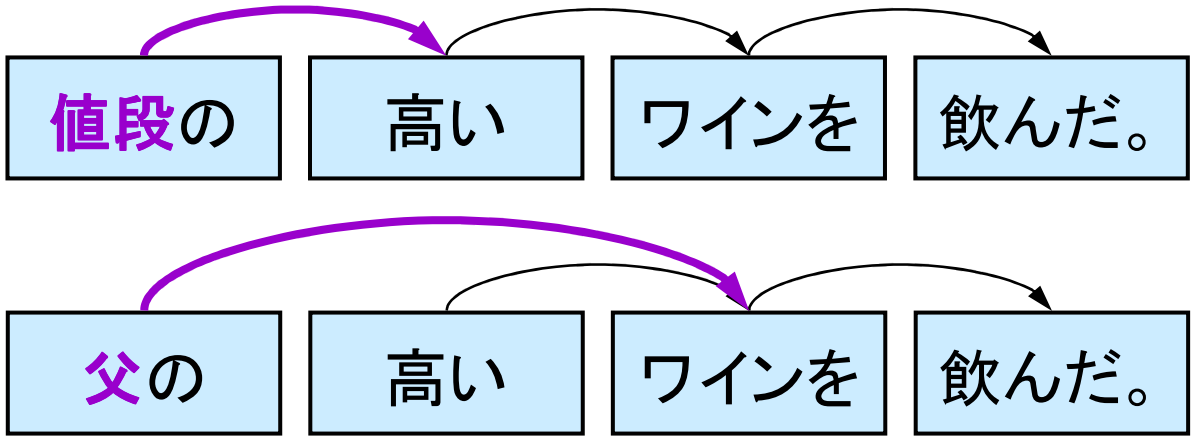
思いつくままにやってみる（品詞編）

- 係り関係になりそうな品詞組みを考える
  - –名詞，→動詞
  - –名詞は→動詞
  - –名詞の→名詞
  - –名詞を→動詞



- 係り関係になる品詞対をルール化すればよさそう

品詞だけでは解決できない例がすぐ思い浮かぶ



- 同じ名詞なのに係り先が異なる
- 本質的に曖昧な文も存在する
  - –ex. 黒い目の大きな女の子

プログラミング言語の構文解析との違い

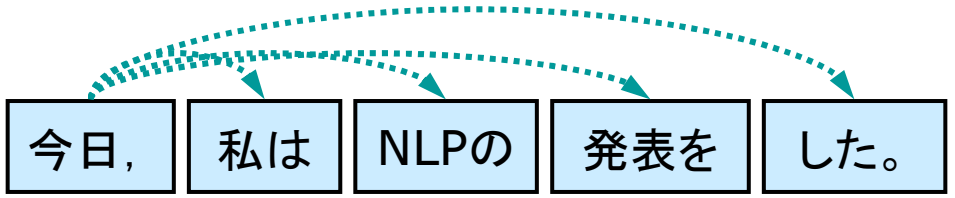
- プログラミング言語の文法は曖昧性がない
  - 1つの文に対して高々1つの構文木しか存在しない
  - 線形時間で必ず解析できるクラスに限定される

- 自然言語の文法は曖昧
  - 1つの文に対して解釈可能な構文木が複数存在する
  - 一般的には文長の指数個候補が存在する

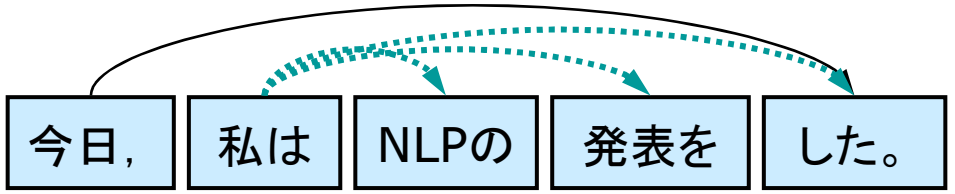
ルールのみでは1つの構文木に決定できない

機械学習を使ってみよう

- それぞれの単語の係り先を多値分類



- 交差しないよう，全単語に対して繰り返す



- これで結構いい精度がでる
  - –通称，相対モデル [工藤&松本04]
  - –多値分類は最大エントロピー法を使う

係り受け解析の難しさ

- 先の手法は全候補を網羅していない
- 出力の候補は入力長の指数オーダー以上
  - 各単語の係り先候補がnくらいあるので，出力Dの候補は大体n^nくらいある
- これを解消するために色んなテクを駆使する
  - 文法理論，機械学習，動的計画法，グラフ理論，整数計画法

本質的には膨大な候補存在する
ところをどう抑えるかが鍵

目次

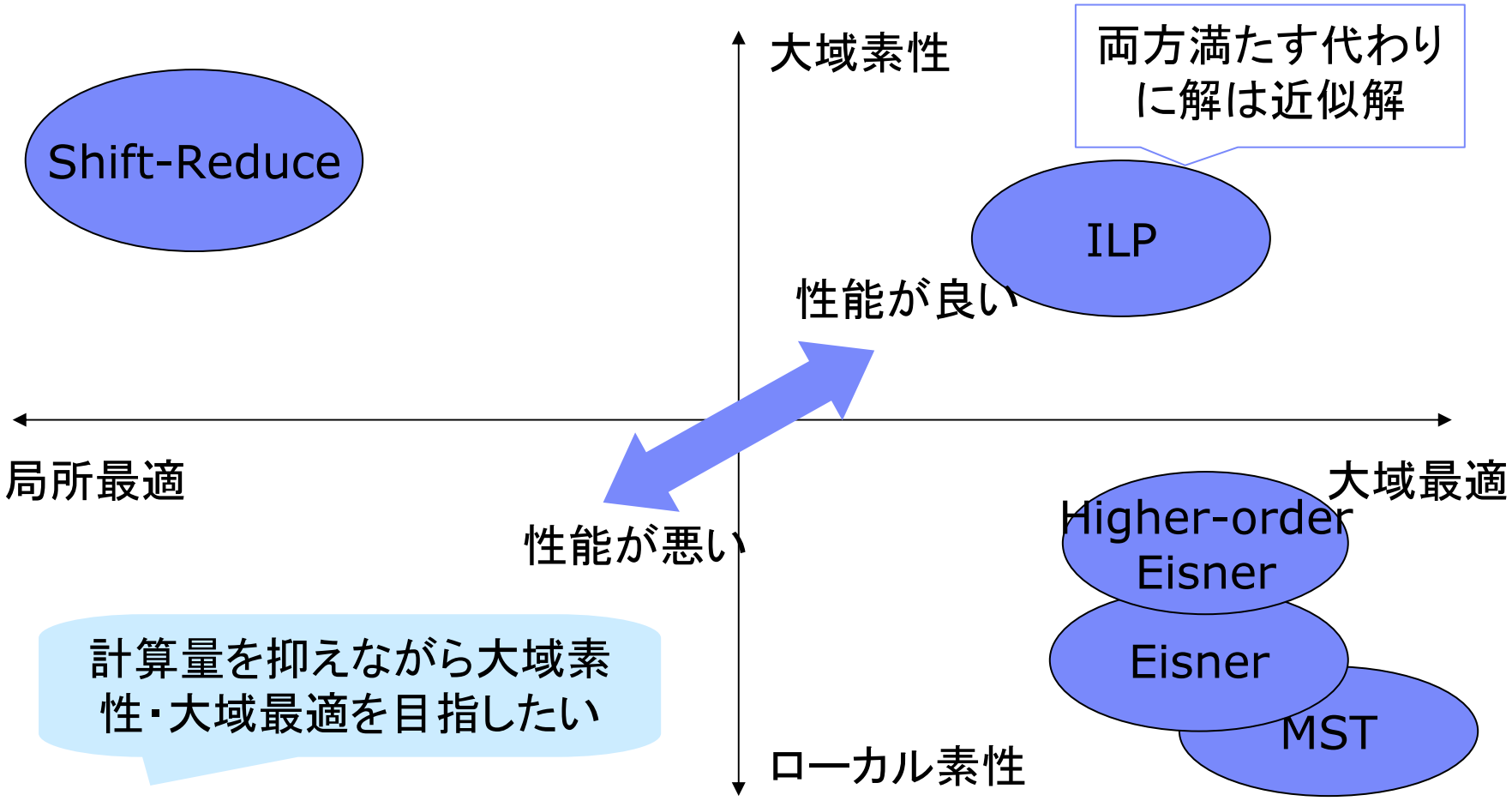
- 係り受け解析とは何か
  - 用語の説明
  - タスクの説明
- 手法の紹介
  - Shift-reduce法
  - Eisner法
  - その他の手法
- まとめ

係り受け解析の2大派閥

膨大な候補から選ぶ戦略として2つの手法方向性がある

- Transition-based（局所最適型）
  - –局所的な選択を繰り返すため，大域最適ではない
  - –複雑な特徴量を使える
  - –Shift-Reduce
- Graph-based（大域最適型）
  - –全候補の中からスコア最大の木を選ぶ
  - –単純な特徴量しか使えない
  - –MST，Eisner

両者を比較する



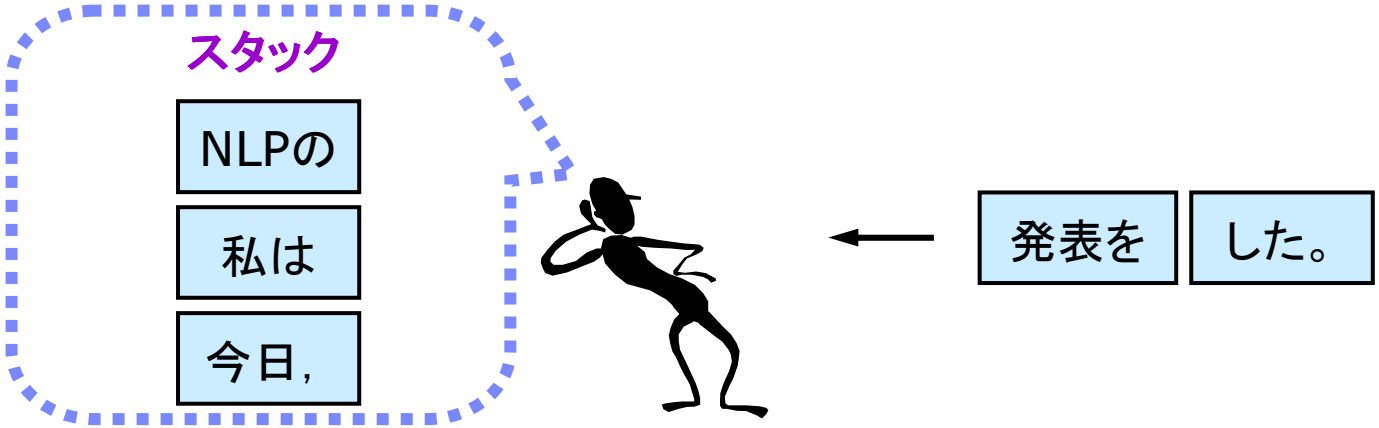
今日紹介する手法

- Shift-Reduce法
  - –Transition-basedの代表的な手法
  - –実装簡単，高速
- Eisner法
  - –Graph-basedの代表的な手法の一つ
  - –動的計画法は楽しい
  - –※こっちがメインです

Shift-Reduce法 [Nivre03, Yamada&Matsumoto03]

- 前から順に読んで，未処理の単語をスタックに積む
- スタックと次の単語間に係り関係があったら消す
- 前から読んでいく人間の理解に近い（？）

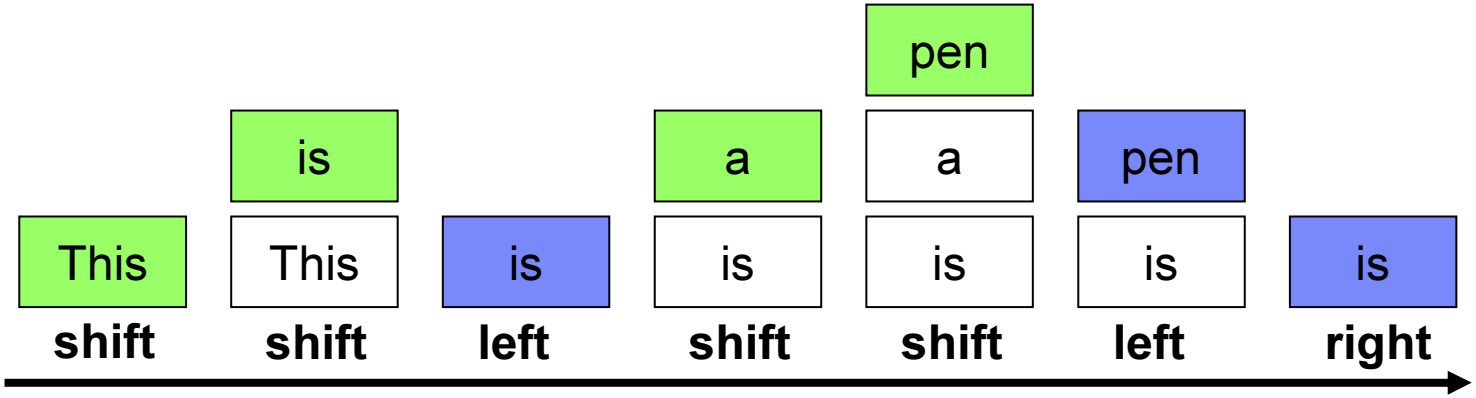
例）今日，私はNLPの・・・



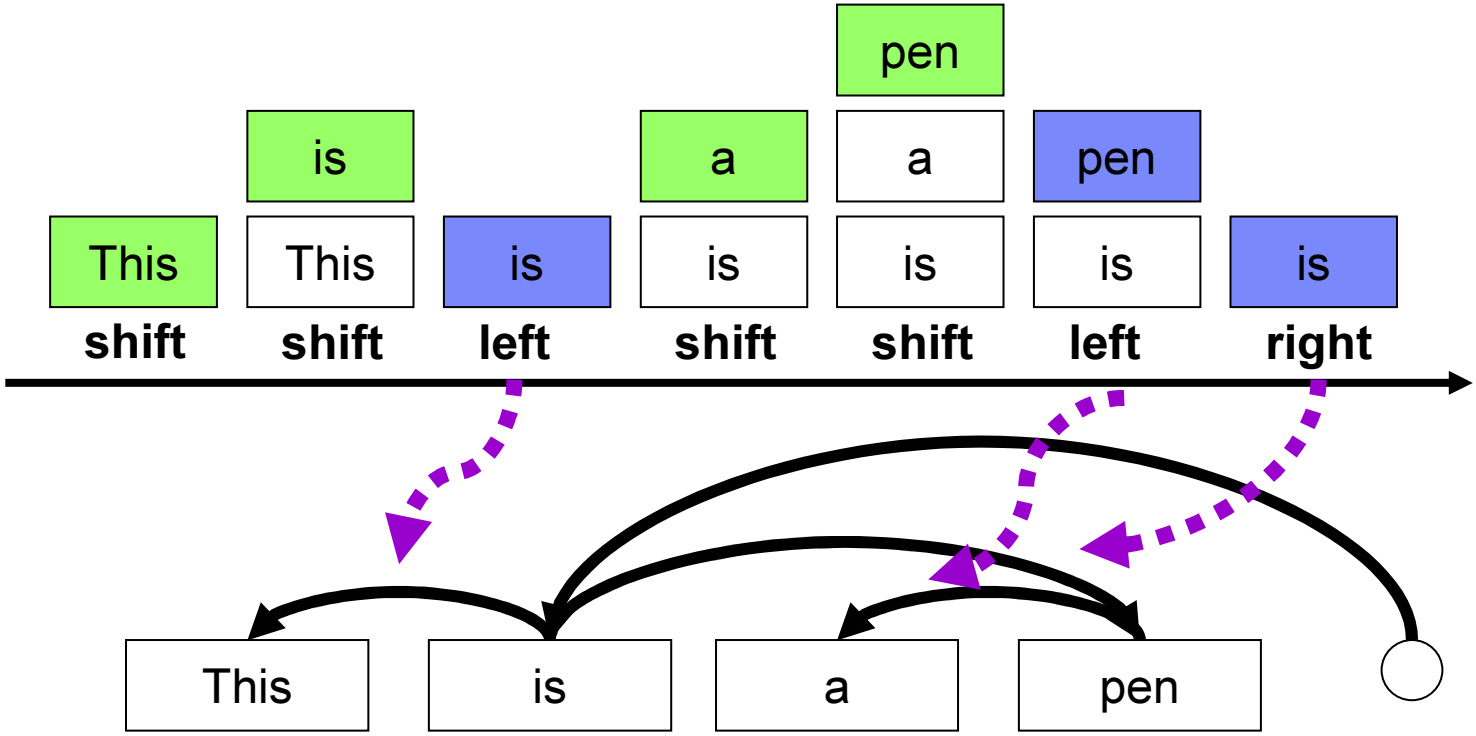
※係り先が決まっていない単語

Shift-Reduce法の特徴

- 積む(shift)か消す(reduce)の分類を繰り返す
- 利点
  - –高速(線形時間)，作るのが簡単
  - –遠くの構造を素性に入れられる
- 欠点
  - –大域最適ではない(garden pathなどに弱い)
  - –自明には交差に対応できない



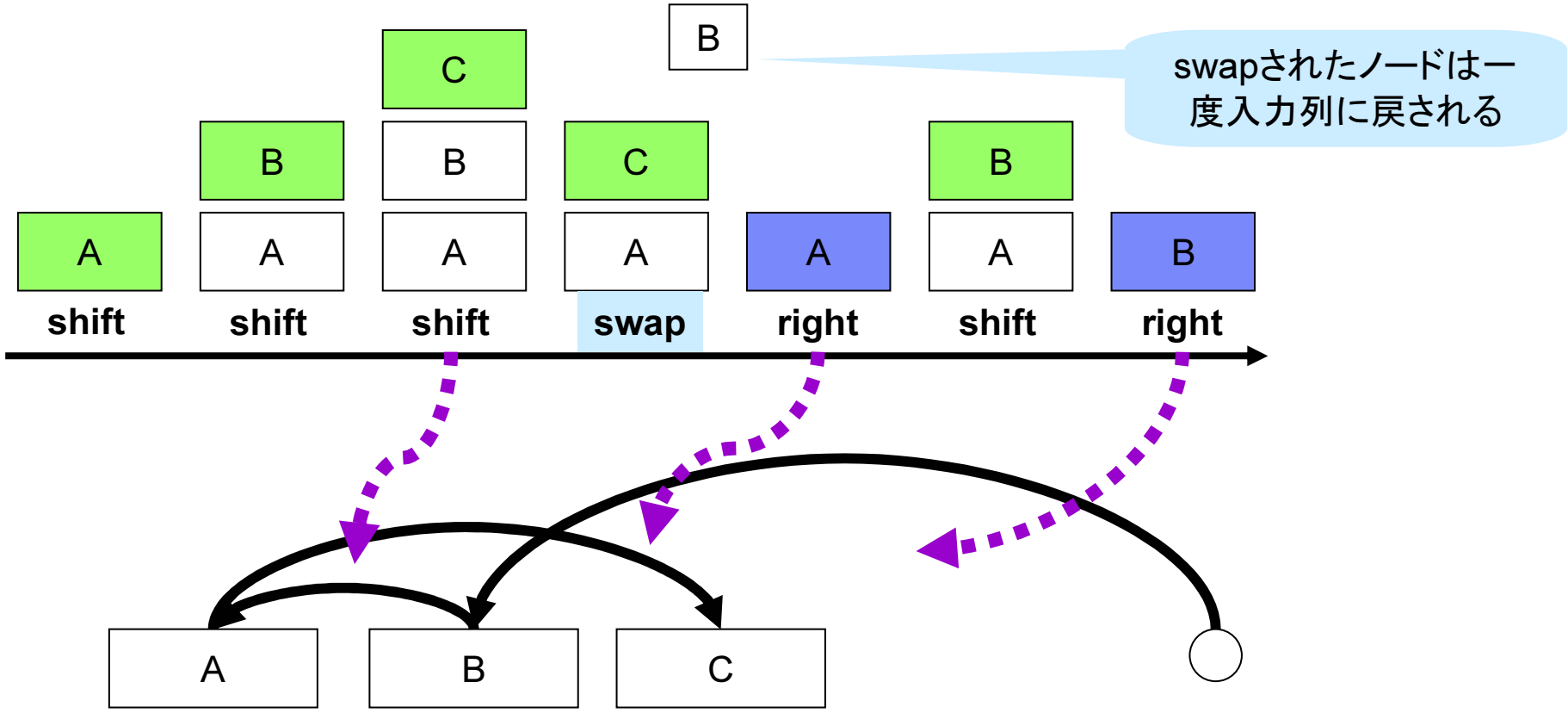
Shift-Reduceの詳細 [Nivre04]



- 入力をそのまま積む(shift)
- スタックの上2つに係り関係をつける(left / right)
- 操作の異なる亜種は多い
- これらの操作で生成される構造に交差はない

Swap操作で交差へ対応 [Nivre09]

- スタックの2番目の要素を戻すSwap操作を追加

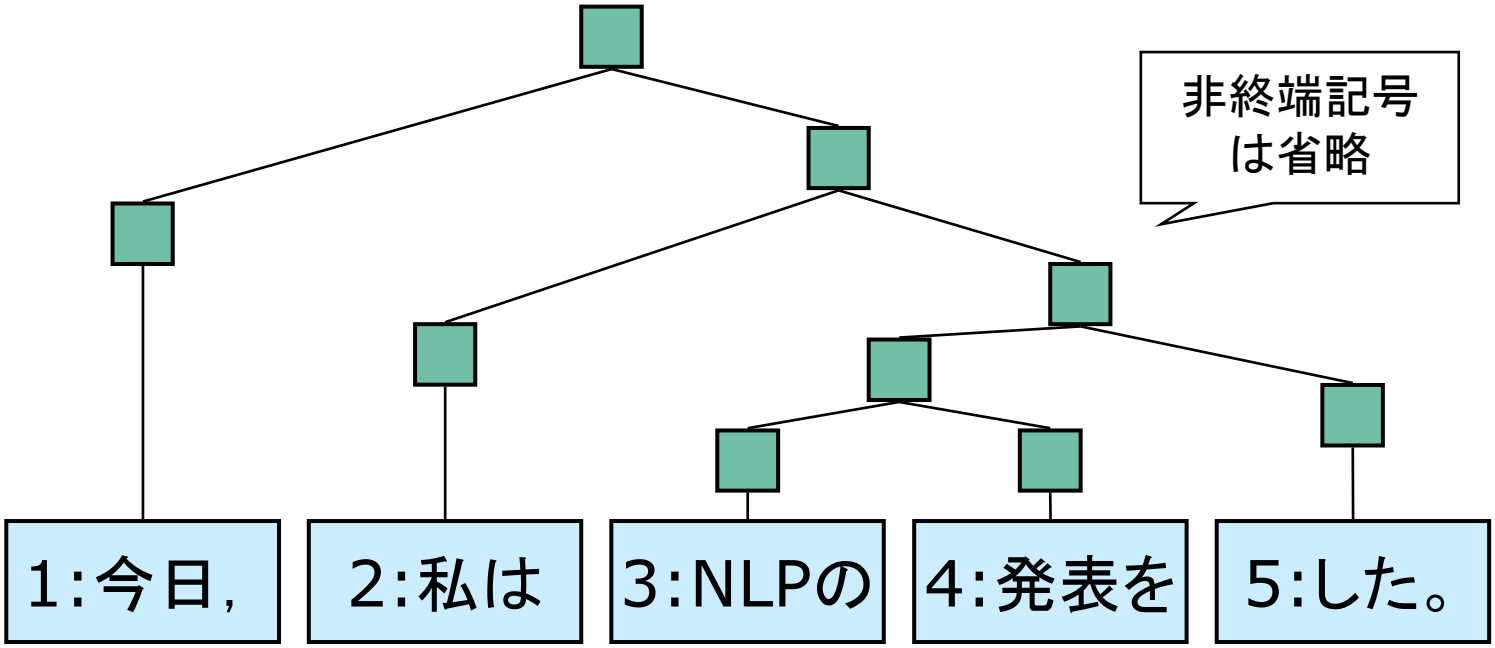


Eisner法

- 動的計画法により大域最適を探す
- CKYアルゴリズムと同種の方法を使う
  - 基本的に非交差しか扱えない
- 局所的な素性しか扱えない
  - “局所”の度合いで複雑度が異なる（cf. 高階Eisner）

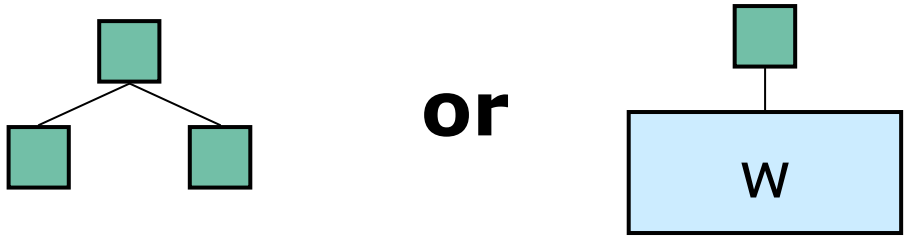
おさらい：CKYアルゴリズム

- 句構造文法の構文解析手法
- 下図のような木構造を出力する
- 動的計画法で計算量は$O(n^3)$



チョムスキー標準形 (Chomsky Normal Form, CNF)

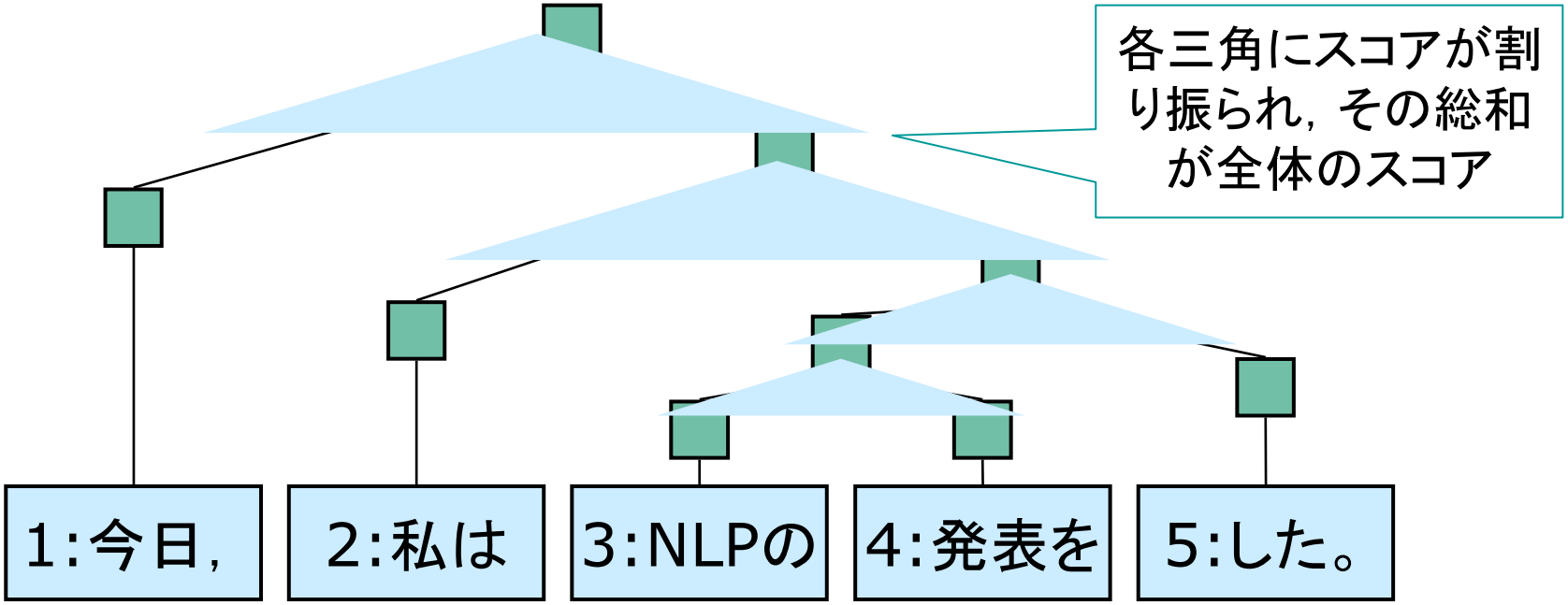
- 書き換え規則が2分割か，単語生成しかない文法



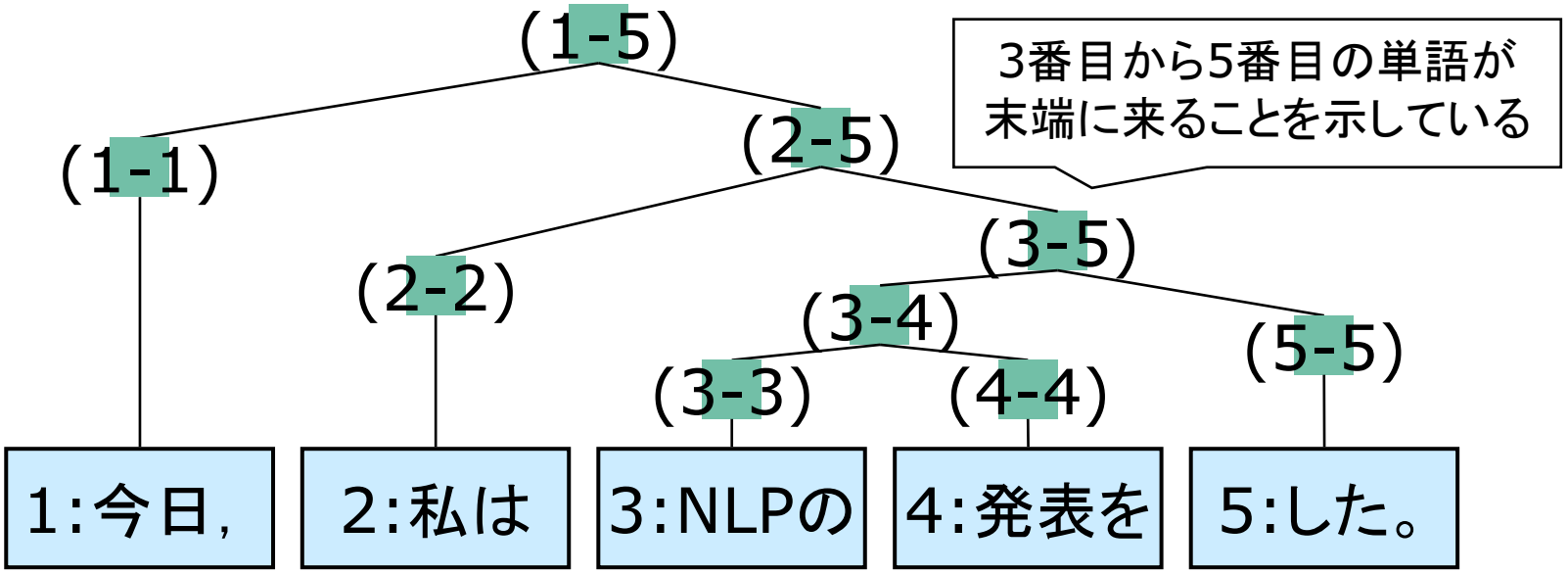
- 一般的には非終端記号（ ）にはラベルが振られている
- CKYアルゴリズムは文法がCNFである必要がある

CKYアルゴリズムの定式化

- sequenceに対するViterbiアルゴリズムのtree版
- 局所的なスコアの総和が最大になるtreeを選択する
- treeの候補は指数なので，全列挙はできない



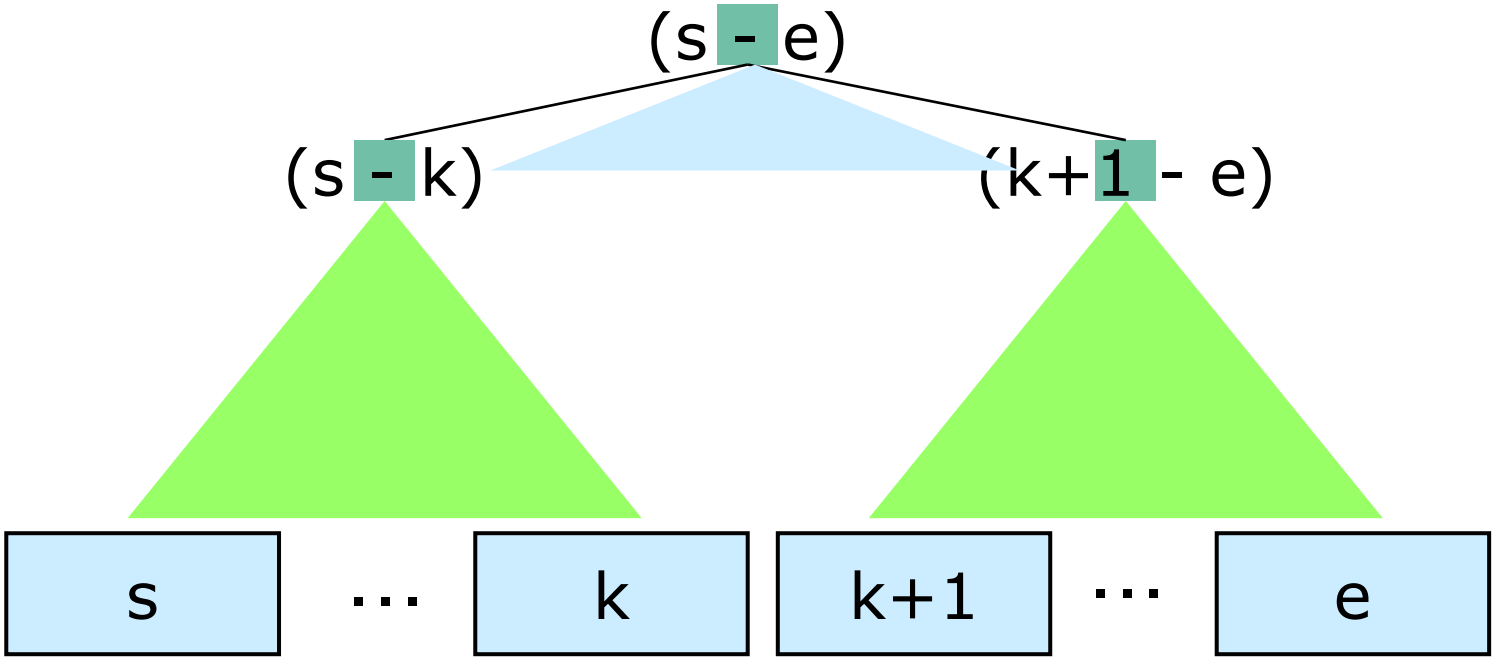
CKYアルゴリズムを再帰で理解する(1/3)



- 領域に関して，以下のルールが成り立つ
  - (s, e) → (s, k) + (k+1, e)
  - (s, s) → 終端

左側の木と右側の木の境目を
再帰的に決定している

CKYアルゴリズムを再帰で理解する(2/3)



- sからeの領域をkとk+1で分けたときの最大スコアは，sからkの領域の最大と，k+1からeの領域の最大と，三角の局所スコアの和
- kをsからe-1まで変化させて最大値をとるkを探す
- sからk，k+1からeの領域も再帰的に最大の構造を探す

CKYアルゴリズムを再帰で理解する(3/3)

領域(s,e)の最大スコア f(s, e) は

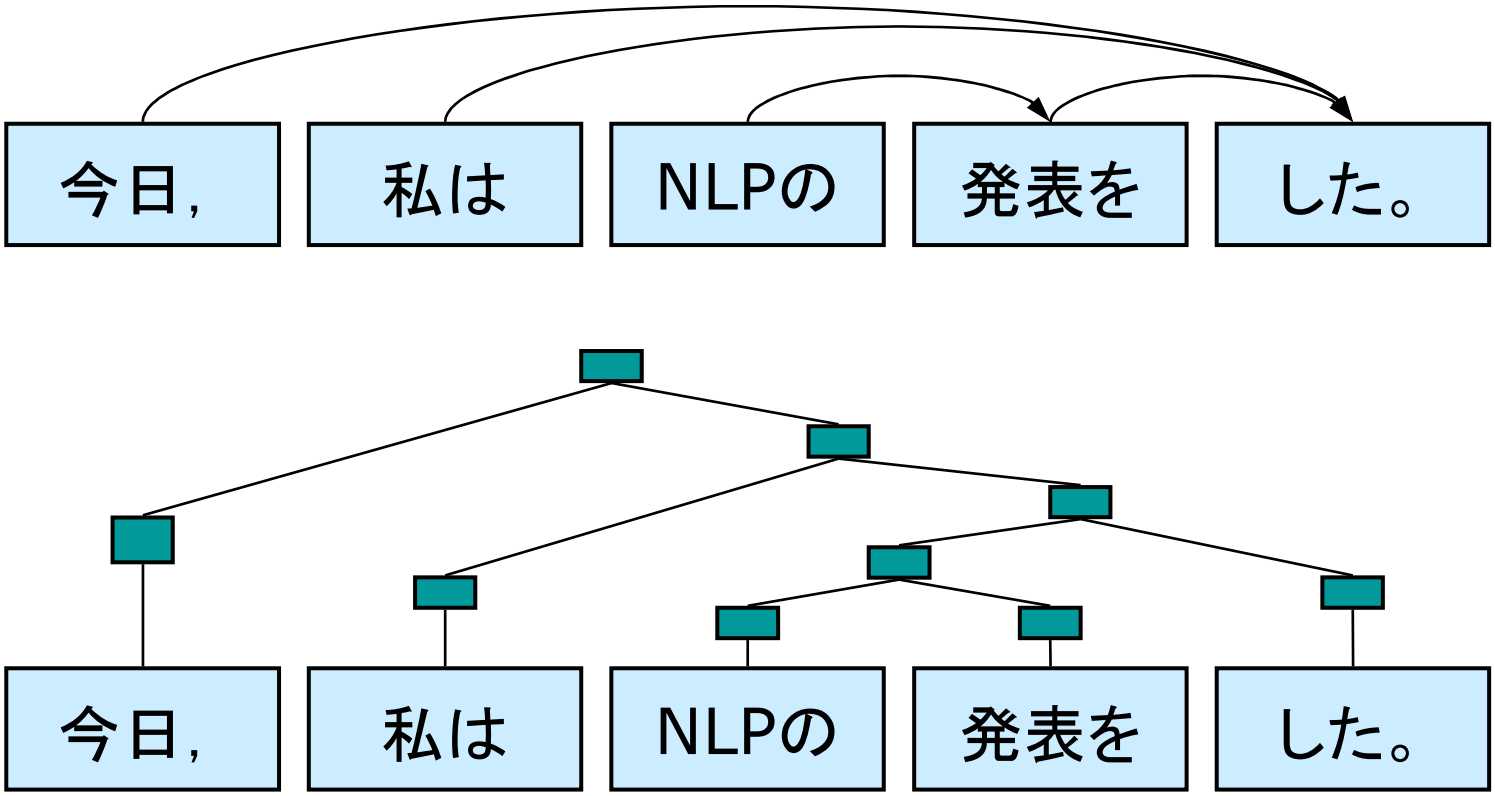
- $f(s, e) = \max_k \{S(s,k,e) + f(s, k) + f(k+1, e)\}$
- S(s,k,e) は (s,e) → (s,k) (k+1,e) という規則に対する局所的なスコア

- 領域(s,s)のスコア f(s,s) は
  - f(s, s)=0

- 上記再帰式をメモ化すればOK

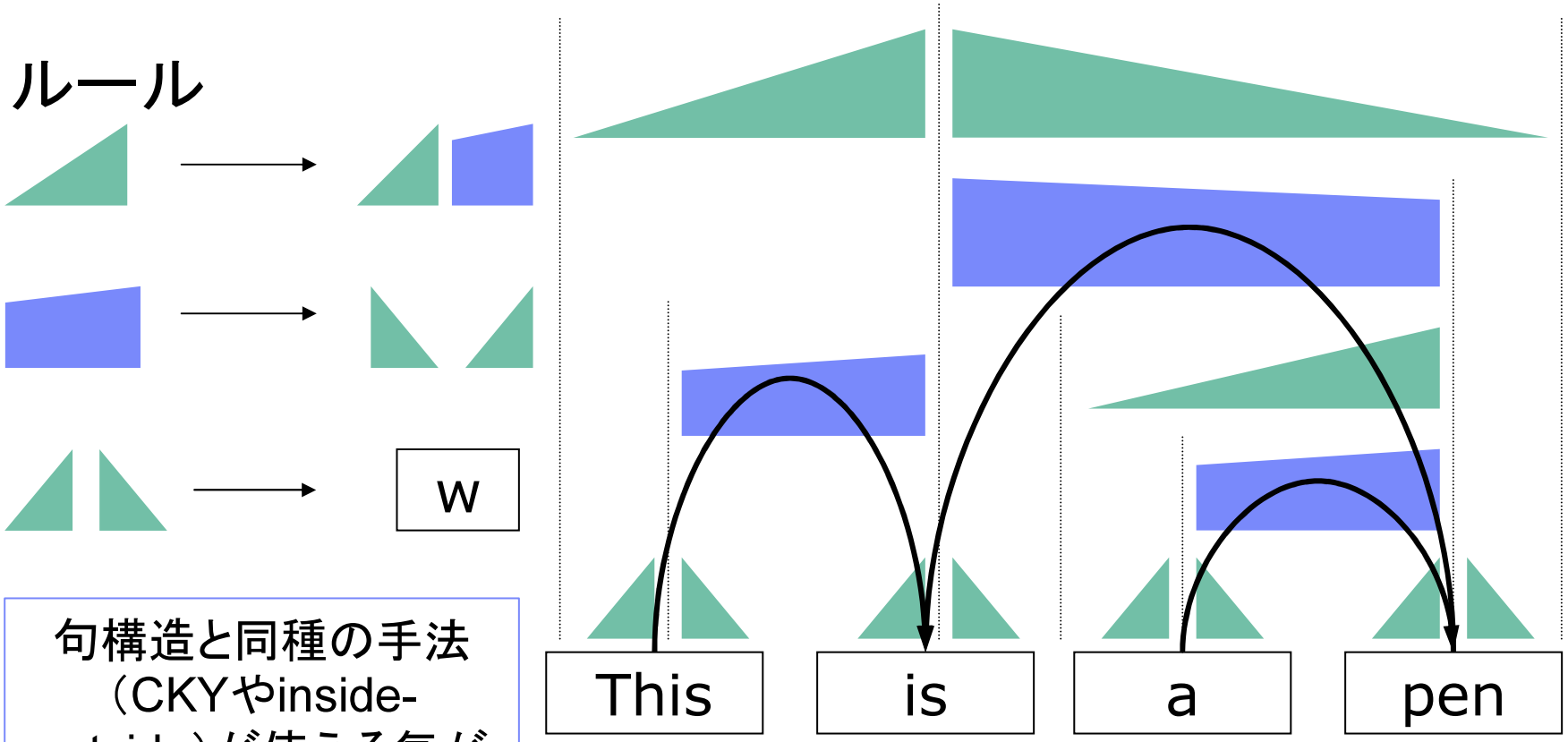
係り受けと句構造の比較

- なんとなく形が似ている気もする・・・



係り受けを句構造風に解釈するには？

- 関連の薄そうな2つのは，以下の記号を導入すると・・・

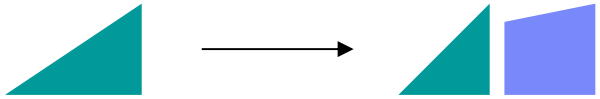


句構造と同種の手法（CKYやinside-outside）が使える気がしてくる！

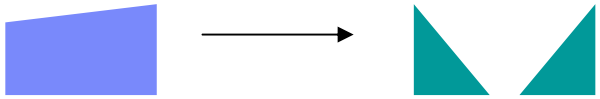
のところに係り受け

Eisner法を数式で書いてみる

Complete span



Incomplete span



- $C(s, e) = \max_k(C(s, k) + I(k, e))$
- $I(s, e) = S(s, e) + \max_k(C(k, s) + C(k+1, e))$
  - ただし，S(s, e)はsがeに係ることに対するスコア
- 上記再帰式をメモ化すればOK

Eisner法を擬似コードで書いてみる

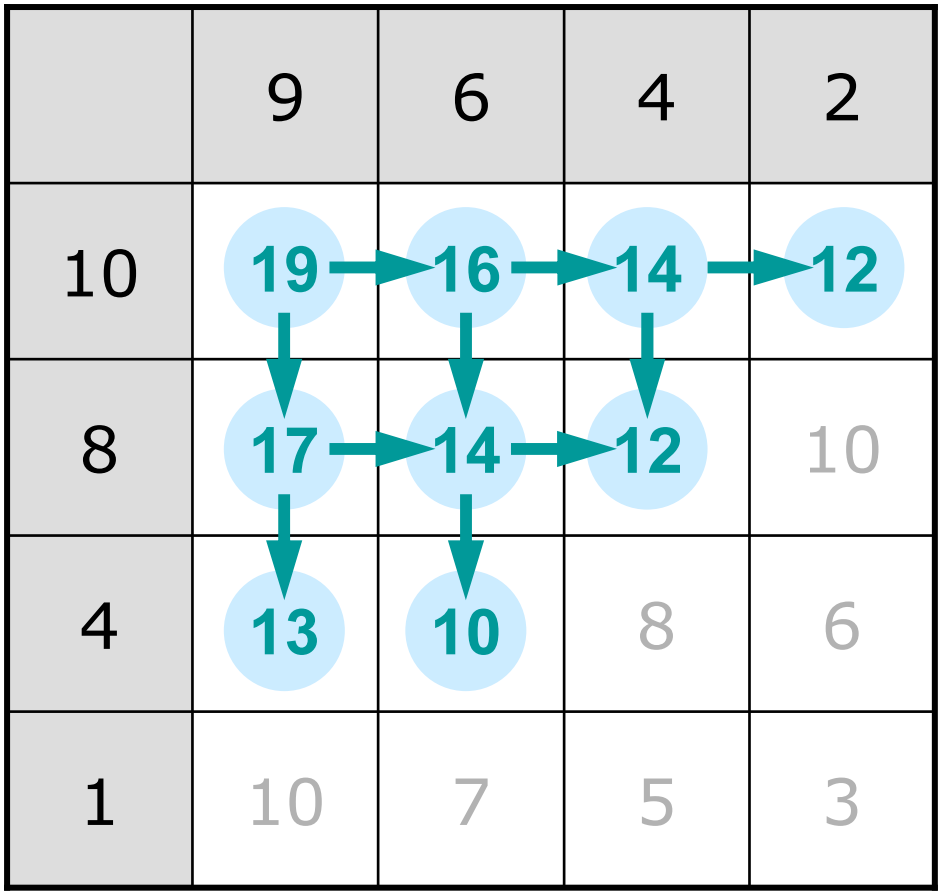
```
function comp(s, e):
  if s = e:
    return 0
  elif (s, e) in comp_cache:
    return comp_cache[(s, e)]
  else:
    m = -INF
    for k in {s, …, e}:
      m = max(m, comp(s, k) + incomp(s, k))
    return comp_cache[(s, e)] = m

function incomp(s, e):
  if (s, e) in incomp_cache:
    return incomp_cache[(s, e)]
  else:
    m = -INF
    for k in {s, …, e-1}:
      m = max(m, S(s, e) + comp(k, s) + comp(k+1, e))
    return incomp_cache[(s, e)] = m
```

Nベスト出力 [Jimenez&Marzal2000]

- スコアの高い上位N件を出力したい
- 元の論文はPCFGをN-best化する方法

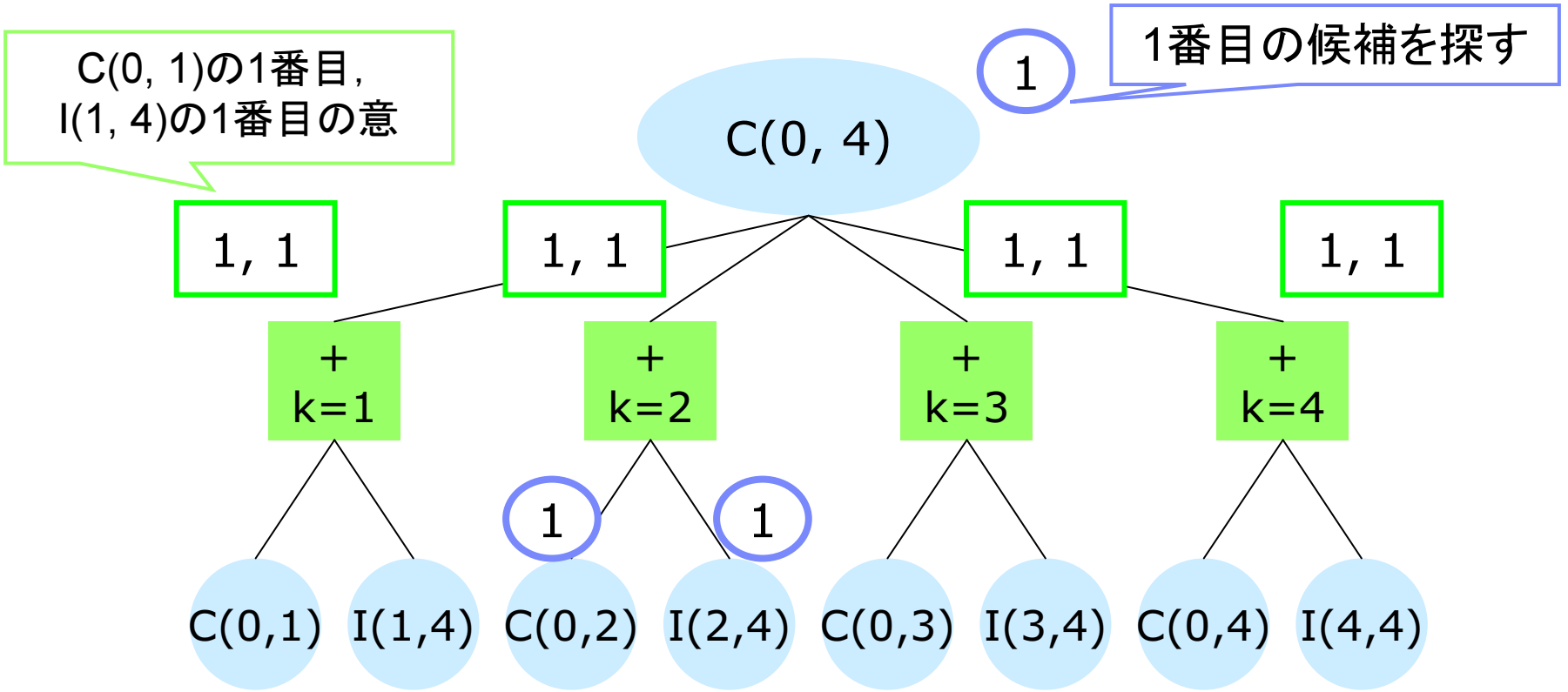
ソート済み配列の足し算



- ソート済み配列の足し算をソートする
- priority queueを使えば簡単に実現できる

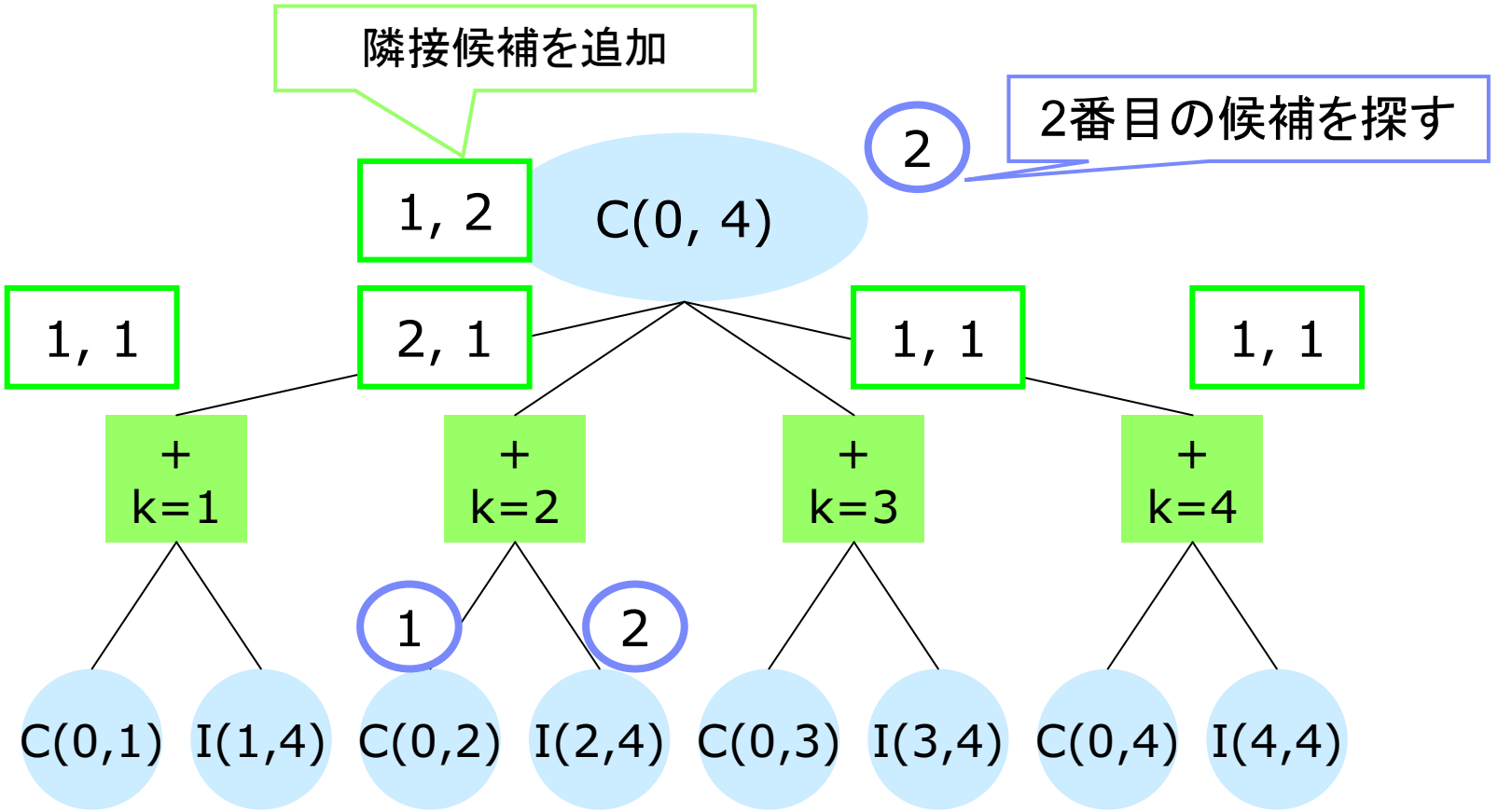
CKYもEisnerも足し算の塊 (1/2)

- $C(s, e) = \max_k(C(s, k) + I(k, e))$



CKYもEisnerも足し算の塊 (2/2)

- $C(s, e) = \max_k(C(s, k) + I(k, e))$



先の整列済み配列の足し算を再帰呼び出ししている

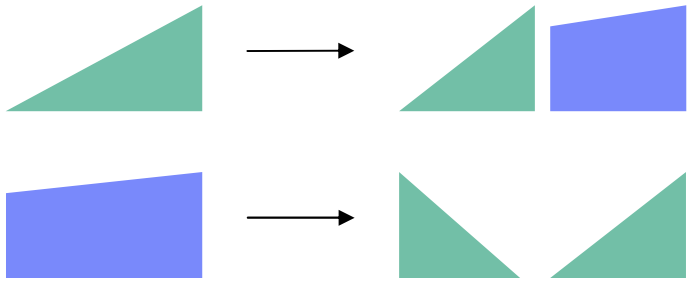
Higher order Eisner

- 以上は 1st order Eisner と呼ばれる
  - 1st order とは，係り元と係り先しか見ない
- より多くの依存関係を同時に見たくなる
  - 2nd order
    - 同じ係り先の兄弟
  - 3rd order
    - 祖父と親と子，あるいは親と子の兄弟
  - いずれも多項式時間で実現できる

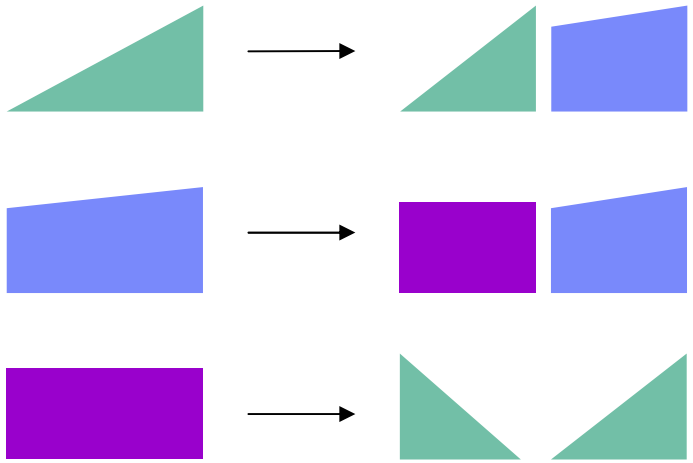
2nd order Eisner

- 兄弟に関する制約や素性を追加したい
  - –例えば同じ格助詞を持つ文節は兄弟にしたくない
- 以下のノードを追加すると解決する

1st order

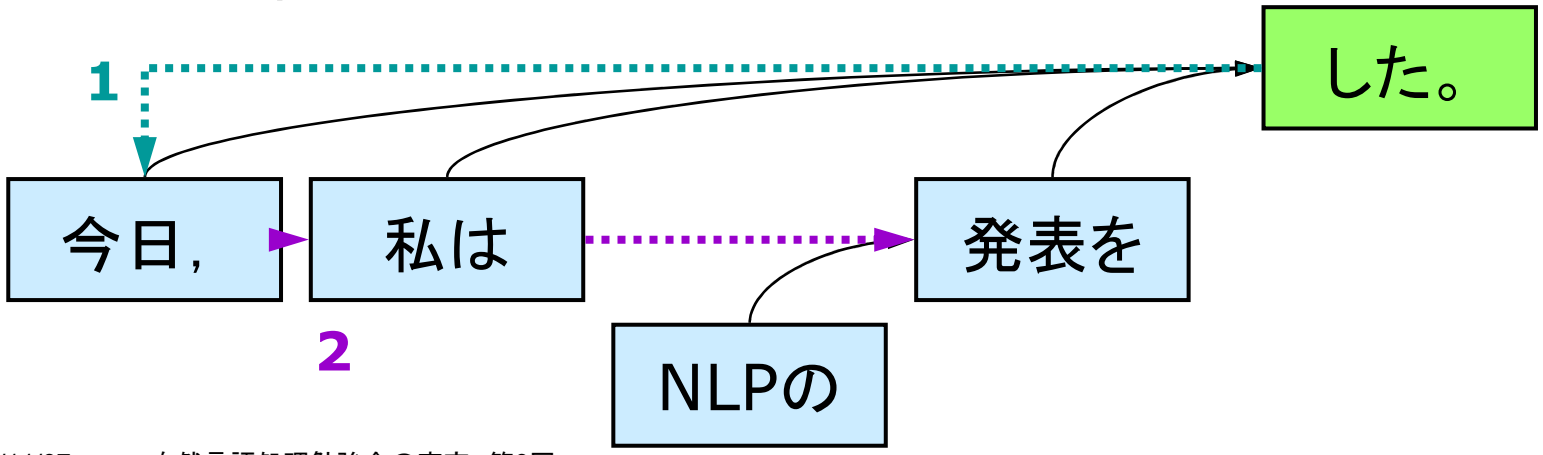


2nd order

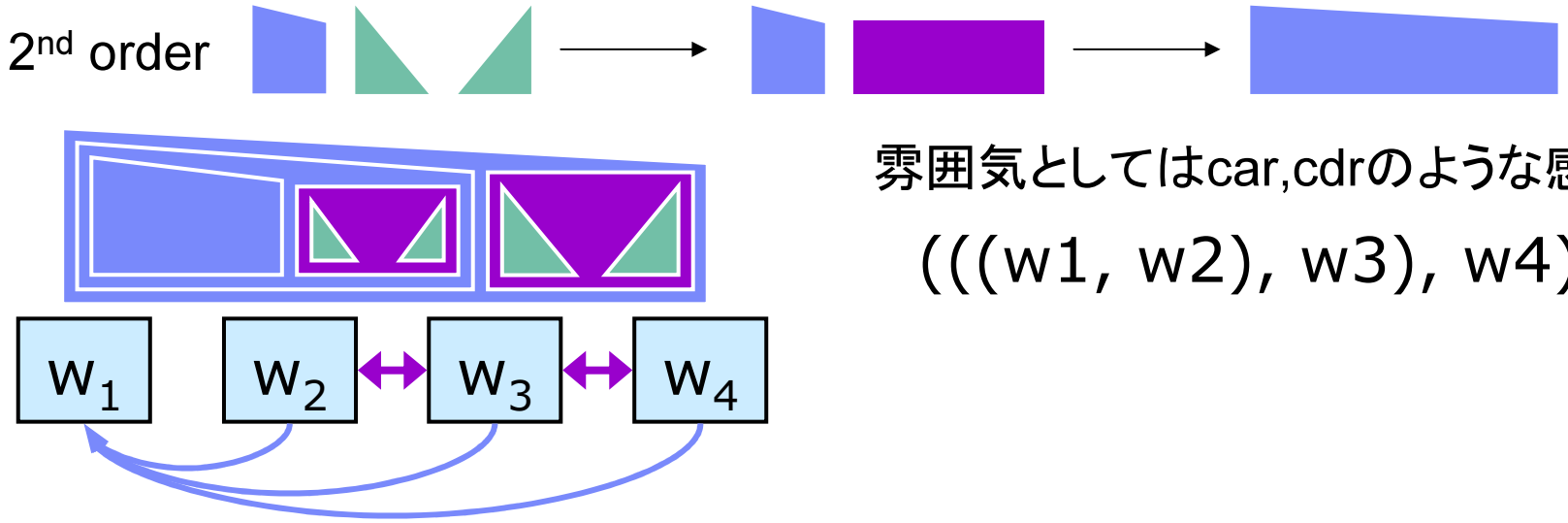


2nd order Eisnerを直感的に理解する

- 木をたどる操作には2種類ある
  1. 子ノードに遷移する
  2. 兄弟ノードを遷移する
- 1st order は，いわば2を1で無理やり表現したため，兄弟間の素性を設計できない
- 2nd order では2を直接表現しているため，兄弟間の素性を設計できる

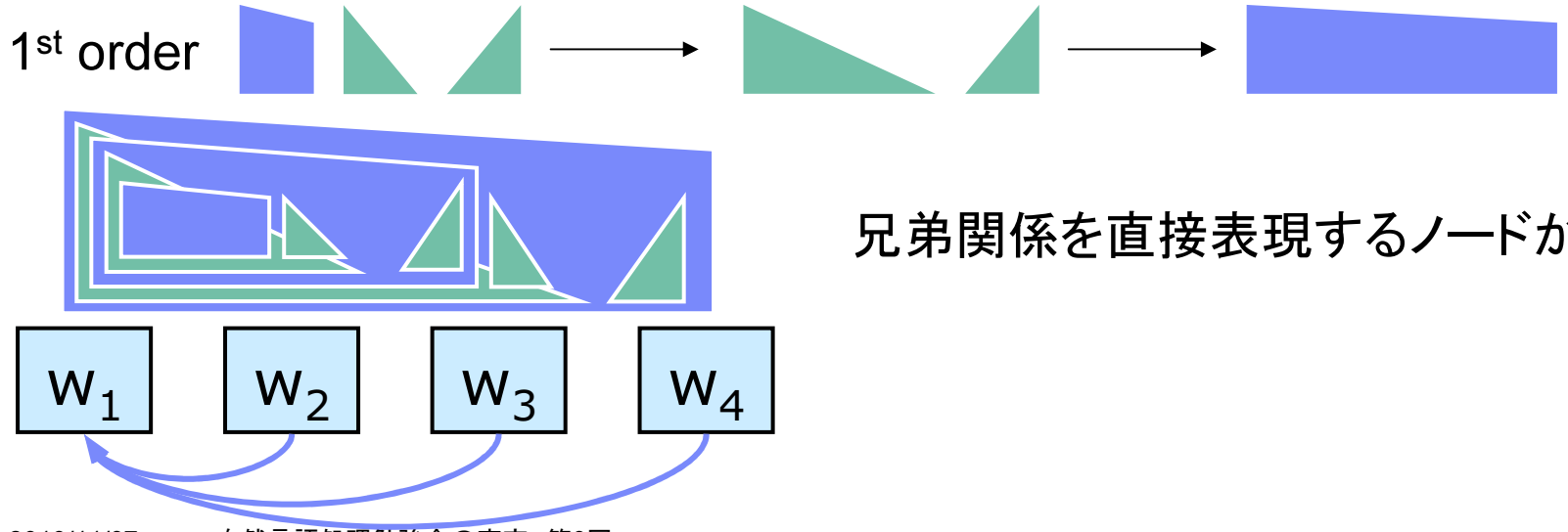


2nd order Eisnerを絵で理解する



雰囲気としてはcar,cdrのような感じ

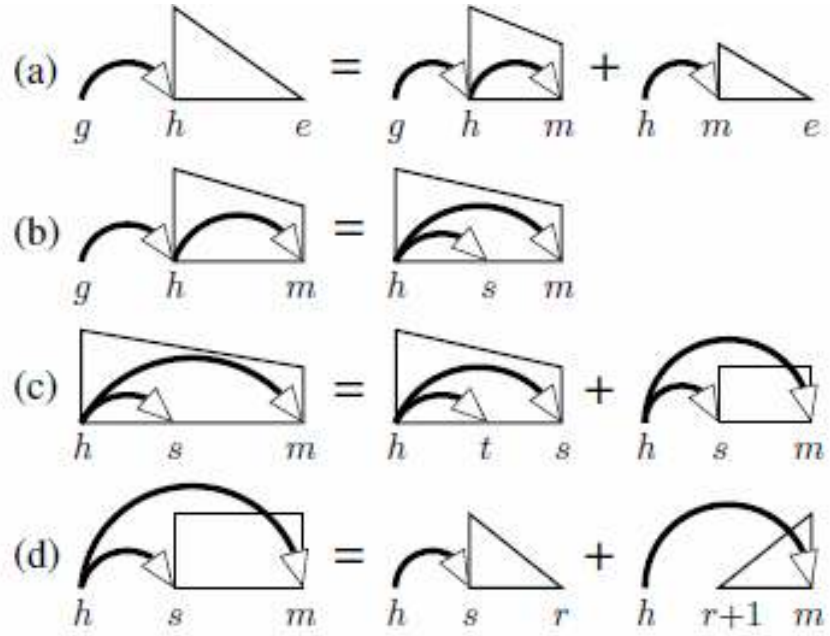
(((w1, w2), w3), w4)



兄弟関係を直接表現するノードがない

3rd order Eisner [Koo&Colins2010]

- 親子だけではなくて孫も同時に見たい
  - –uni-gramからbi-gramにする感覚
- 親子と同時に子の兄弟も見たい
  - –親と兄弟の3項関係



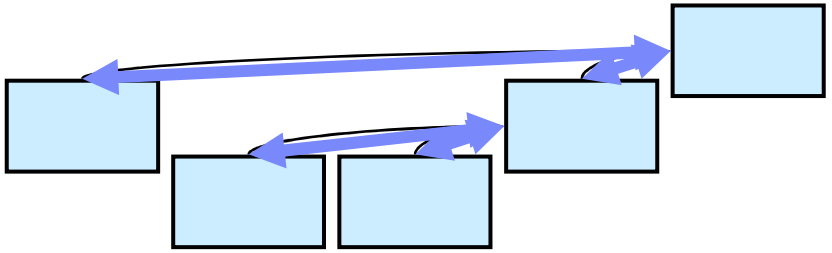
祖父を見ている

親と兄弟を同時に見ている

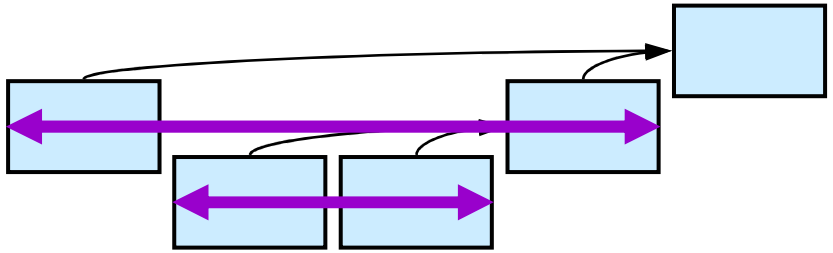
T. Koo and M. Collins, Efficient third-order dependency parsers. ACL 2010.

1st, 2nd, 3rd の比較

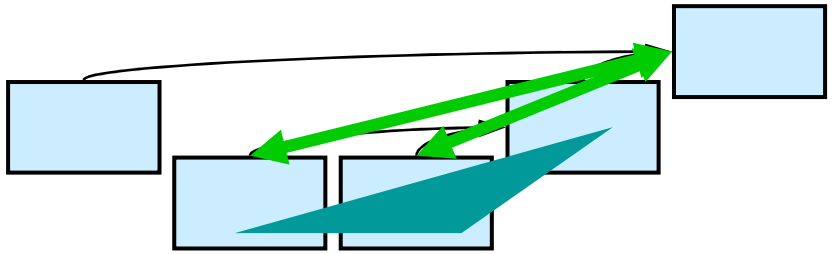
1st order
親子のみ



2nd order
兄弟を追加



3rd order
祖父，親，孫を追加
親，兄弟を追加



表現できる素性がだんだん増えている

他の手法

簡単な紹介だけ

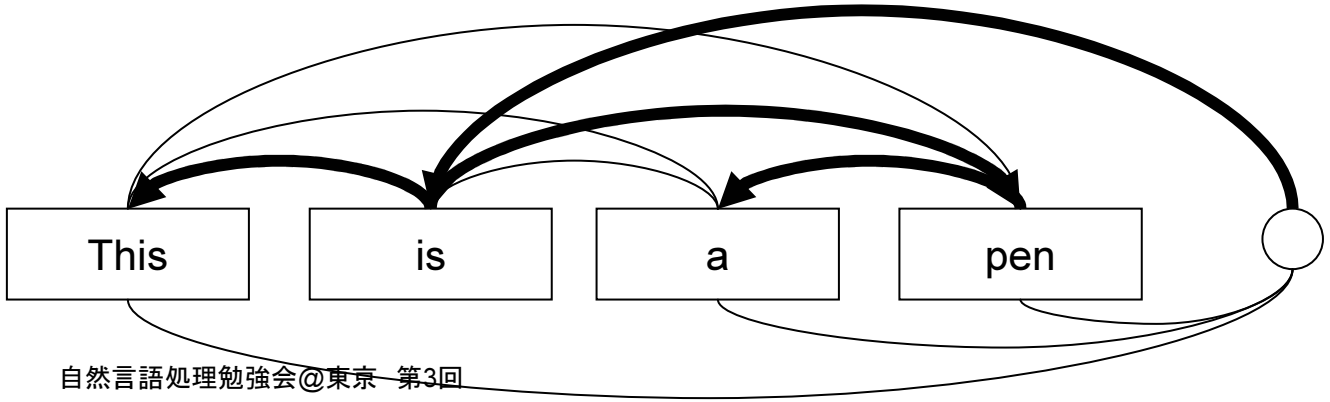
- MST
- ILP

MSTパーザー

- 最大全域木（Maximum Spanning Tree）アルゴリズムを使う
- 詳しく紹介しないが，現在主流の手法の一つ

MSTパーザーの特徴 [McDonald05]

- 最大全域木（Maximum Spanning Tree）アルゴリズムを使う
- 利点
  - 大域最適である
  - 解析時の計算量が$O(n^2)$と意外と小さい
  - 交差を自然に解いてくれる
- 欠点
  - 直接の係り関係しか使えない
  - 2^{nd} order以上になるとNP hard



ILP (Integer Linear Programming)パーザーの特徴 [Riedel&Clarke06, Martins+09]

- MSTパーザーのスコアを線形の目的関数にする
- 「木である」条件をがんばって整数の線形式で書き表す
- 線形式の制約による，線形目的関数の最大化問題

線形整数計画問題で解ける！

- 線形式で書けさえすれば，大域的なスコアを目的関数に入れたり，大域的な制約を入れられる

目次

- 係り受け解析とは何か
  - 用語の説明
  - タスクの説明
- 手法の紹介
  - Shift-reduce法
  - Eisner法
  - その他の手法
- まとめ

おさらい

- 係り受け解析とは？
  - 単語の係り先を当てる問題
  - 構造を当てる問題なので，候補がたくさんあってタイヘン
- 主な手法は2つの派閥
  - 局所最適だが大域素性のTransition-based
  - 大域最適だが局所素性のGraph-based
- 2つの手法を紹介した
  - Shift-Reduce法はスタックを使った方法
  - Eisner法は動的計画法を使った方法

使ってみよう

- 日本語
  - KNP (@黒橋研)
    - http://www-lab25.kuee.kyoto-u.ac.jp/nl-resource/knp.html
    - ルールベース
  - CaboCha (工藤さん)
    - http://chasen.org/~taku/software/cabocha/
    - Cascaded Chunking Model
- 英語
  - MSTParser (R. McDonald)
    - http://sourceforge.net/projects/mstparser/
    - MST, Eisner
  - MaltParser (J. Nivre)
    - http://maltparser.org/
    - Shift-Reduce

ご清聴ありがとうございました

参考文献

- Shift-Reduce
  - [Nivre03] J. Nivre, **An Efficient Algorithm for Projective Dependency Parsing.** IWPT 003.
  - [Yamada&Matsumoto03] H. Yamada and Y. Matsumoto, **Statistical Dependency Analysis with Support Vector Machines.** IWPT 2003.
  - [Nivre04] J. Nivre, **Incrementality in Deterministic Dependency Parsing.** Workshop on Incremental Parsing 2004.
  - [Nivre09] J. Nivre, **Non-Projective Dependency Parsing in Expected Linear Time.** ACL-IJCNLP 2009.
- Eisner
  - [Eisner96] J. M. Eisner, **Three New Probabilistic Models for Dependency Parsing: An Exploration.** COLING 1996.
  - [Jimenez&Marzal01] V. Jimenez and A. Marzel, **Computation of the N best parse trees for weighted and stochastic context-free grammars.** Advances in Pattern Recognition.
  - [Koo&Colins10] T. Koo and M. Collins, **Efficient third-order dependency parsers**. ACL 2010.

参考文献

- MST
  - [McDonald+05] R. McDonald, F. Pereira, K. Ribarov and J. Hajic, **Non-projective Dependency Parsing using Spanning Tree Algorithms.** f HLT-EMNLP 2005.
- ILP
  - [Riedel&Clarke06] S. Riedel and J. Clarke, **Incremental Integer Linear Programming for Non-projective Dependency Parsing.** EMNLP 2006.
  - [Martins et.al.09] A. F. T. Martins, N. A. Smith and E. P. Xing, **Concise Integer Linear Programming Formulations for Dependency Parsing.** ACL-IJCNLP 2009.
- その他
  - [工藤&松本04] 工藤 拓, 松本 裕治. **相対的な係りやすさを考慮した日本語係り受け解析モデル**. SIGNL-162, 2004.