2017年度III期

自然言語処理
（Natural Language Processing）

2017/10/26

Hiroyuki Shindo

目次

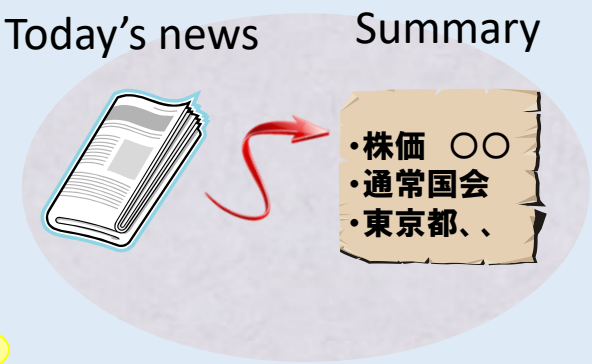
Advanced topics in natural language processing

- 自然言語処理の概要
- 構文解析
  - 特に依存構造解析のアルゴリズム
- 意味解析
  - 述語項構造解析

自然言語処理

機械翻訳

自動要約



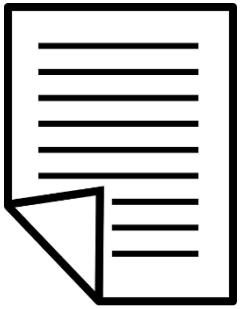
自動誤り訂正

He have a pen.

has?

解析，編集，生成

テキストデータ
（英語，日本語，etc）



自然言語生成

Google翻訳の品質が大幅に向上

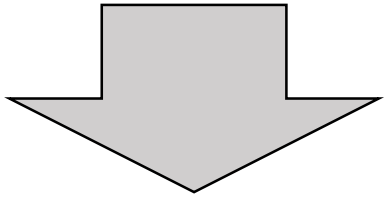
→　ディープラーニングに基づく機械翻訳によって実現

自然言語生成

自動文法誤り訂正

IN:

There is no a doubt, tracking system has brought many benefits in this information age.



OUT:

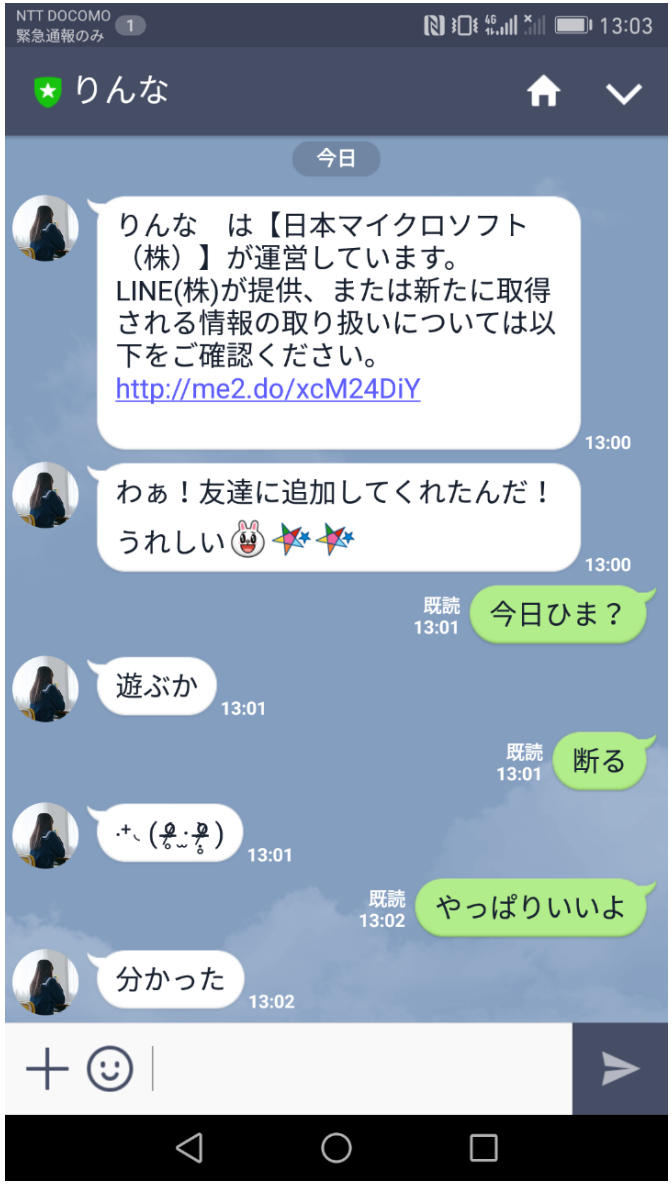
There is no ~~a~~ doubt, tracking *systems* *have* *~~system~~* *~~has~~* *brought many benefits in this information age.*

自然言語生成

自動文法誤り訂正

自然言語生成

対話ボット



http://www.rinna.jp/

自然言語処理

生成

言葉の文法や
意味を計算

- 形態素解析（単語分割＋品詞付与）
- 構文解析
- 意味解析

解析

テキストデータ
（英語，日本語，etc）

言葉の意味を計算する

- 質問に答えるコンピュータ
- 翻訳をするコンピュータ（機械翻訳）
- 言葉で書かれた情報を探し出すコンピュータ

実際には，これらを実現する基礎技術の研究が重要

- 文の構造（構文）がわからなければならない
- 動詞の主語や目的語がわからなければならない
- 言語を解析するための辞書や知識をコンピュータに持たせなければならない

自然言語処理における様々な解析

文の文法や意味を計算機で処理する

- 形態素解析 （単語分かち書き，品詞付与）
- 構文解析 （依存構造解析，句構造解析）
- 複単語表現解析 （イディオム，慣用句）
- 並列句解析 （A, B, and C）
- 意味解析 （述語項構造解析）
- 共参照解析

形態素解析(日本語)

私は新しく建った図書館へ行った

私	は	新しく	建っ	た	図書館	へ	行っ	た
代名詞	助詞	形容詞	動詞	助動詞	名詞	助詞	動詞	助動詞
		連用形	連用形	基本形			連用形	基本形
		新しい	建つ				行く	

日本語の形態素解析ツール: Mecab

形態素解析(日本語)

品詞(ひんし、英: parts of speech[1])は、単語を文法的な機能や形態などによって分類したもの。

日本語の品詞 [編集]

「日本語#品詞体系」も参照

日本語においては、さまざまな品詞分類が試みられている。語か、活用の有無、活用の形態などによって以下のように分

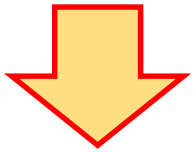
- 自立語 – 単独で文節を構成できる品詞
  - 活用するもの(特に用言と言う)
    - 動詞
    - 形容詞
    - 形容動詞(学校文法では品詞として立てているが
  - 活用しないもの(特に体言と言う)
    - 名詞
      - 代名詞(学校文法では名詞の一つとされてい
      - 数詞(学校文法では名詞の一つとされている
    - 連体詞
    - 副詞
    - 接続詞
    - 感動詞
- 付属語 – 単独で文節を構成できない品詞
  - 活用するもの
    - 助動詞
  - 活用しないもの
    - 助詞

https://ja.wikipedia.org/wiki/%E5%93%81%E8%A9%9E

形態素解析(英語)

入力

The auto maker sold 1000 cars last year.



出力

The auto maker sold 1000 cars last year.

DT N N V CD N JJ N

形態素解析(英語)

英語の品詞 [編集]

英語の品詞としては、次の8品詞とする説が一般的である。過去には4品詞とする説などもあった。

- 名詞 (Noun, *n.*; Substantive, *subst.*)
- 代名詞 (Pronoun, *pron.*)
- 形容詞 (Adjective, *adj.*または*a.*)
  - 冠詞 (Article, *art.*) (広い意味では形容詞に含める)
  - 数詞 (Numeral, *num.*) (広い意味では形容詞に含める)
- 動詞 (Verb, *vb.*, *v.*)
  - 代動詞 (Auxiliary verb, *aux. v.*, e.g. do, does, did) (広い意味では動詞に含める)
  - 助動詞 (Modal verb) (広い意味では動詞に含める)
- 副詞 (Adverb, *adv.*)
- 前置詞 (Preposition, *prep.*)
- 接続詞 (Conjunction, *conj.*)
- 感動詞(間投詞) (Interjection, *interj.*または*int.*)

8品詞とは別に、次のような用語もよく使われる。

- 関係詞 (Relatives) (代名詞と接続詞の機能をもつ関係代名詞と、副詞と接続詞の機能をもつ関係副詞がある)
- 疑問詞 (Interrogatives)

また、近年、生成文法の観点から、**限定詞** (Determiner, determinative) 概念も有力となっている。

https://ja.wikipedia.org/wiki/%E5%93%81%E8%A9%9E

固有表現解析

固有名詞（人名，会社名，場所名）の範囲と種類を当てる

入力

He went to Tokyo Tower.

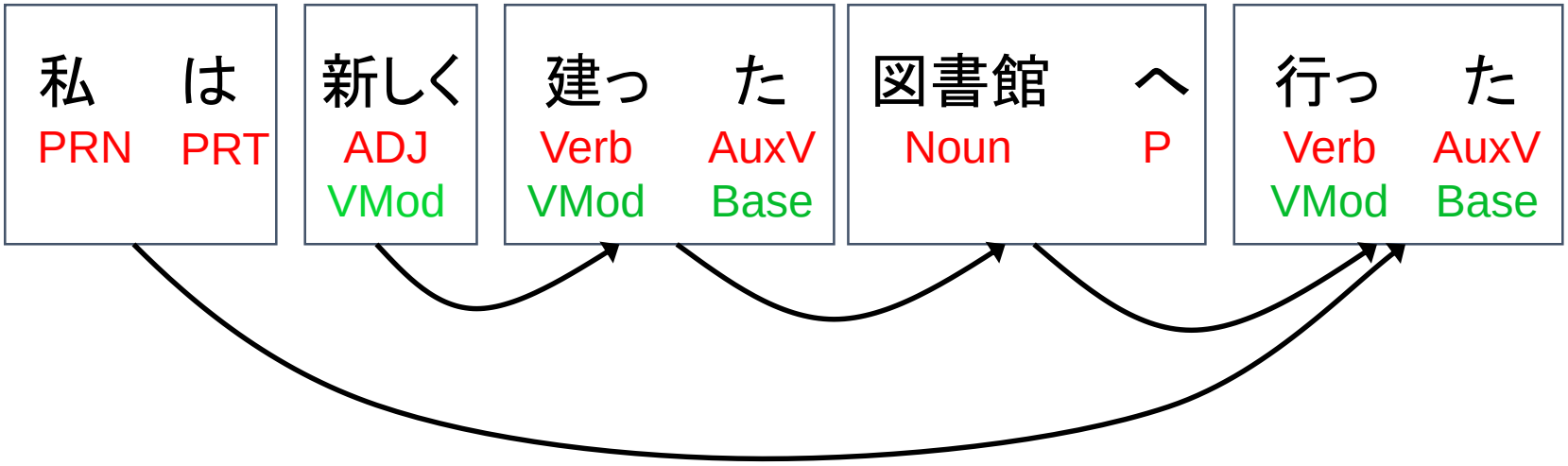
出力

Location

He went to Tokyo Tower.

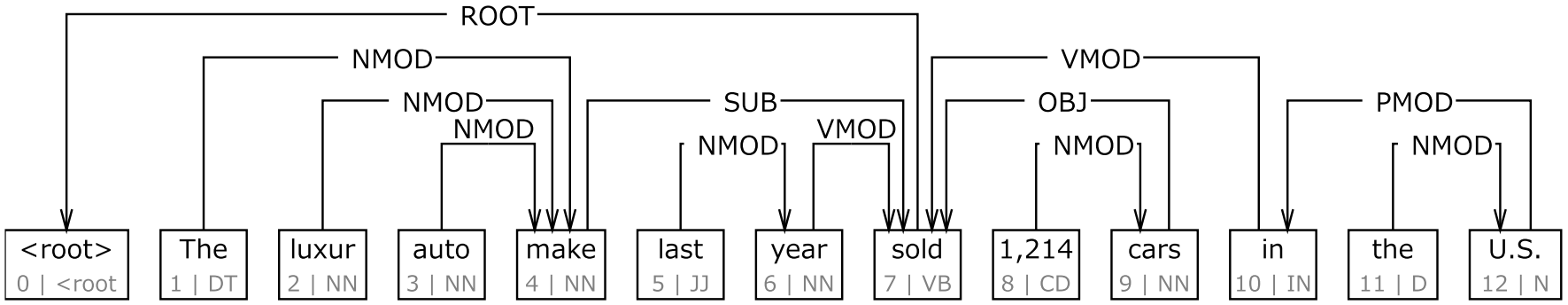
構文解析

係り受け解析


私 は
PRN PRT
新しく
ADJ
VMod
建っ た
Verb AuxV
VMod Base
図書館 へ
Noun P
行っ た
Verb AuxV
VMod Base

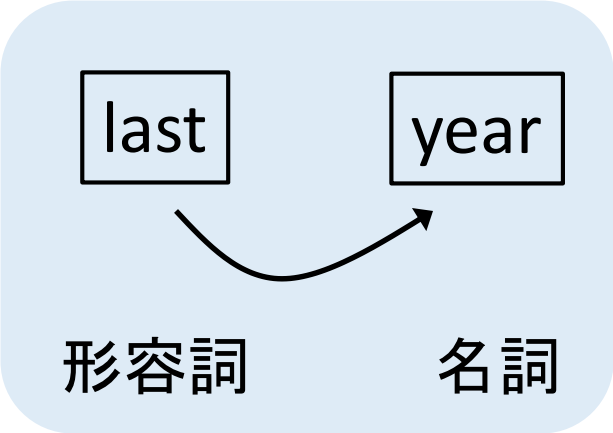
構文解析

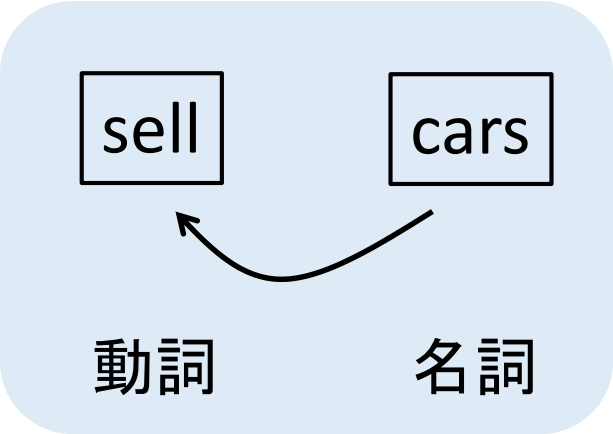
係り受け解析（英語の場合）



構文解析

係り受け関係とは





どの単語も，必ず1つの単語に依存する

構文解析

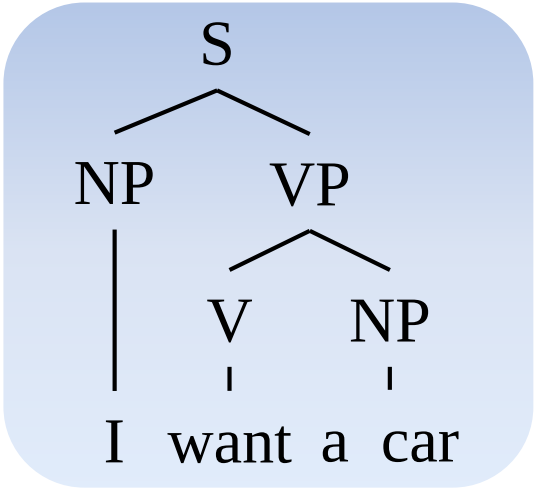
句構造解析

入力

I want a car.

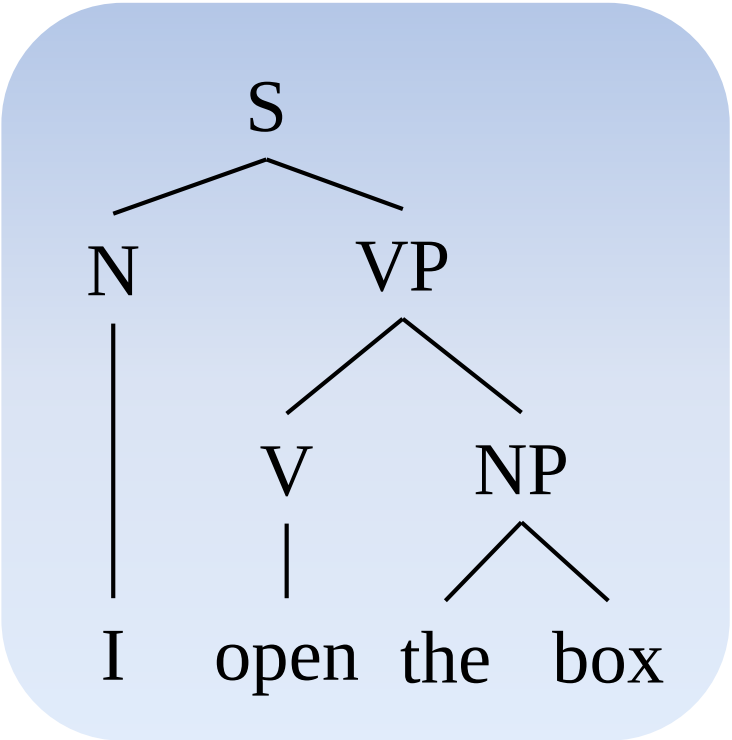
句構造

出力

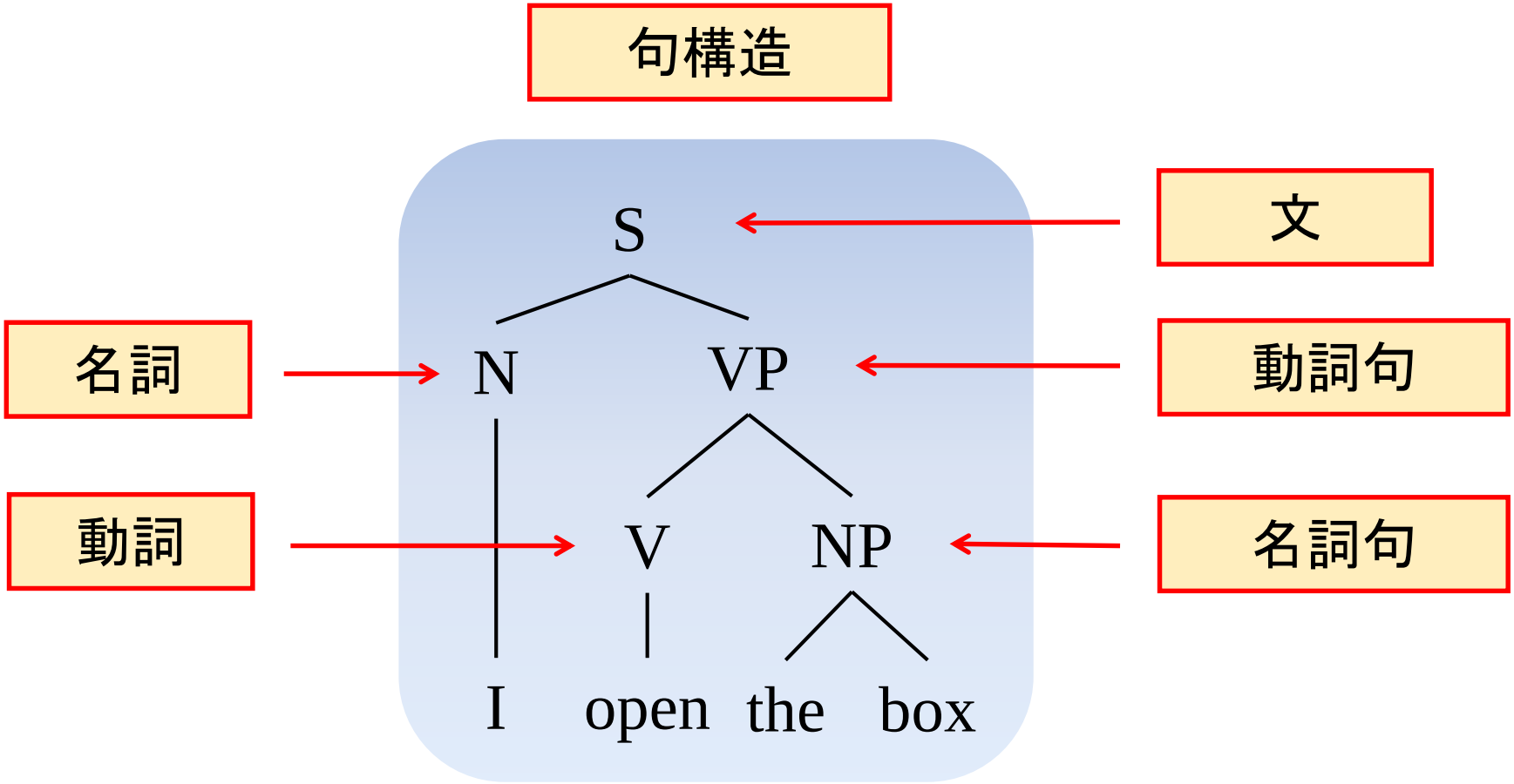


構文解析

句構造



構文解析



単語がどのように組み合わさって文になるかを表す

構文解析はどう役に立つのか？

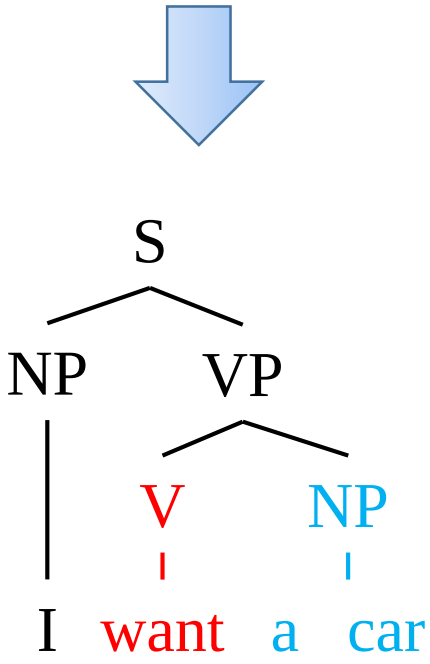
例：英語から日本語への機械翻訳

I want a car -> 私は 車が 欲しい

構文解析はどう役に立つのか？

例：英語から日本語への機械翻訳

I want a car -> 私は 車が 欲しい

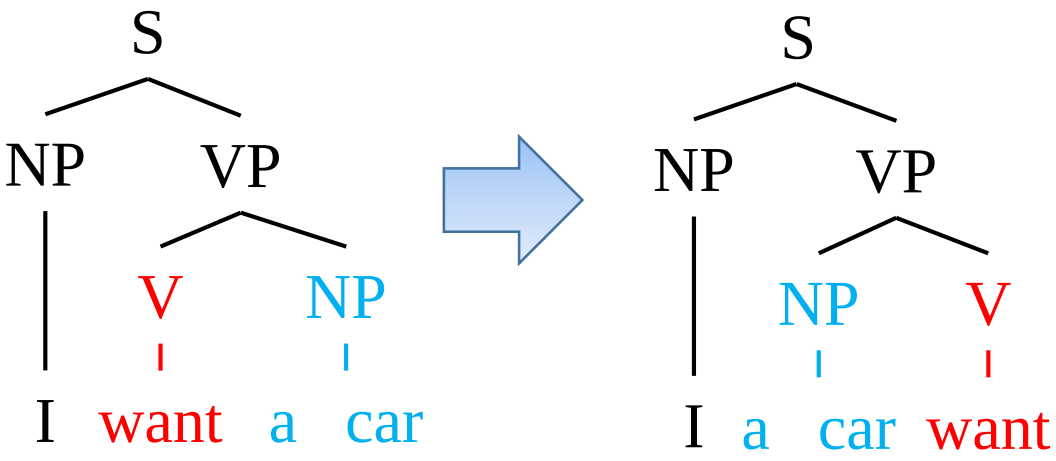


構文解析はどう役に立つのか？

例：英語から日本語への機械翻訳

I want a car -> 私は 車が 欲しい

☺ 動詞 (V) と 目的語 (NP) を並べ替える

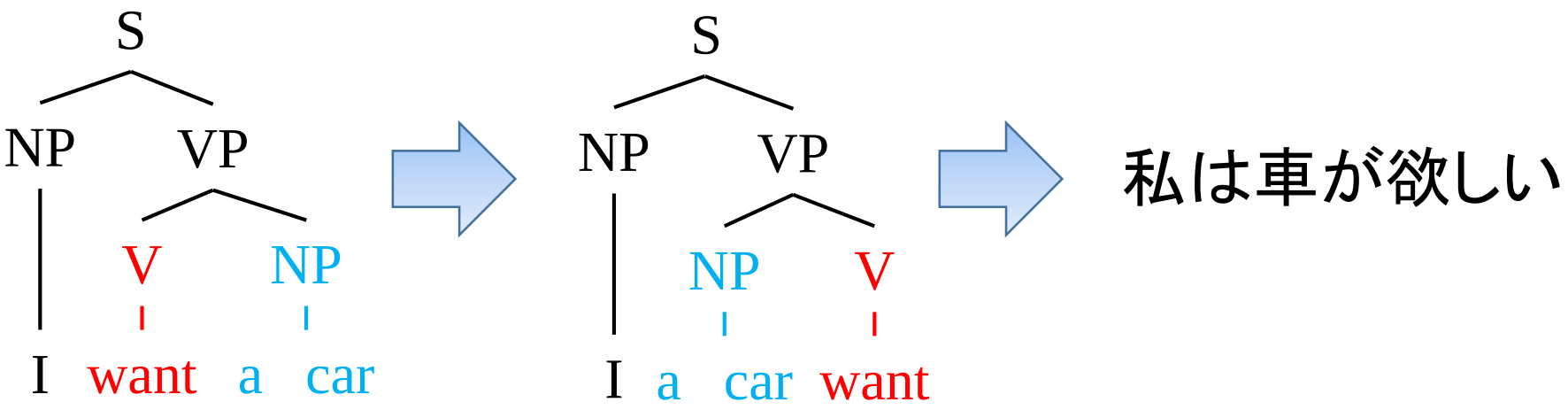


構文解析はどう役に立つのか？

例：英語から日本語への機械翻訳

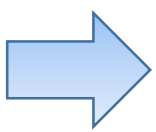
I want a car -> 私は 車が 欲しい

☺ 動詞 (V) と 目的語 (NP) を並べ替える

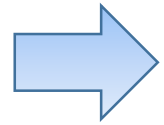


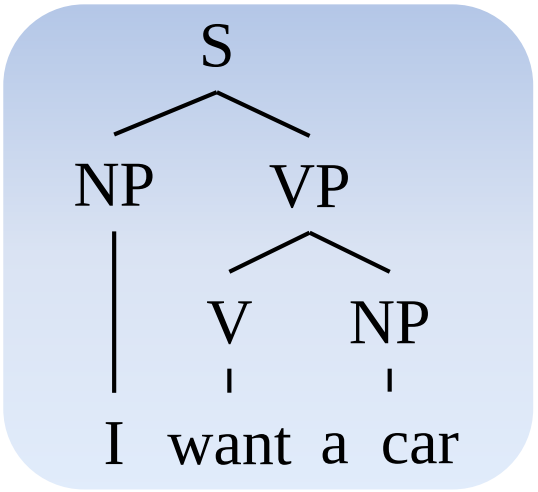
解析とは曖昧性解消である

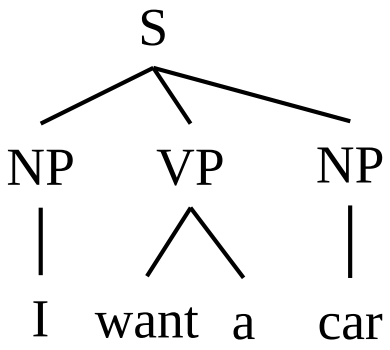
I want a car

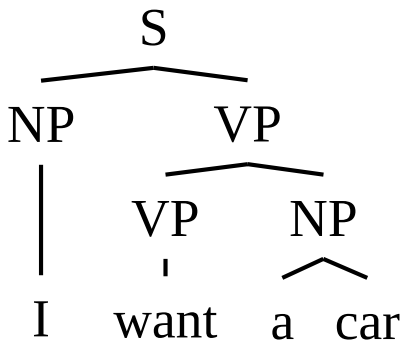


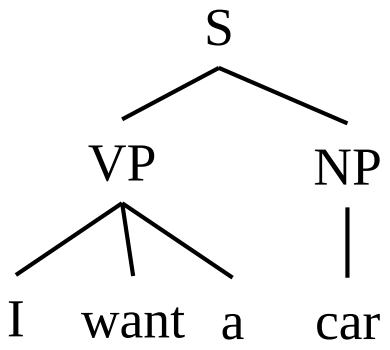
解析





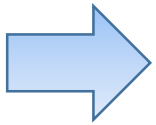




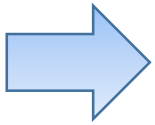


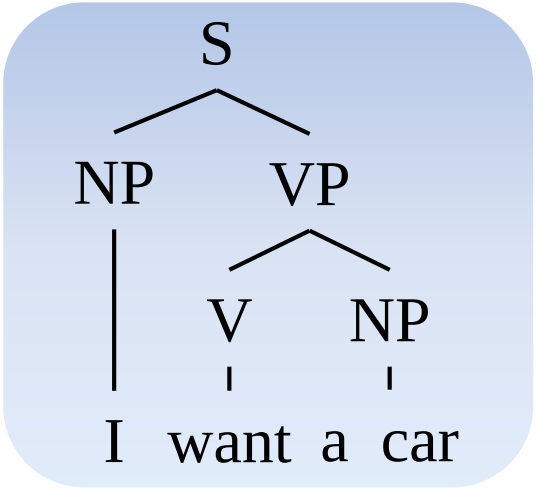
解析とは曖昧性解消である

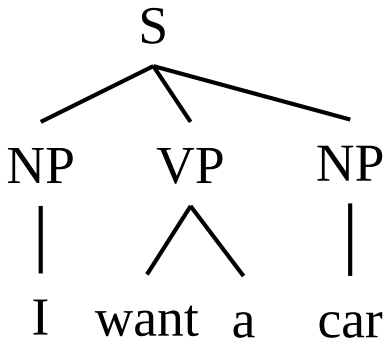
I want a car



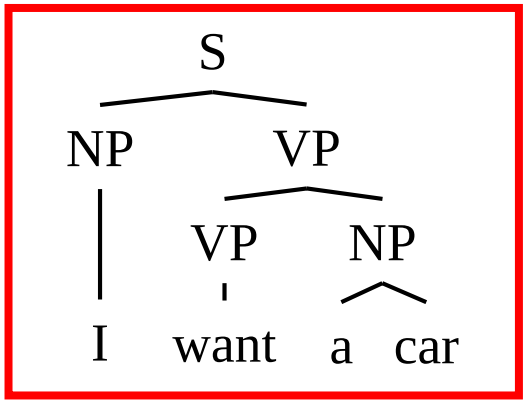
解析



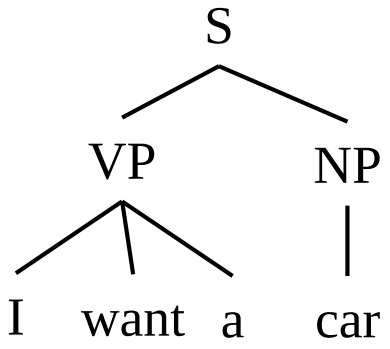




p : 0.001



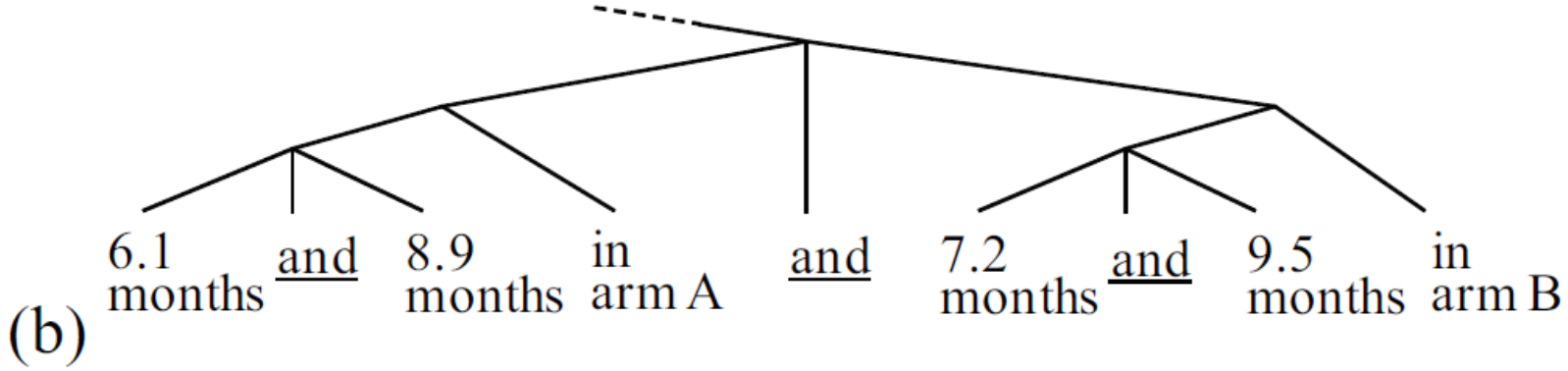
p : 0.3



p : 0.02

並列句解析

and, or が何と何を並列に比べているか？



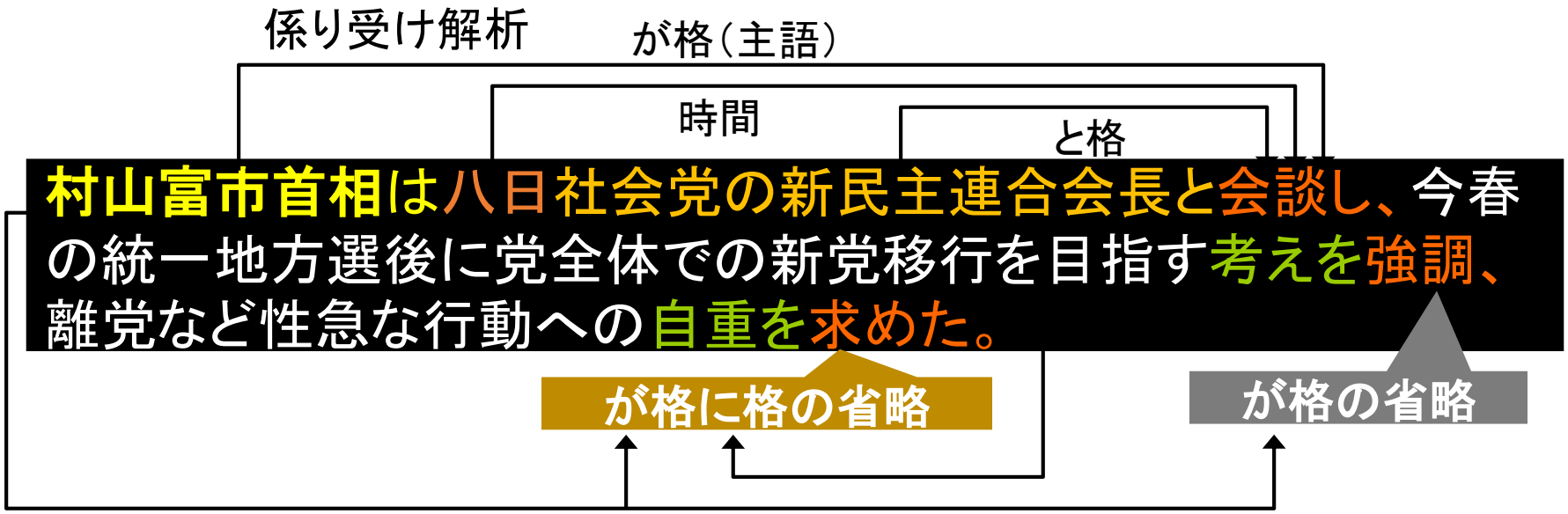
6.1 months　and　8.9 months

7.2 months　and　9.5 months

6.1 months … in arm A　and　7.2 months … in arm B

意味解析

述語項構造解析

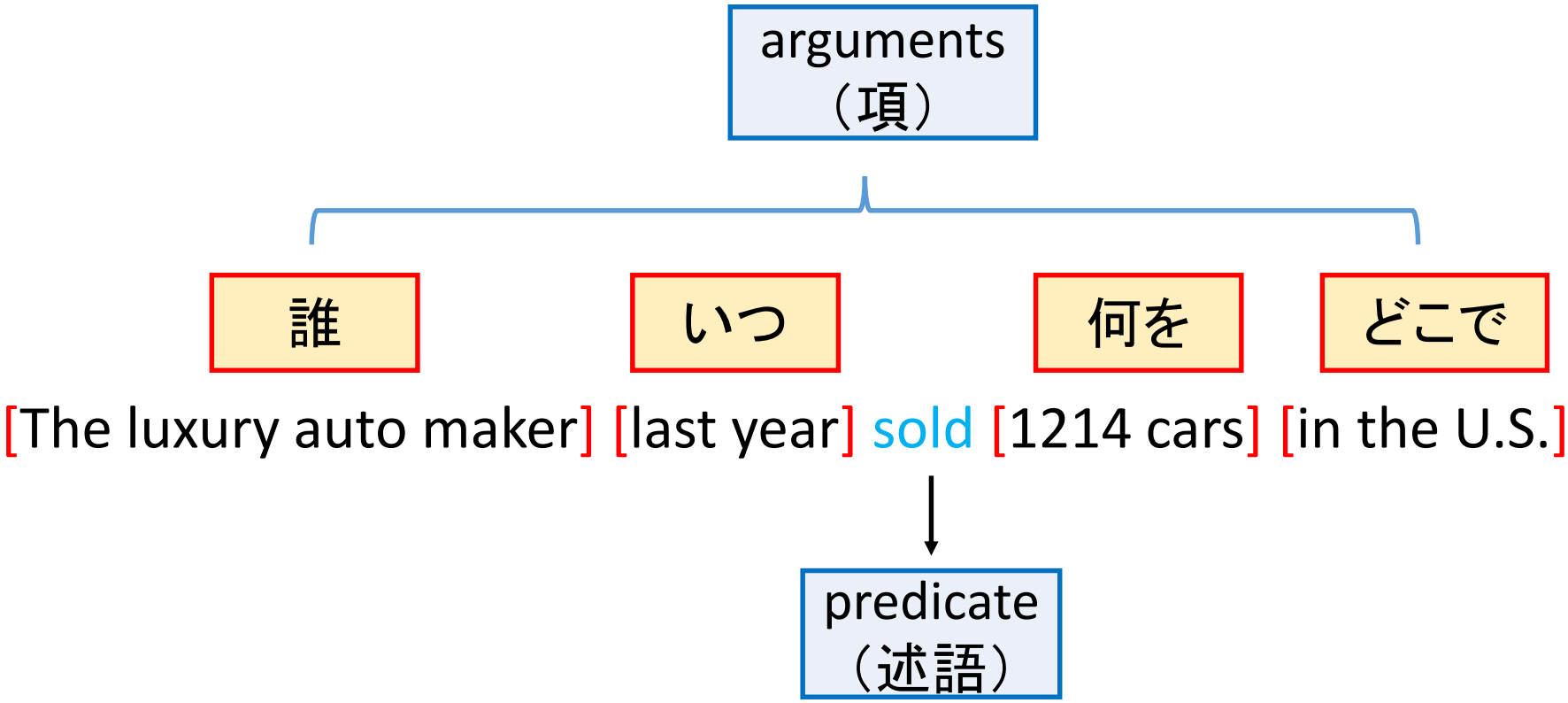


係り受けだけではわからない省略された語の推定

意味解析

述語項構造

→ 文の意味表現の基本



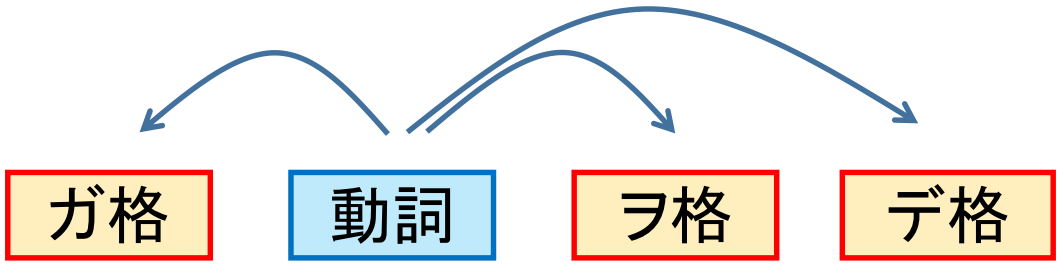
意味解析

述語項構造

→ 文の意味表現の基本

The Case for Case [Fillmore '68]

格:
- ガ格
- ヲ格
- ニ格
- デ格



文の意味を，動詞を中心とした格構造によって理解する

格文法

意味解析

誰が　いつ　どこで　何を　した？

Syntactic variations

TEMP HITTER THING HIT INSTRUMENT

Yesterday, Kristina hit Scott with a baseball

- Scott was hit by Kristina yesterday with a baseball
- Yesterday, Scott was hit with a baseball by Kristina
- With a baseball, Kristina hit Scott yesterday
- Yesterday Scott was hit by Kristina with a baseball
- Kristina hit Scott with a baseball yesterday

Example from (Yih & Toutanova, 2006)

意味解析

誰が　いつ　どこで　何を　した？

IBM appointed John.

John was appointed by IBM.

IBM’s appointment of John.

The appointment of John by IBM.

John is the current IBM appointee.

自然言語処理

生成

言葉の文法や
意味を計算

- 形態素解析（単語分割＋品詞付与）
- 構文解析
- 意味解析

解析

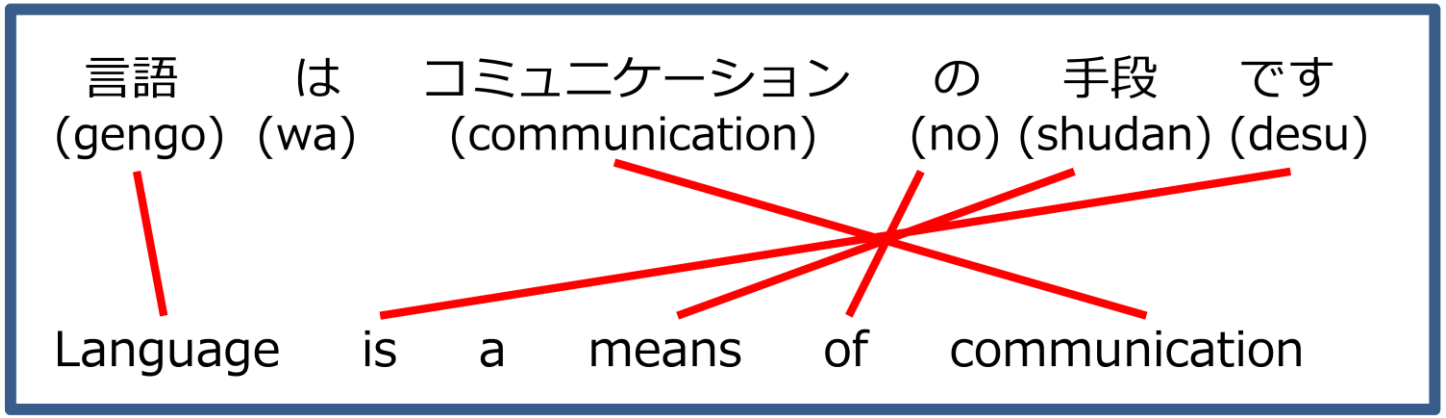
テキストデータ
（英語，日本語，etc）

機械翻訳

単語アライメント

source
J

target
E



翻訳テーブル

言語	language	0.52
コミュニケーション	communication	0.99
手段	means	0.24

自動要約

国連安全保障理事会は２０日、西アフリカ・マリ北部を制圧したイスラム過激派掃討のため軍事介入を認める決議を全会一致で採択した。周辺国で構成する西アフリカ諸国経済共同体（ＥＣＯＷＡＳ）による３３００人規模の部隊派遣を承認。混迷のマリ情勢は新たな局面に入る。
決議では、アフリカ国際マリ支援部隊（ＡＦＩＳＭＡ）に対し「必要なあらゆる手段の行使」を認めた。派遣部隊はまずマリ軍兵士の教育や作戦を支援する。派遣期間は１年。軍事行動の開始は来年秋以降になる見通し。
マリでは３月、首都バマコで反乱軍によるクーデターが発生。イスラム過激派が北部を制圧し、国土は事実上二分された。今月１０日には政府軍兵士らがディアラ暫定政府首相の身柄を拘束。首相は翌１１日に退陣を表明し、混乱が加速した。ロイター通信によると、ディアラ首相拘束はクーデターを主導したサノゴ大尉の指示。背景には首相とトラオレ暫定大統領らとの間の政治的対立があったとされ、トラオレ大統領は職務を継続している

国連安全保障理事会は２０日、西アフリカ・マリ北部を制圧したイスラム過激派掃討のため軍事介入を認める決議を全会一致で採択した。混迷のマリ情勢は新たな局面に入る。決議では、アフリカ国際マリ支援部隊（ＡＦＩＳＭＡ）に対し「必要なあらゆる手段の行使」を認めた。派遣部隊はまずマリ軍兵士の教育や作戦を支援する。

- 単一文章要約
- 複数文章要約
- 文圧縮

目次

Advanced topics in natural language processing

- 自然言語処理の概要
- 構文解析
  - 特に依存構造解析のアルゴリズム
- 意味解析
  - 述語項構造解析

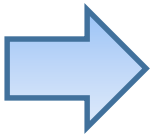
Penn Treebank

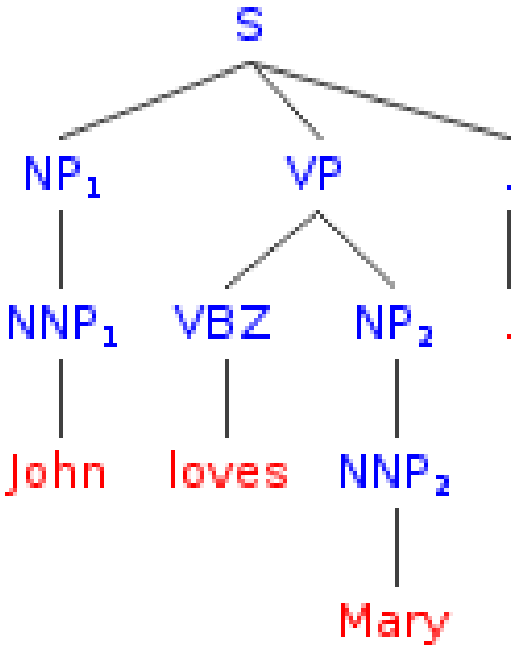
自然言語解析で用いられる標準的なデータセット

“Building a Large Annotated Corpus of English: **The Penn Treebank**”
[Marcus et al. 1993]

S-expression (S式)

```
(S  (NP  (NNP John))
    (VP  (VBZ loves)
         (NP  (NNP Mary)))
    (.  .))
```

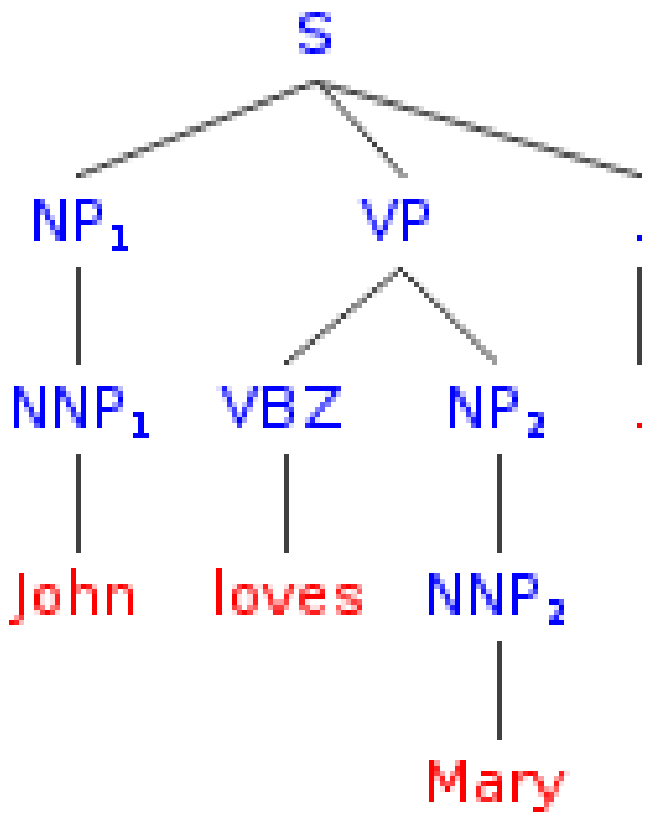




Penn Treebank

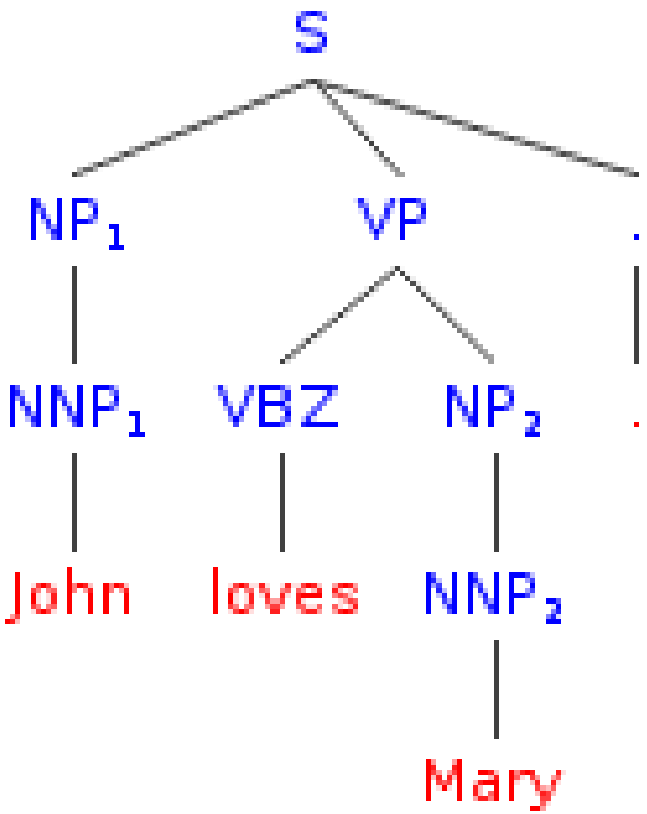
S-expression (S式)

```
(S  (NP  (NNP  John))
    (VP  (VBZ  loves)
         (NP  (NNP  Mary)))
    (.  .))
```



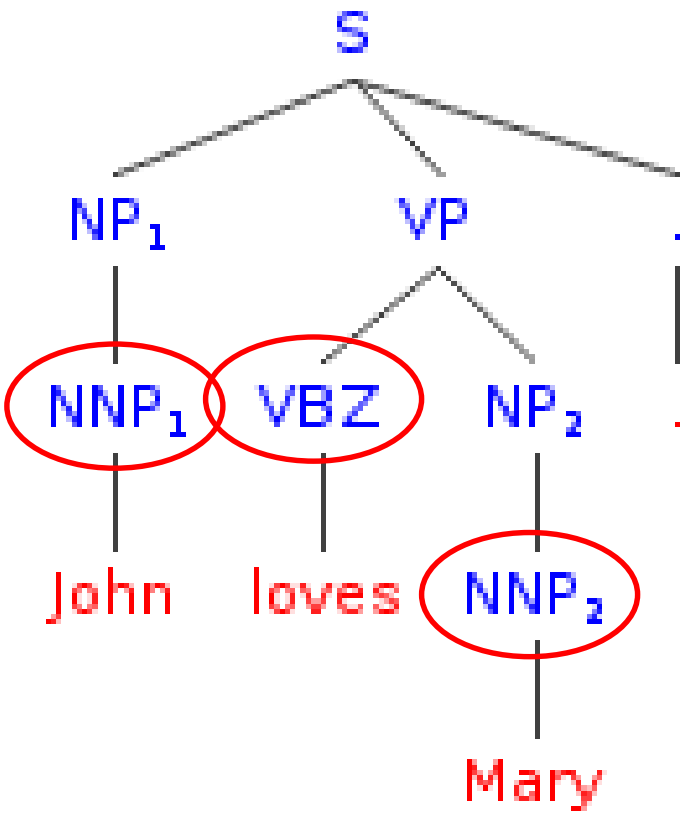
Penn Treebank

Syntax Tree (構文木)



Penn Treebank

Syntax Tree (構文木)

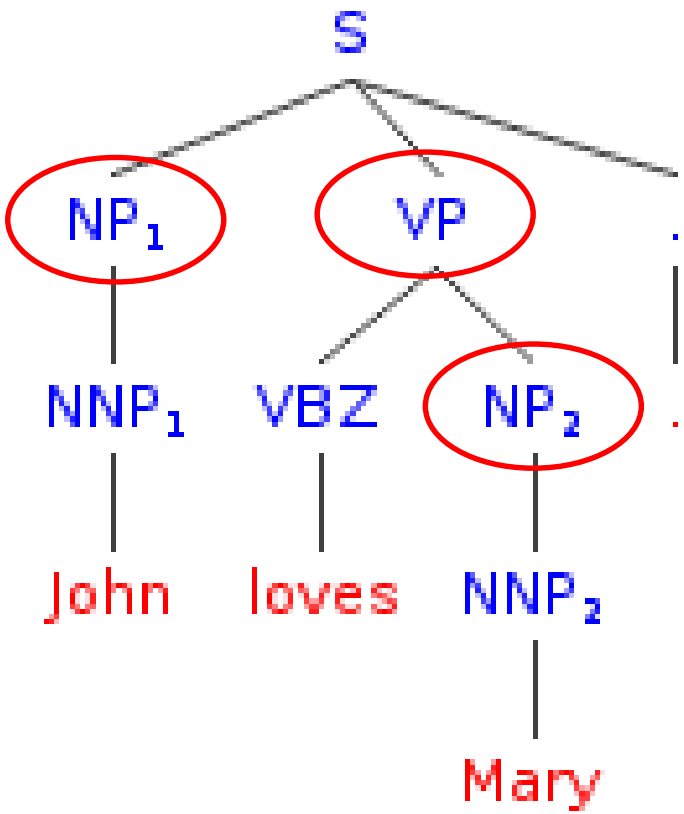


POS Tag
(品詞タグ)

NNP - Proper noun, singular
VBZ - Verb, 3rd person singular present

Penn Treebank

Syntax Tree (構文木)



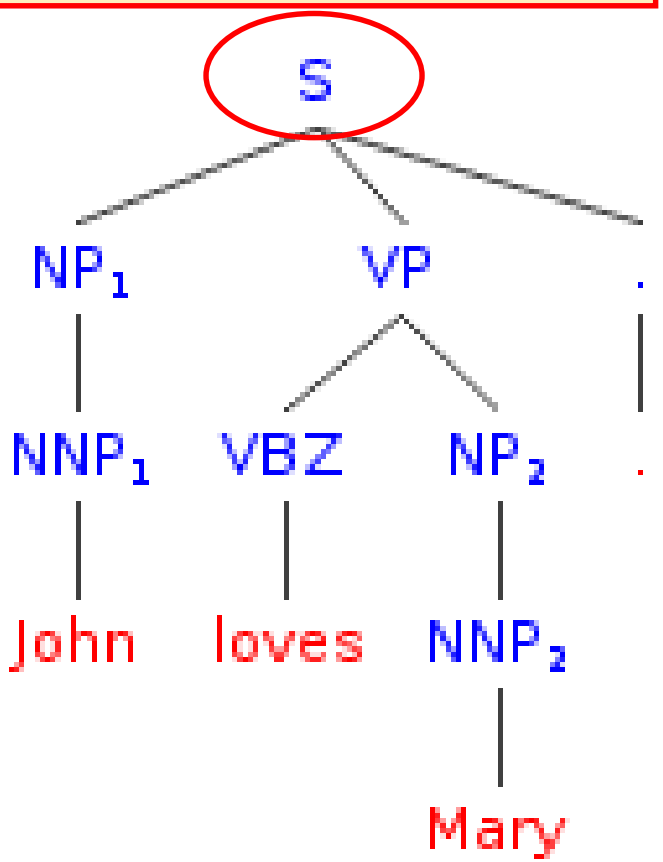
Phrase (句)

NP - Noun Phrase

VP - Verb Phrase

Penn Treebank

Syntax Tree (構文木)



Clause (節）

S - Simple declarative clause (平叙文)

Word Level Tags (POS Tag)

CC - Coordinating conjunction
CD - Cardinal number
DT - Determiner
EX - Existential there
FW - Foreign word
IN - Preposition or subordinating conjunction
JJ - Adjective
JJR - Adjective, comparative
JJS - Adjective, superlative
LS - List item marker
MD - Modal
NN - Noun, singular or mass
NNS - Noun, plural
NNP - Proper noun, singular
NNPS - Proper noun, plural
PDT - Predeterminer
POS - Possessive ending
PRP - Personal pronoun
PRP$ - Possessive pronoun
RB - Adverb
RBR - Adverb, comparative
RBS - Adverb, superlative
RP - Particle
SYM - Symbol
TO - to
UH - Interjection
VB - Verb, base form
VBD - Verb, past tense
VBG - Verb, gerund or present participle
VBN - Verb, past participle
VBP - Verb, non-3rd person singular present
VBZ - Verb, 3rd person singular present
WDT - Wh-determiner
WP - Wh-pronoun
WP$ - Possessive wh-pronoun
WRB - Wh-adverb

Phrase Level Tags

ADJP - Adjective Phrase
ADVP - Adverb Phrase
CONJP - Conjunction Phrase
FRAG - Fragment
INTJ - Interjection
LST - List marker
NAC - Not a Constituent
NP - Noun Phrase
NX - Used within certain complex NPs to mark the head of the NP
PP - Prepositional Phrase
PRN - Parenthetical
PRT - Particle
QP - Quantifier Phrase
RRC - Reduced Relative Clause
UCP - Unlike Coordinated Phrase
VP - Vereb Phrase
WHADJP - *Wh*-adjective Phrase
WHAVP - *Wh*-adverb Phrase. Introduces a clause with an NP gap
WHNP - *Wh*-noun Phrase
WHPP - *Wh*-prepositional Phrase
X – Unknown

Clause Level Tags

S - Simple declarative clause
SBAR - Clause introduced by a subordinating conjunction
SBARQ - Direct question introduced by a *wh*-word or a *wh*-phrase
SINV - Inverted declarative sentence
SQ - Inverted yes/no question

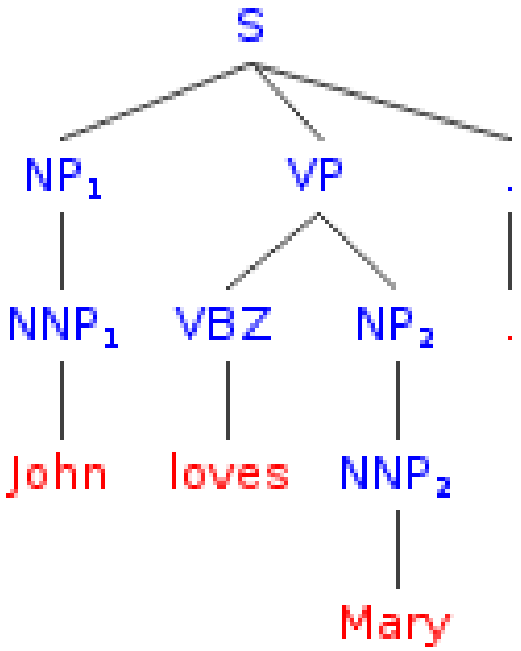
構文解析(句構造)

Input

John loves Mary .

Parse

Output



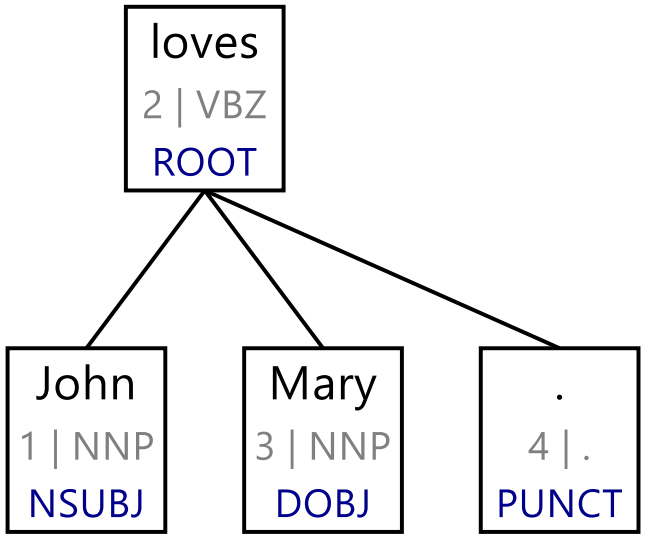
構文解析(依存構造)

Input

John loves Mary .

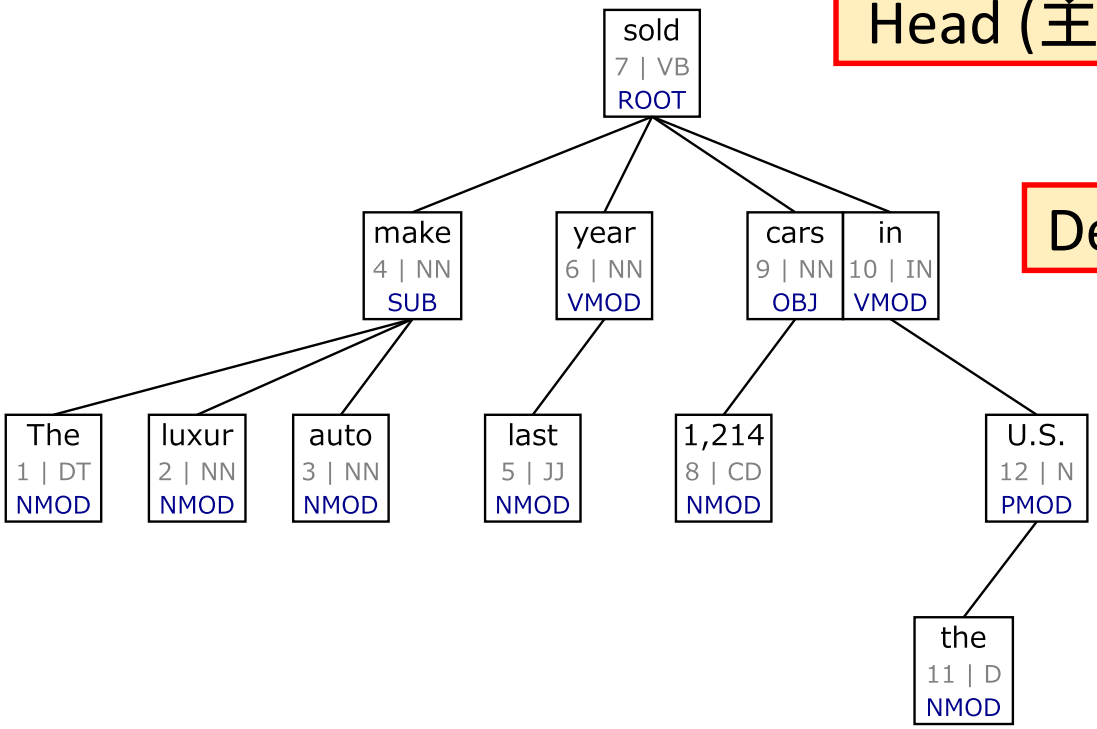
Parse

Output



依存構造

Tree View



Head (主辞, 主要部)

Dependent (従属部)

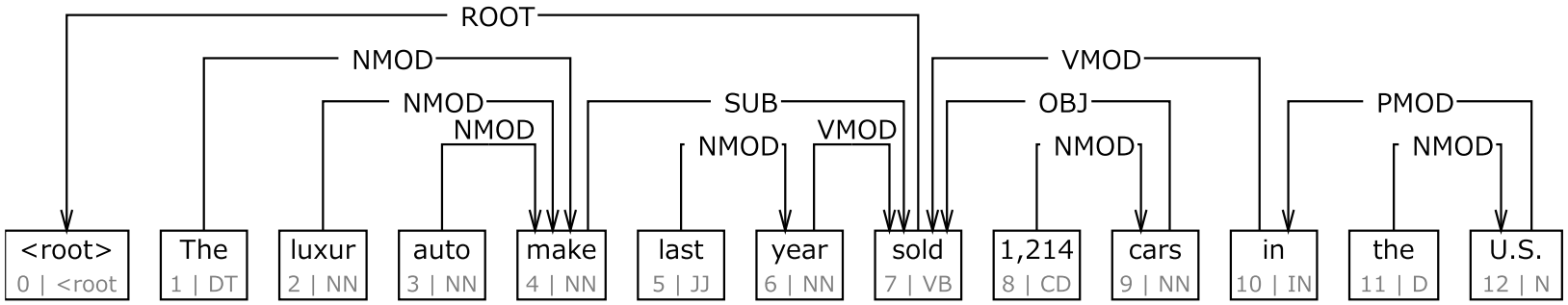
The luxury auto maker last year sold 1,214 cars in the U.S.

依存構造

Horizontal View

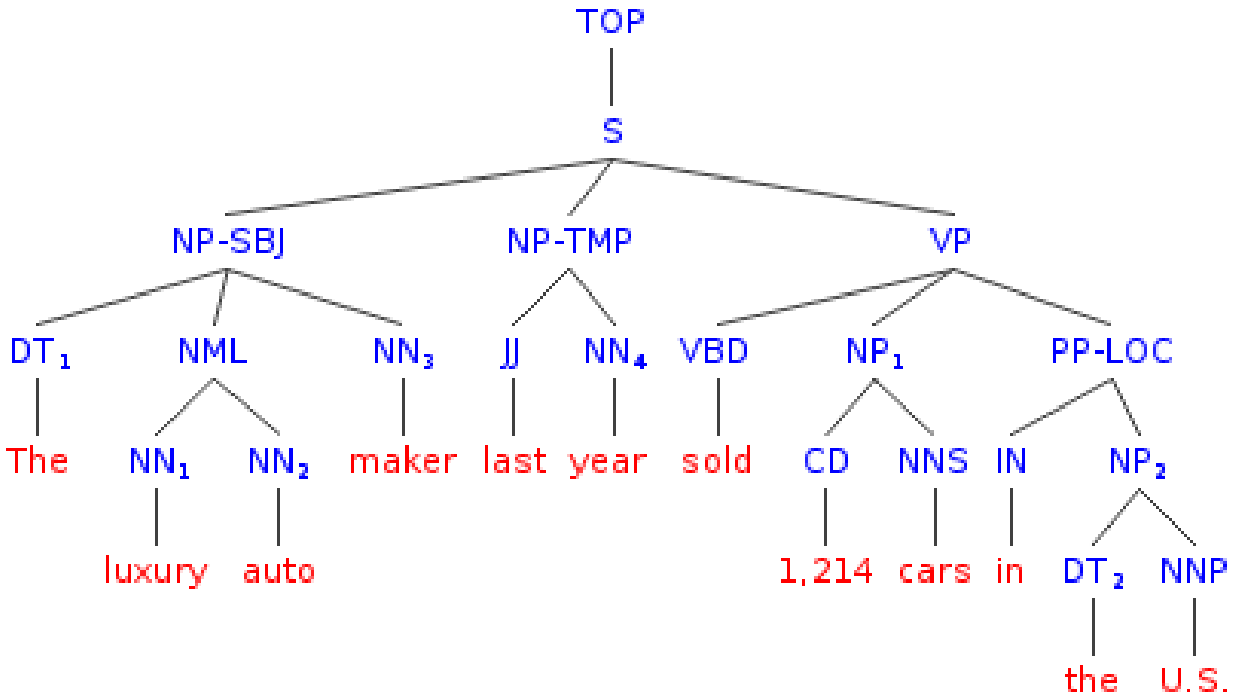
arc: head-dependent relations

arc-label: NMOD, SUB, OBJ, etc.



The luxury auto maker last year sold 1,214 cars in the U.S.

句構造



The luxury auto maker last year sold 1,214 cars in the U.S.

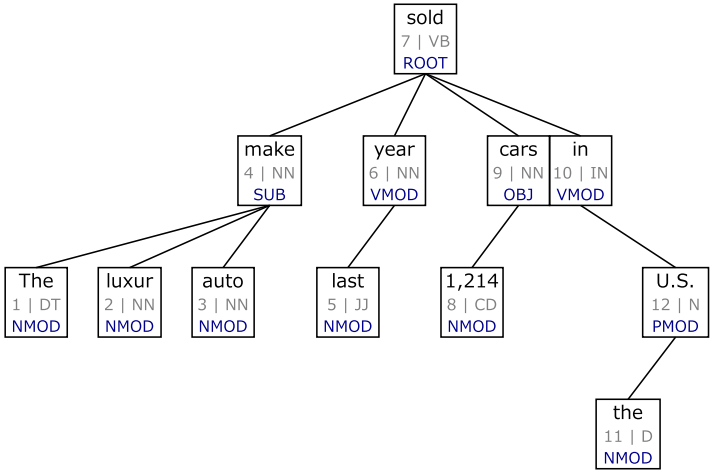
依存構造と句構造の比較

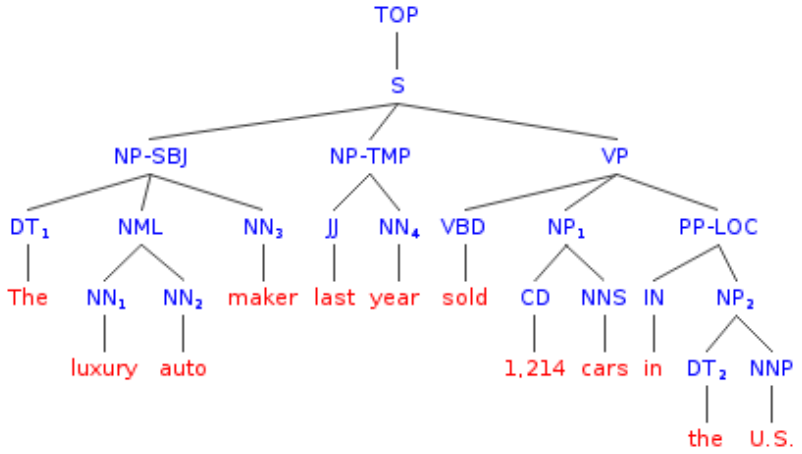
依存構造:

- nodes: 単語
- arcs: head-dependent の関係
- arc labels: 文法カテゴリ (ex. NMOD)

句構造:

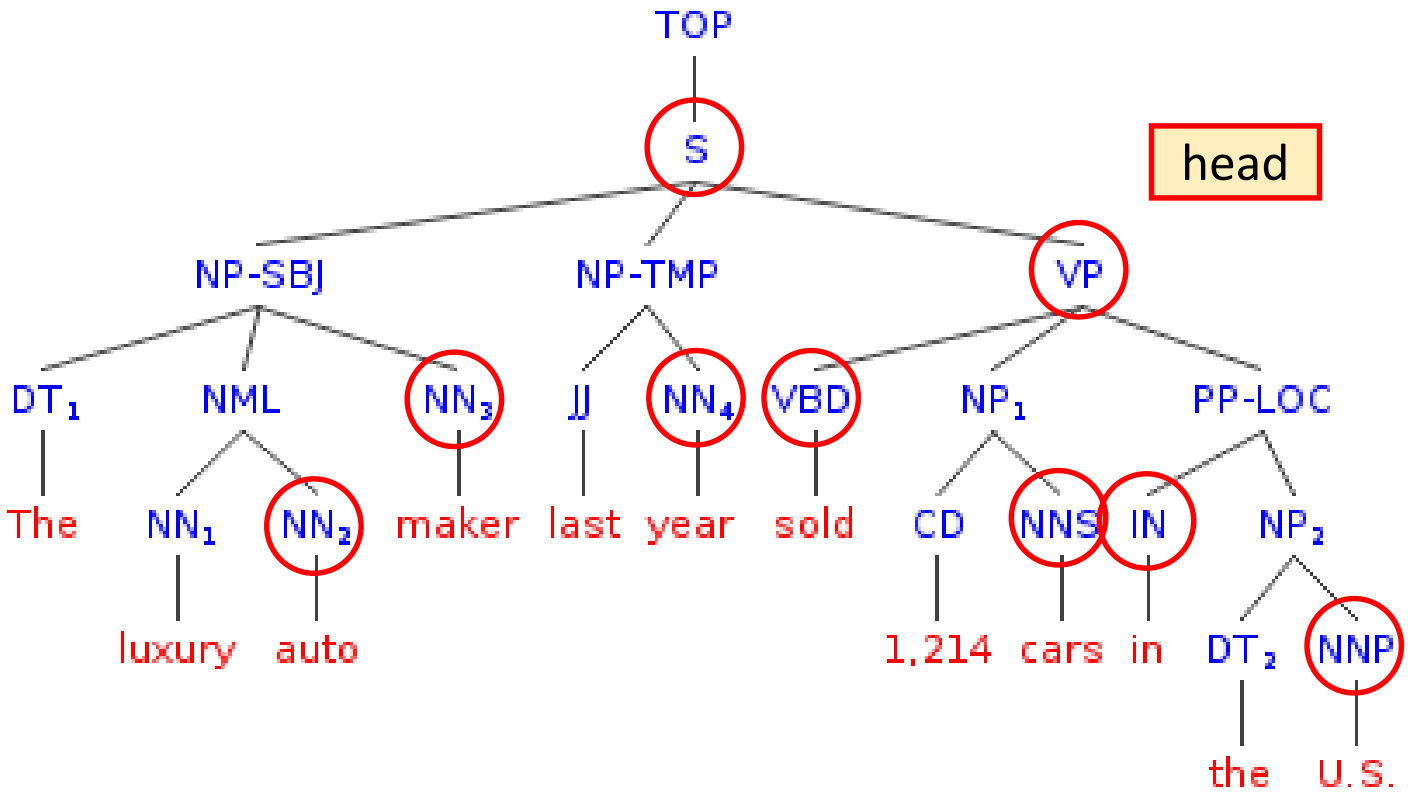
- nodes: 句
- node labels: 文法カテゴリ (ex. NP-SBJ)





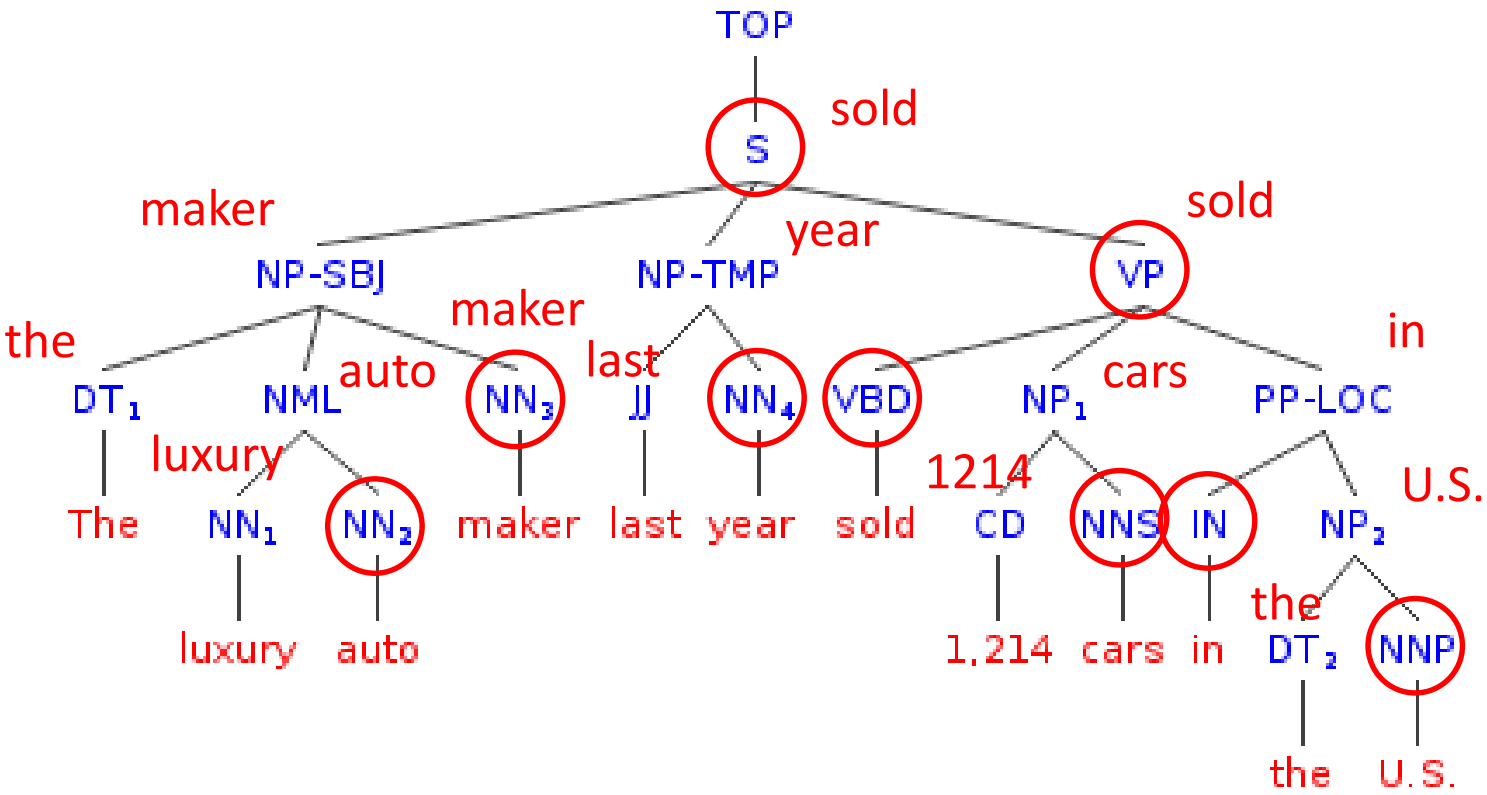
句構造から依存構造への変換

1. 各ノードの子供をスキャンして，headを決定する．



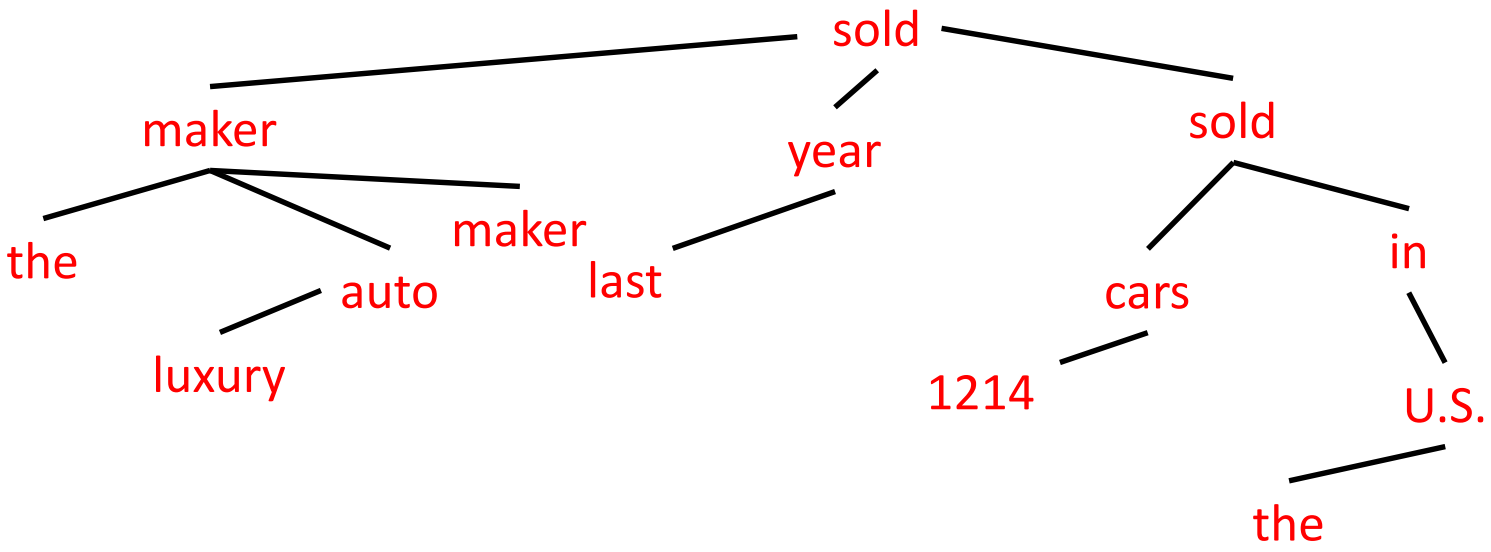
句構造から依存構造への変換

2. ノードにheadの単語を割り当てる.



句構造から依存構造への変換

3. ノードのカテゴリ（非終端記号）を取り除く.



句構造から依存構造への変換ルール

Head percolation table [Magerman '94, 'Collins '99]

Category	Direction	Head Categories
VP	left	TO VBD VBN MD VBZ VB VP VBG VBP ADJP NP
NP	right	EX $ CD QP PRP VBG JJ JJS JJR ADJP DT FW RB SYM
PP	left	IN TO FW
ADVP	left	RBR RB RBS FW ADVP CD JJR JJS JJ
..	..	.. ..

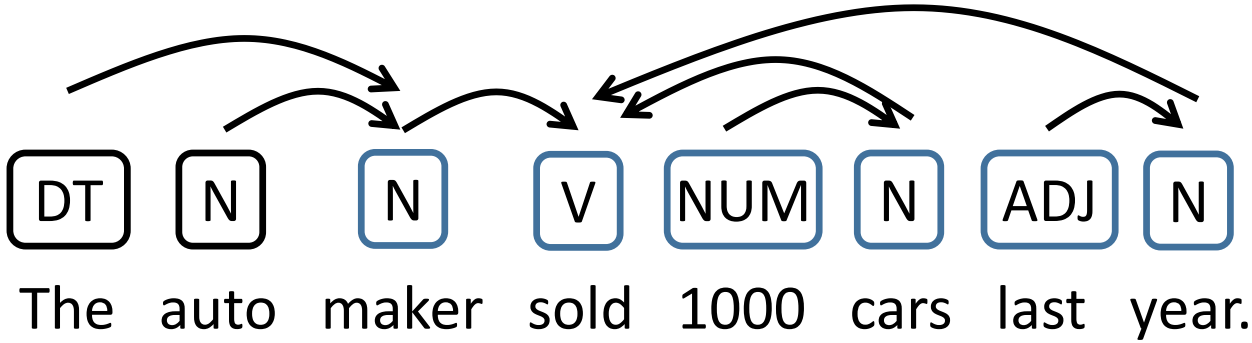
Example of "VP":

a. Scan the children of VP from left to right for the first child of "TO". If "TO" occurs, that child is the head.
Otherwise, scan the children for the first child of "VBD"; and so on.

b. If no child matches any category in the list, the left-most child is the head.

Projective v.s. Non-projective

Projectiveな依存構造木

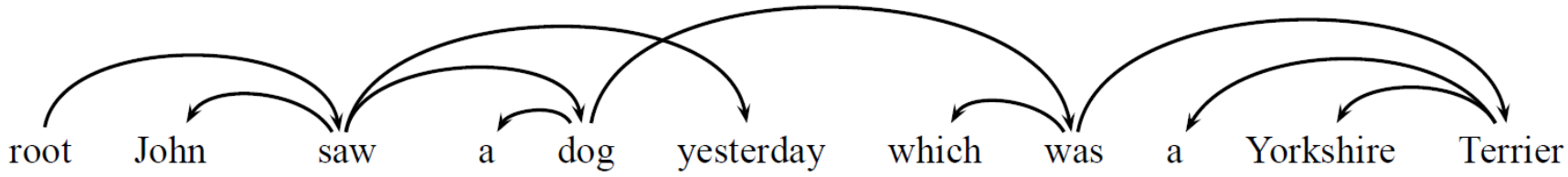


- 交差がない

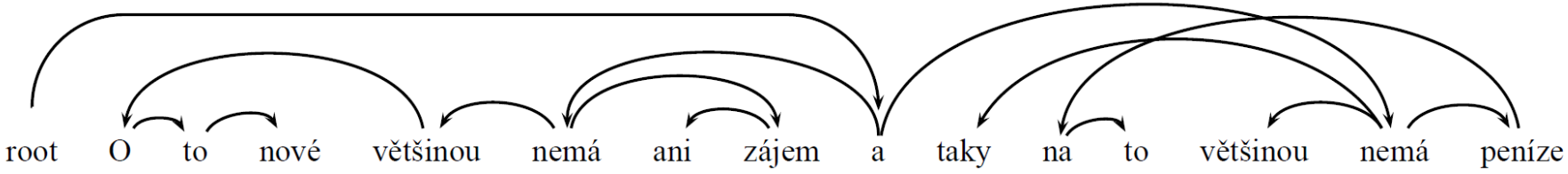
Projective v.s. Non-projective

Non-Projectiveな依存構造

English



Czech



He is mostly not even interested in the new things and in most cases, he has no money for it either.

Taken from “Non-projective Dependency Parsing using Spanning Tree Algorithms”, 2005

依存構造解析のアルゴリズム

- Transition-based Parsing（遷移型構文解析）
  - Shift-reduce algorithm [Yamada+ ‘03, Nivre ‘03]
  - Easy-first (non-directional) algorithm [Goldberg+ '10]

- Graph-based Parsing（グラフ型構文解析）
  - Eisner algorithm (CYK algorithm) [Eisner '96]
  - Chu-Liu-Edmonds algorithm [Chu+ '65, McDonald+ '05]

- その他
  - Sampling-based algorithm [Zhang+ '14]
  - Hill-climbing algorithm [Zhang+ '14]

依存構造解析のアルゴリズム

- Transition-based Parsing（遷移型構文解析） projective
  - Shift-reduce algorithm [Yamada+ '03, Nivre '03] projective
  - Easy-first (non-directional) algorithm [Goldberg+ '10]

- Graph-based Parsing（グラフ型構文解析） projective
  - Eisner algorithm (CYK algorithm) [Eisner '96]
  - Chu-Liu-Edmonds algorithm [Chu+ '65, McDonald+ '05] non-projective

- その他
  - Sampling-based algorithm [Zhang+ '14] non-projective
  - Hill-climbing algorithm [Zhang+ '14] non-projective

Graph-based Parsing Algorithm

Collins' Algorithm

http://www.cs.columbia.edu/~mcollins/

Collins' Algorithm

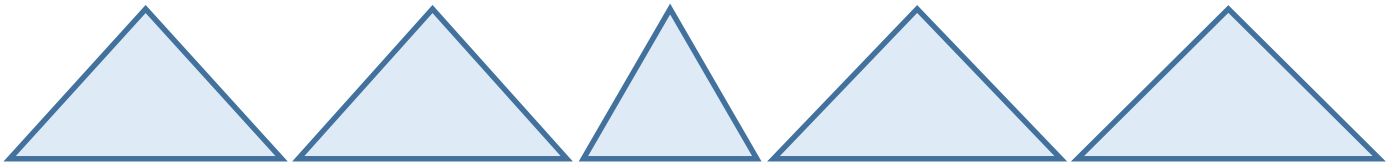
- A simple algorithm for computing the highest-scoring dependency tree under an arc-factored scoring model.
- The algorithm can be understood as an extension of the CKY algorithm to dependency parsing.
- Dependency tree is composed in a bottom-up manner based on dynamic programming.

Collins' Algorithm

She	read	a	short	novel	
0	1	2	3	4	5

初期化

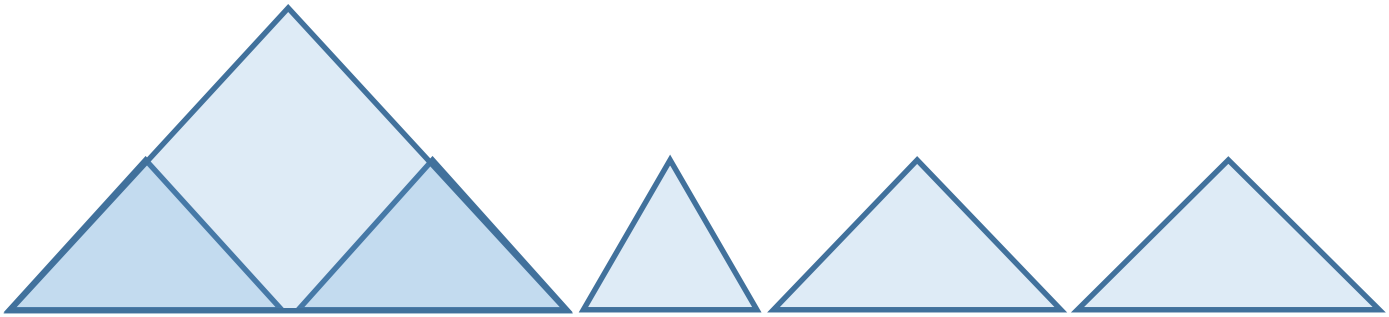
Set 1-length spans



She read a short novel

0 1 2 3 4 5

ボトムアップにスパンを結合



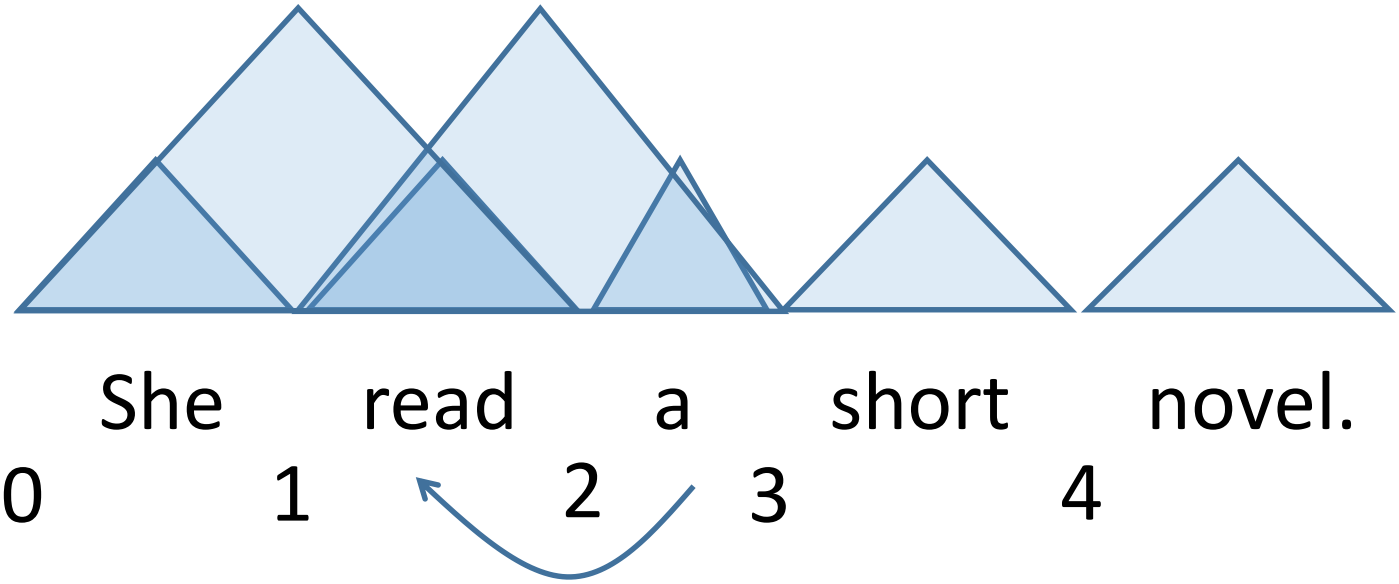
She read a short novel.

0 1 2 3 4 5

dep head

[0, 1] + [1, 2] → [0, 2, l → r]

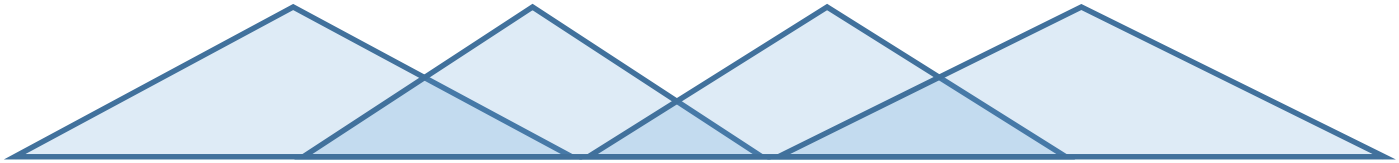
ボトムアップにスパンを結合



[0, 1] + [1, 2] → [0, 2, l → r]

[1, 2] + [2, 3] → [1, 3, l ← r]

Merge Spans in a Bottom-up Manner



She read a short novel.

0 1 2 3 4 5

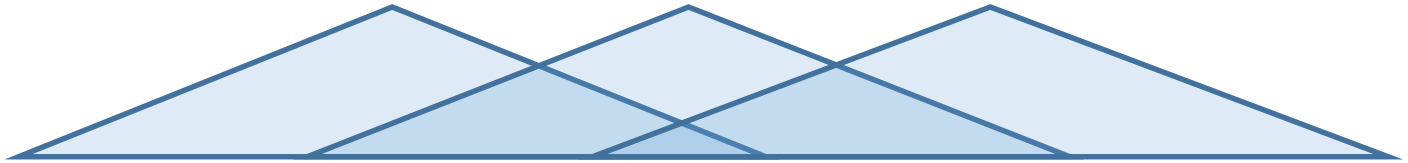
[0, 1] + [1, 2] → [0, 2, l → r] and [0, 2, l ← r]

[1, 2] + [2, 3] → [1, 3, l ← r] and [1, 3, l ← r]

[2, 3] + [3, 4] → … … …

[3, 4] + [4, 5] → … … …

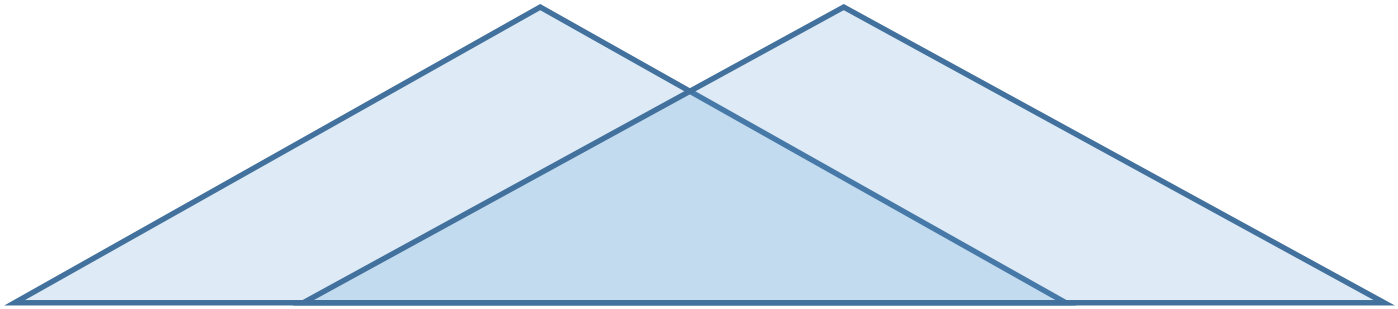
Collins' Algorithm



She read a short novel.

0 1 2 3 4 5

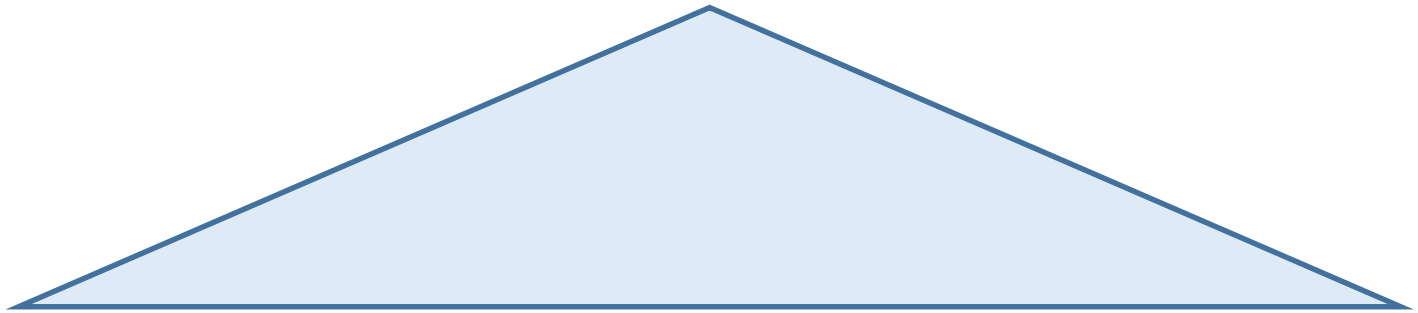
Collins' Algorithm



She read a short novel.

0 1 2 3 4 5

Collins' Algorithm



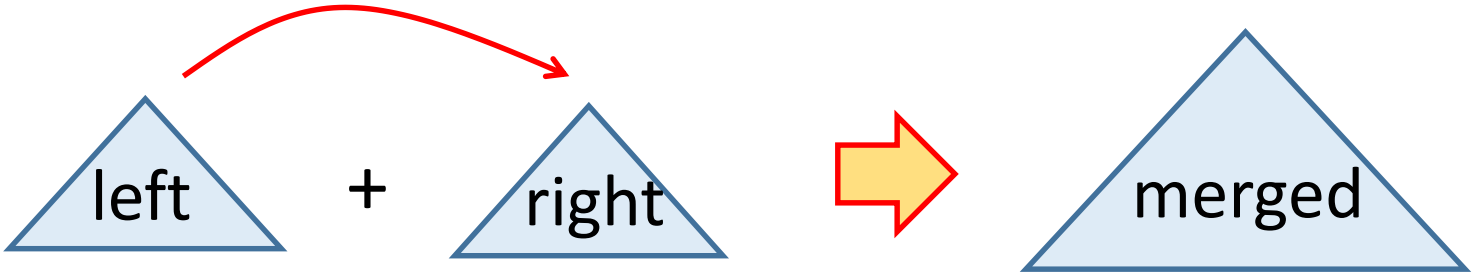
She read a short novel.

0 1 2 3 4 5

- 全ての可能な木構造の作り方を試して，最もスコアが高い木構造を選ぶ.

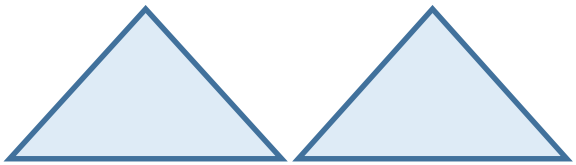
スコア計算の例: Arc-factoredモデル

- 依存構造の良さを計るために, スコアを計算する.
- Arc-factoredモデルでは，左と右の部分木のスコアと，向きのスコアの和が全体のスコアとなる.



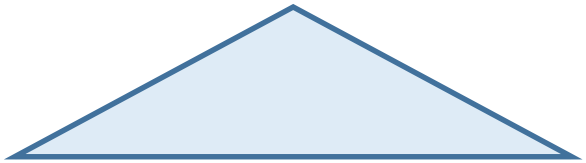
score(merged) =
score(left) + score(right) + score(left → right)

最良な依存構造の探索



She	read	a	short	novel	
0	1	2	3	4	5

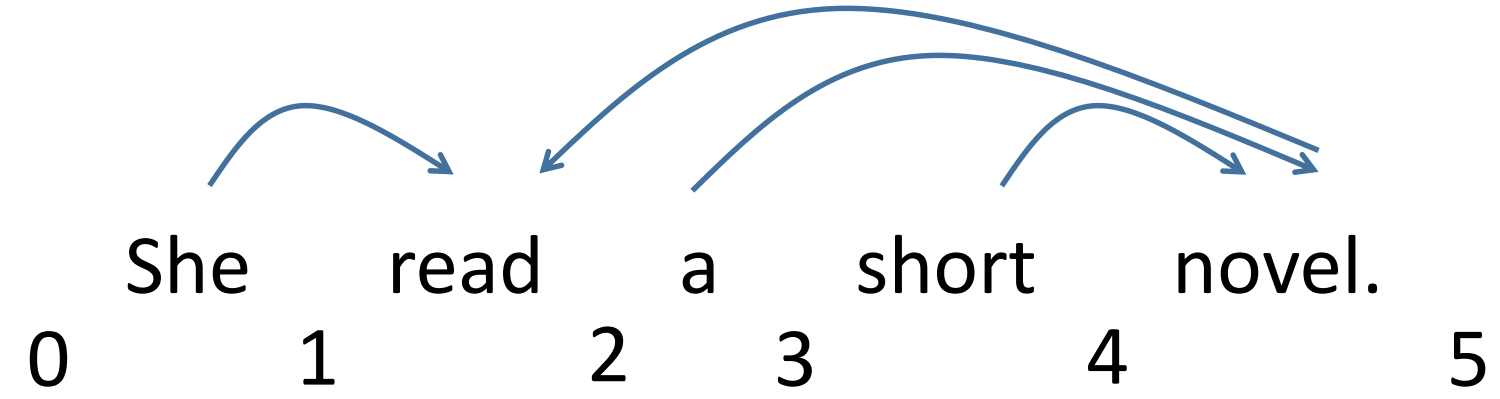
最良な依存構造の探索



She read a short novel.
0 1 2 3 4 5

max (
  score(0, 2, She read, l → r),
  score(0, 2, She read, l ← r)
)

最良な依存構造の探索



- スコアが適切であれば，Collinsアルゴリズムにしたがって正しい依存構造を決定することができる.
- スコアは，機械学習によってデータから統計的に決定する.

Collins' Algorithm

Computational complexity

- 空間計算量: $o(n^3)$
- 時間計算量: $o(n^5)$

Eisner's Algorithm

http://www.cs.jhu.edu/~jason/

Eisner's Algorithm

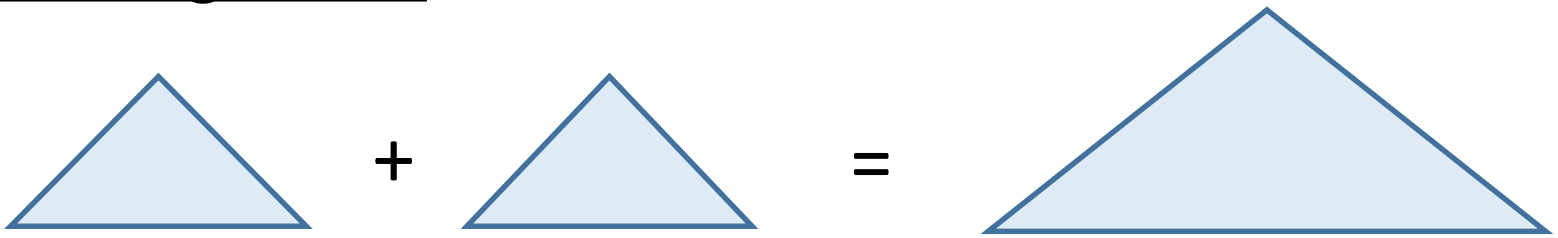
- Collins' algorithm $o(n^5)$ is はとても遅い…

→ Eisnerが$o(n^3)$ アルゴリズムを1996年に提案.

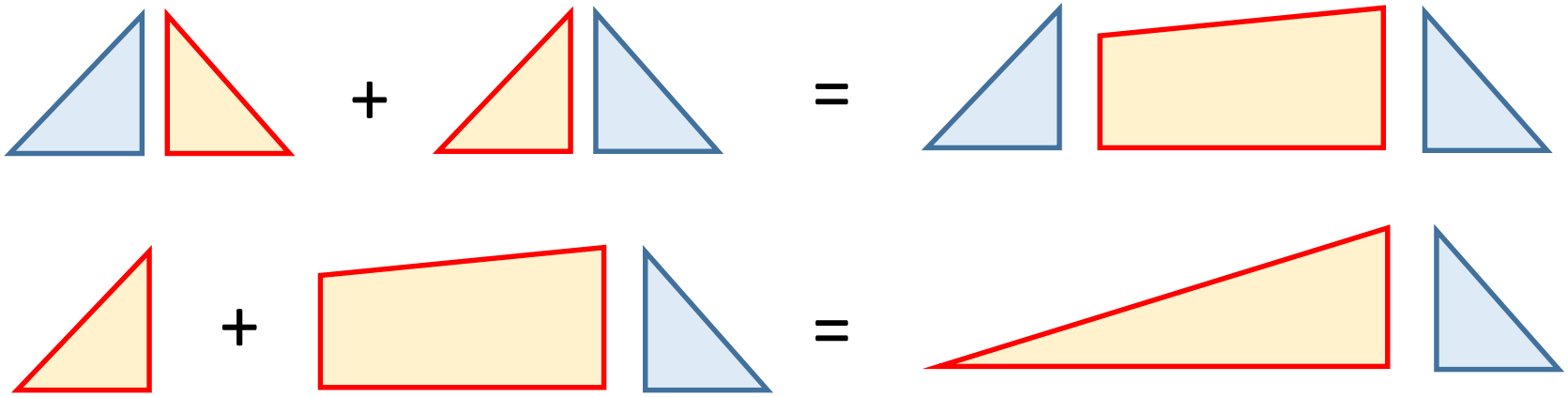
- Eisnerアルゴリズムの基本的な考え:
  左と右の依存構造を別々に組み上げる

Eisner's Algorithm

Collins' algorithm



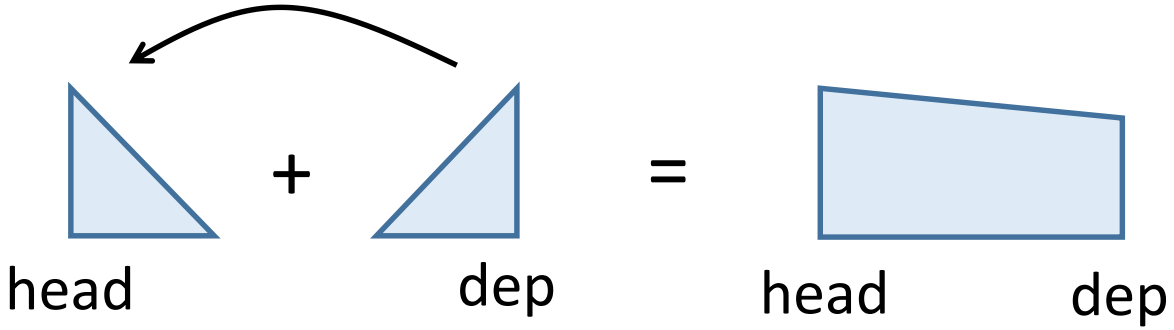
Eisner's algorithm



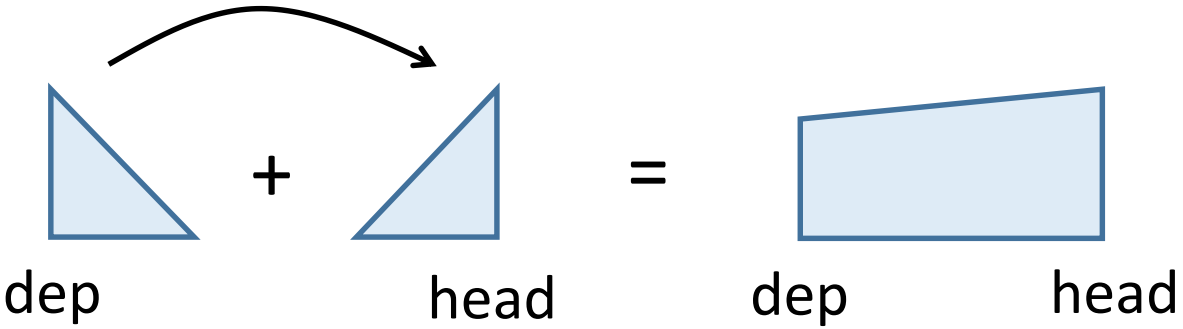
Eisner's Algorithm

スパンを結合する4つの規則

Rule 1:



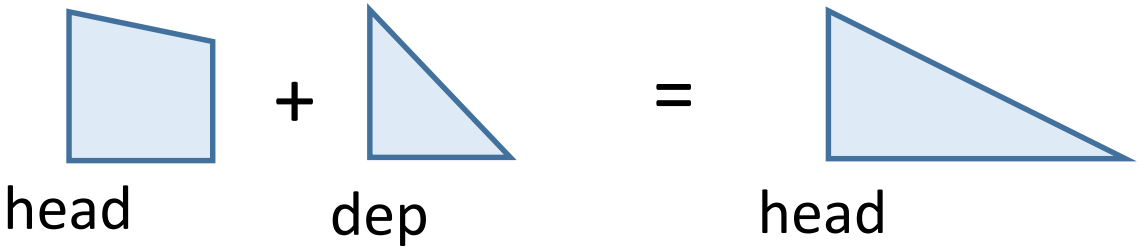
Rule 2:



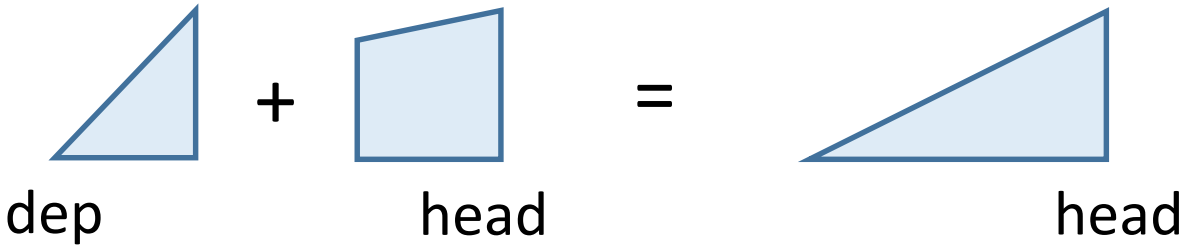
Eisner's Algorithm

スパンを結合する4つの規則

Rule 3:

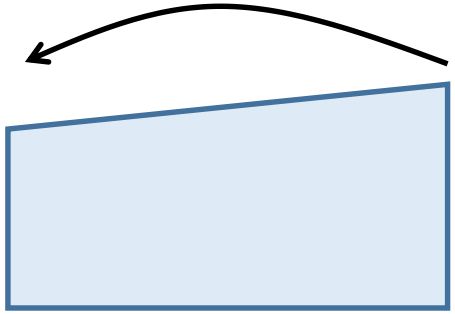


Rule 4:



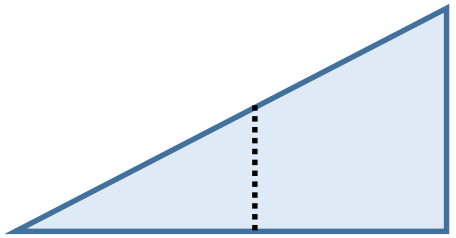
Imcomplete and Complete Spans

Incomplete



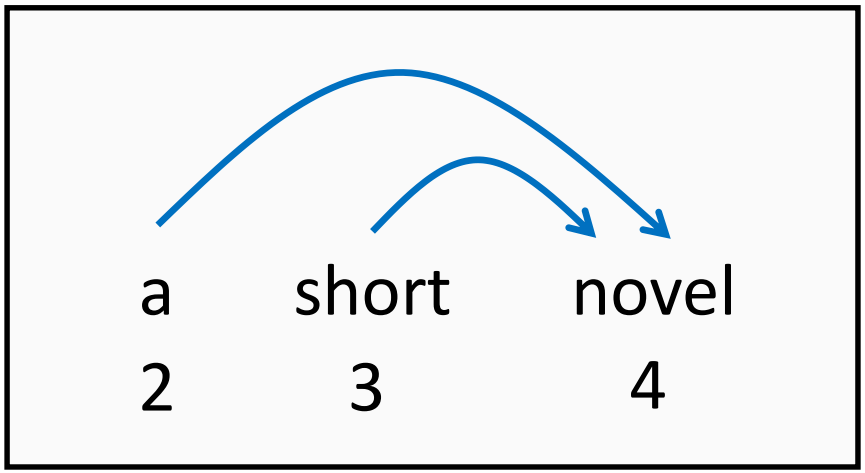
depの単語は左に子供を持つ可能性がある

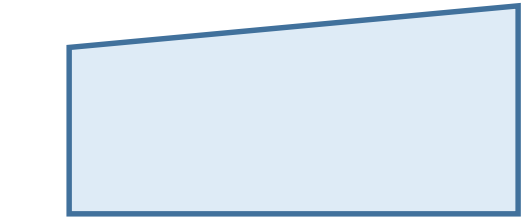
Complete



depの単語は子供を持たない

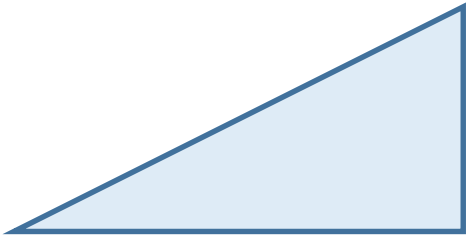
Eisner's Algorithm





i=2 j=4

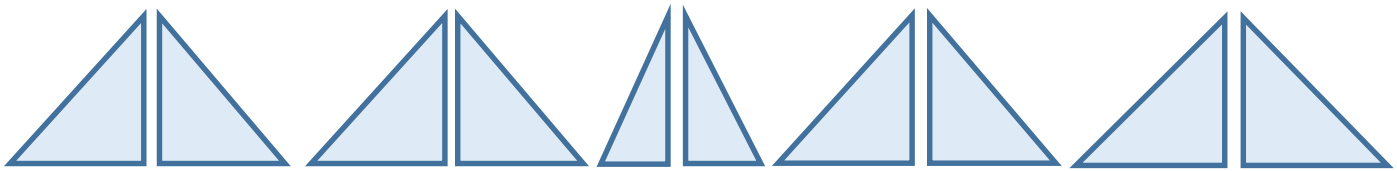
どちらもあり得る



i=2 j=4

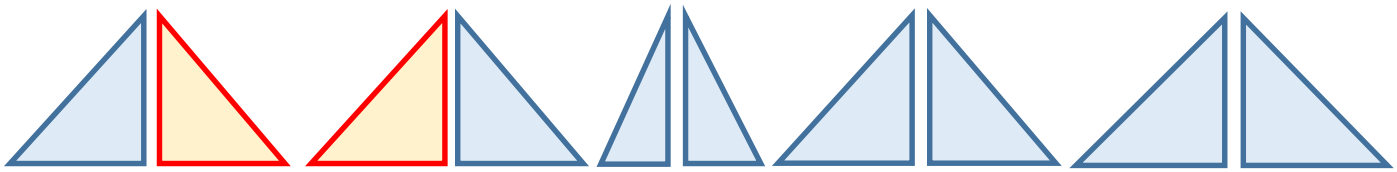
Eisner's Algorithm

初期化



She	read	a	short	novel.
0	1	2	3	4

Eisner's Algorithm

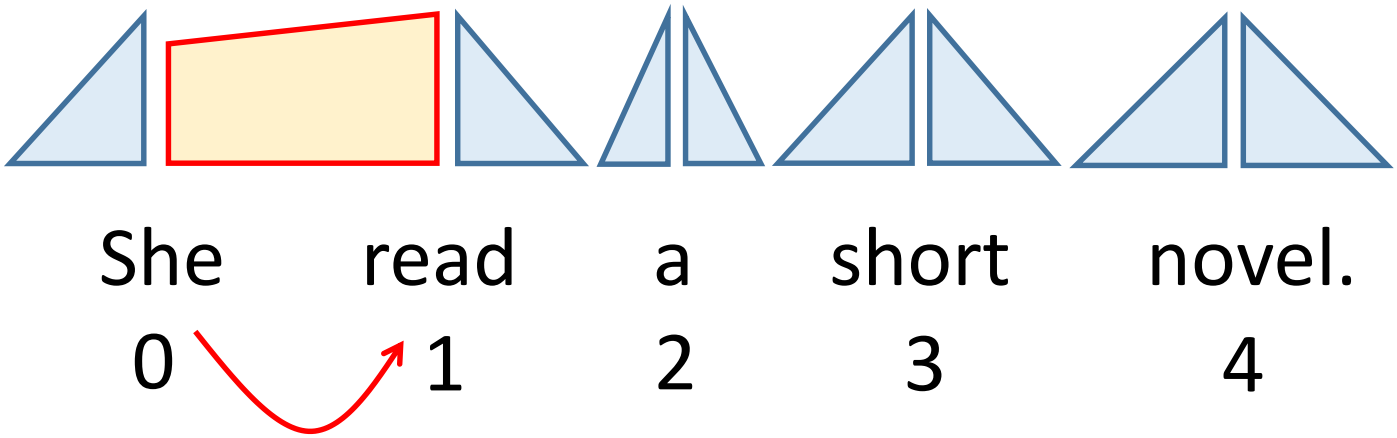


She	read	a	short	novel.
0	1	2	3	4

- Apply rule2:

[0, 1, comp] + [1, 2, comp] → [0, 2, incomp]

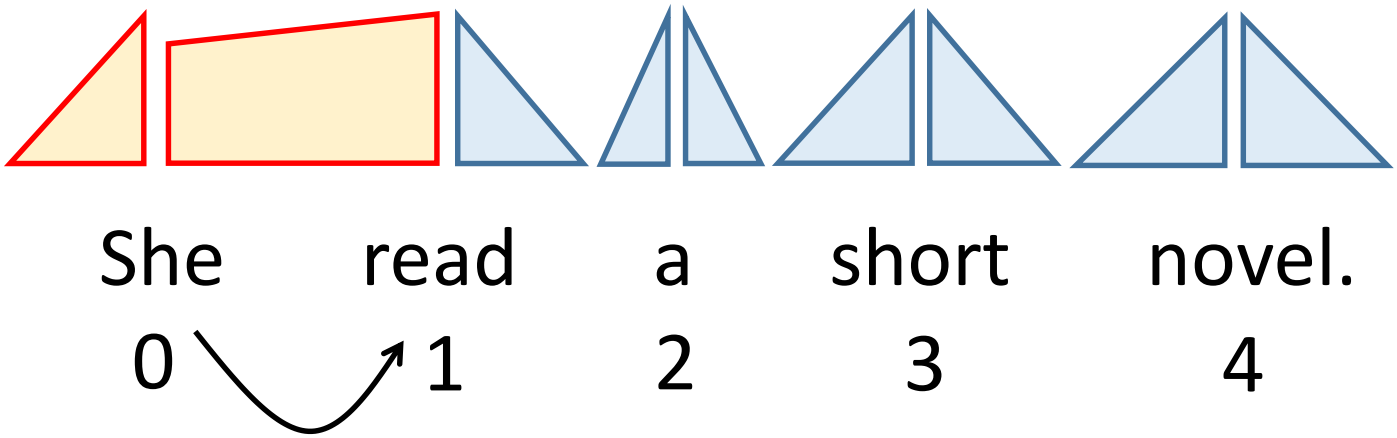
Eisner's Algorithm



- Apply rule2:

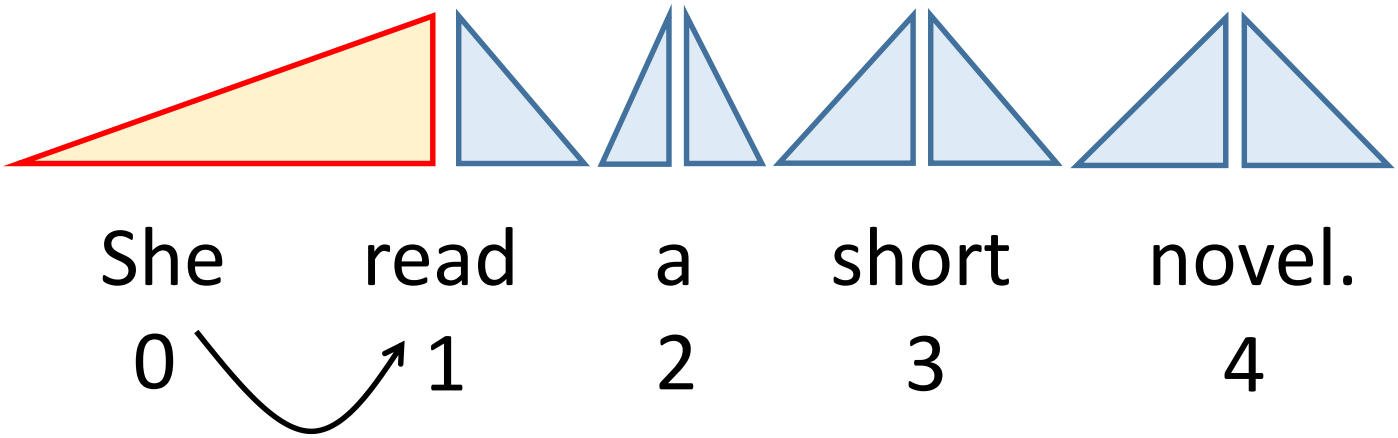
[0, 1, comp] + [1, 2, comp] → [0, 2, incomp]

Eisner's Algorithm



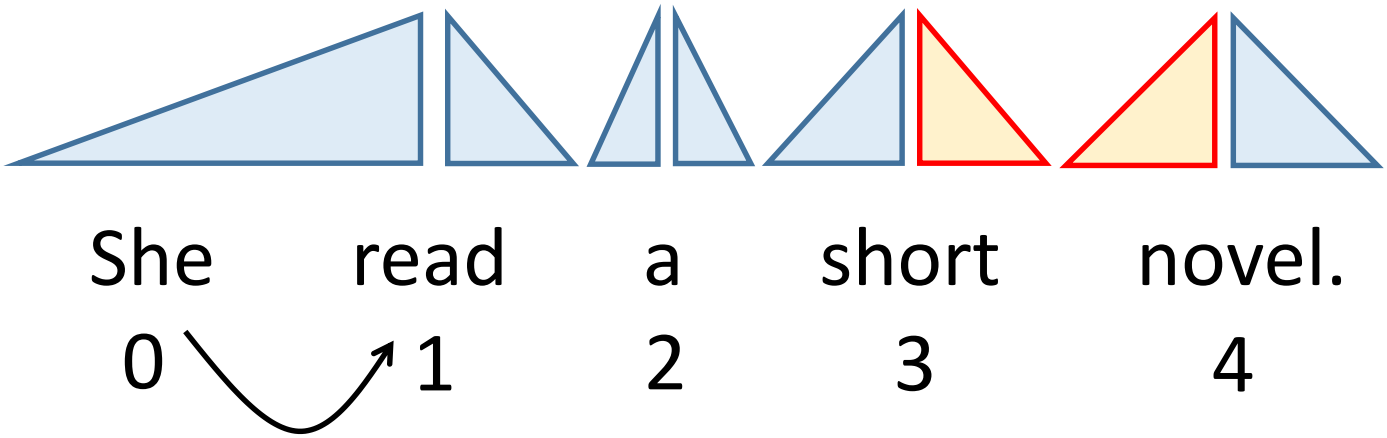
- Apply rule2:
  [0, 1, comp] + [1, 2, comp] → [0, 2, incomp]
- Apply rule4:
  [0, 1, comp] + [1, 2, incomp] → [0, 2, comp]

Eisner's Algorithm

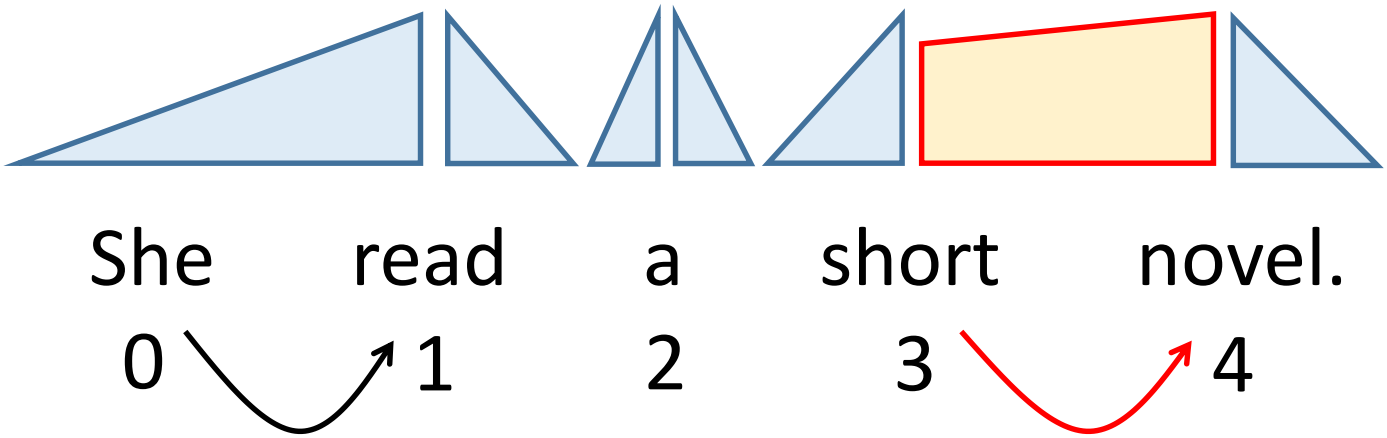


- Apply rule2:
  [0, 1, comp] + [1, 2, comp] → [0, 2, incomp]
- Apply rule4:
  [0, 1, comp] + [1, 2, incomp] → [0, 2, comp]

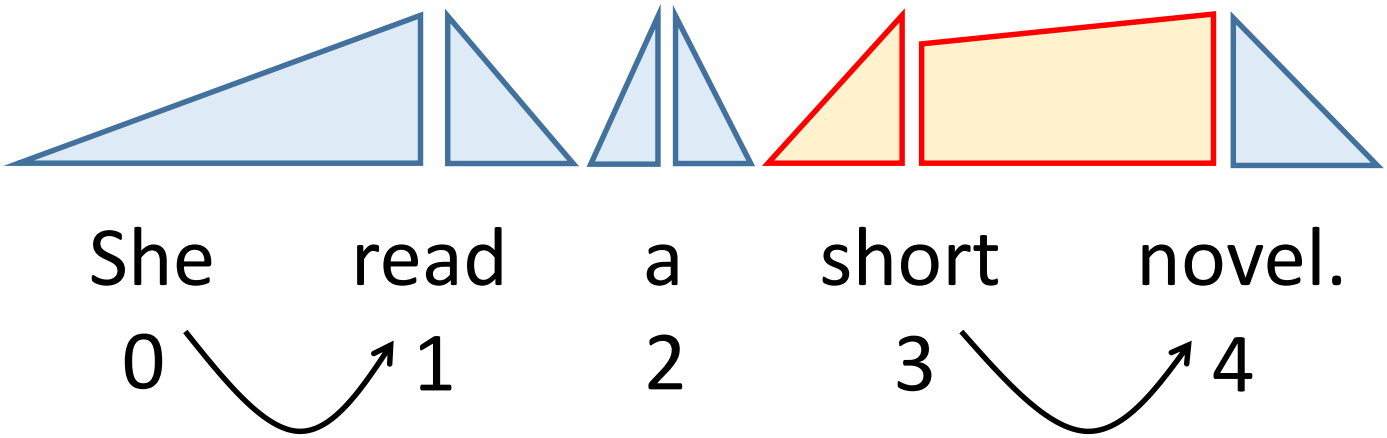
Eisner's Algorithm



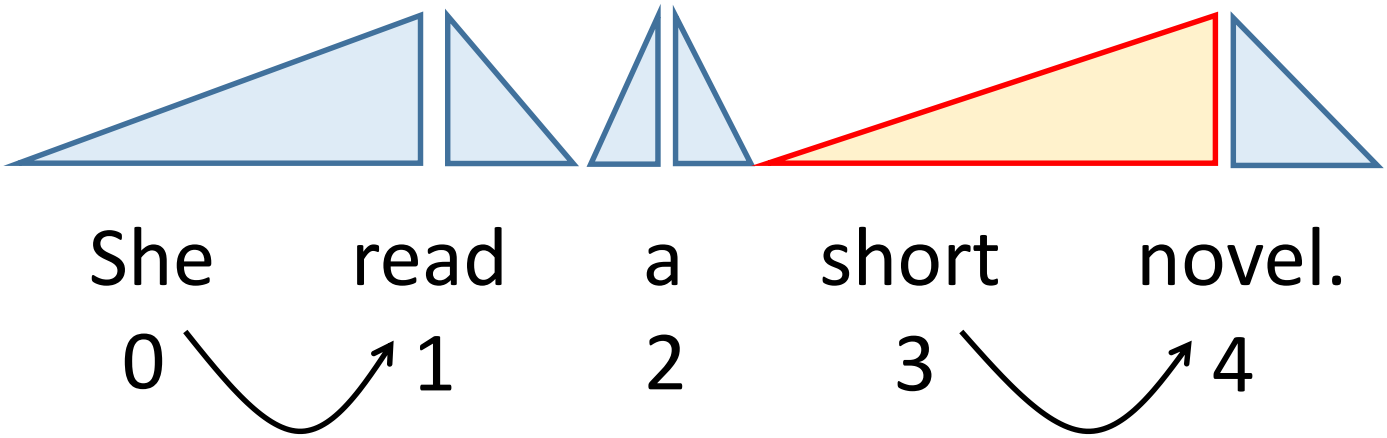
Eisner's Algorithm



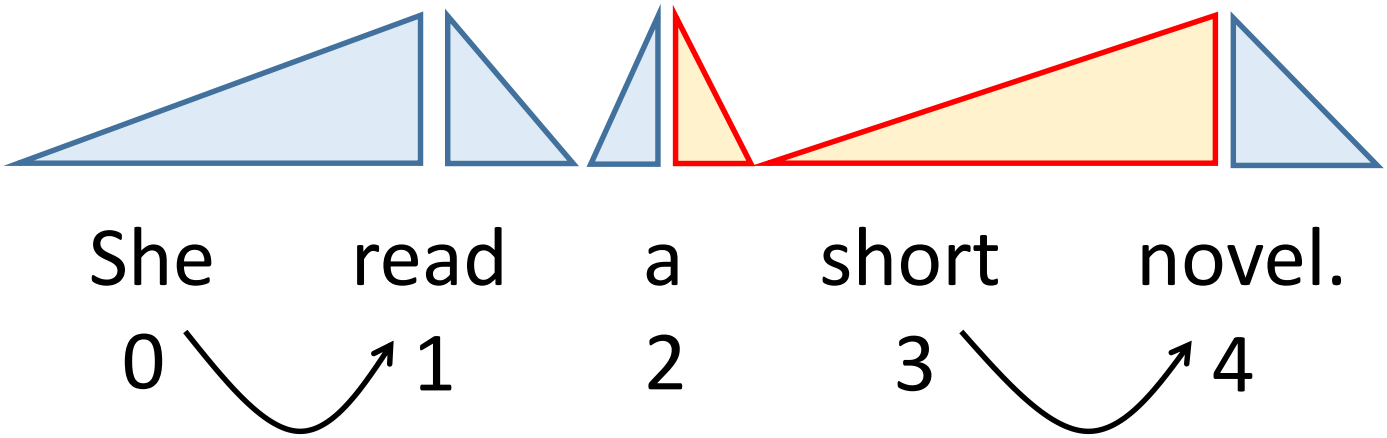
Eisner's Algorithm



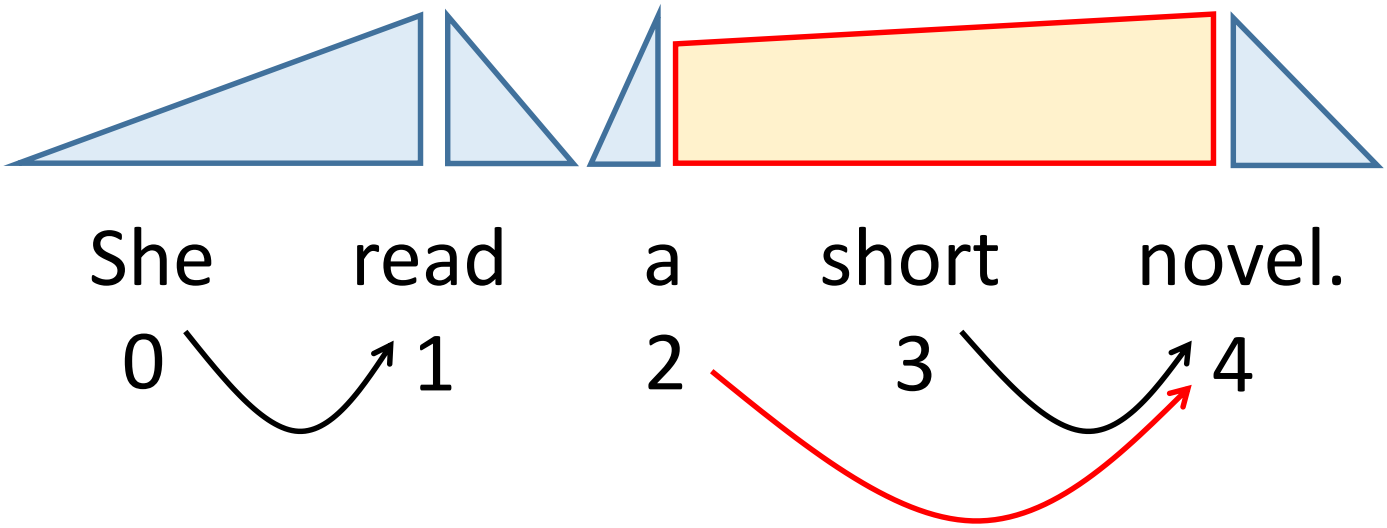
Eisner's Algorithm



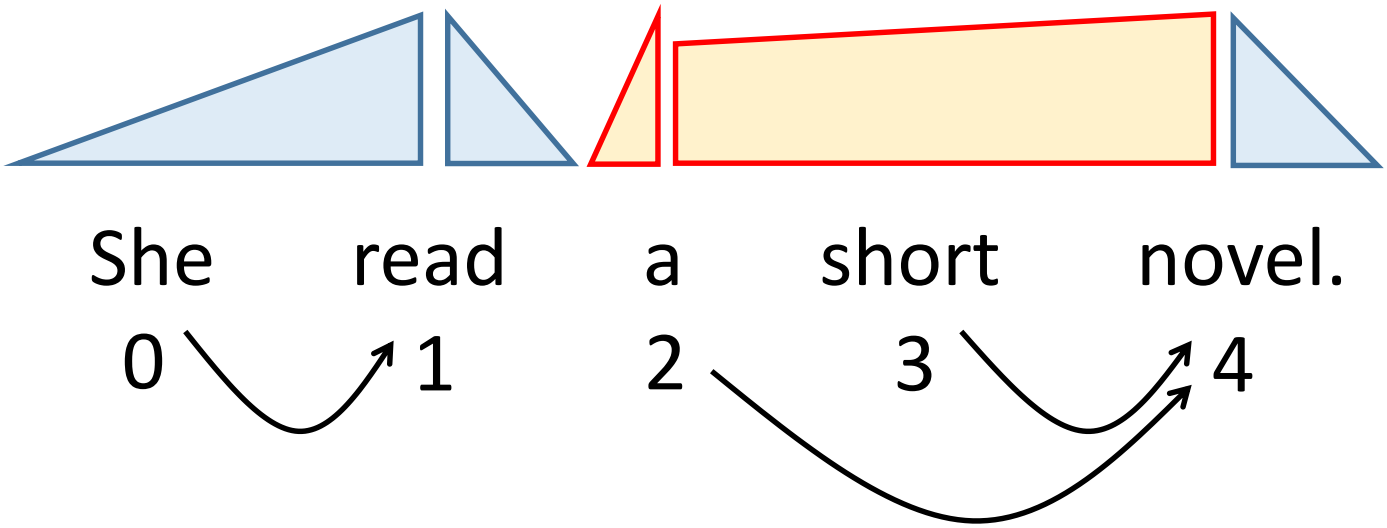
Eisner's Algorithm



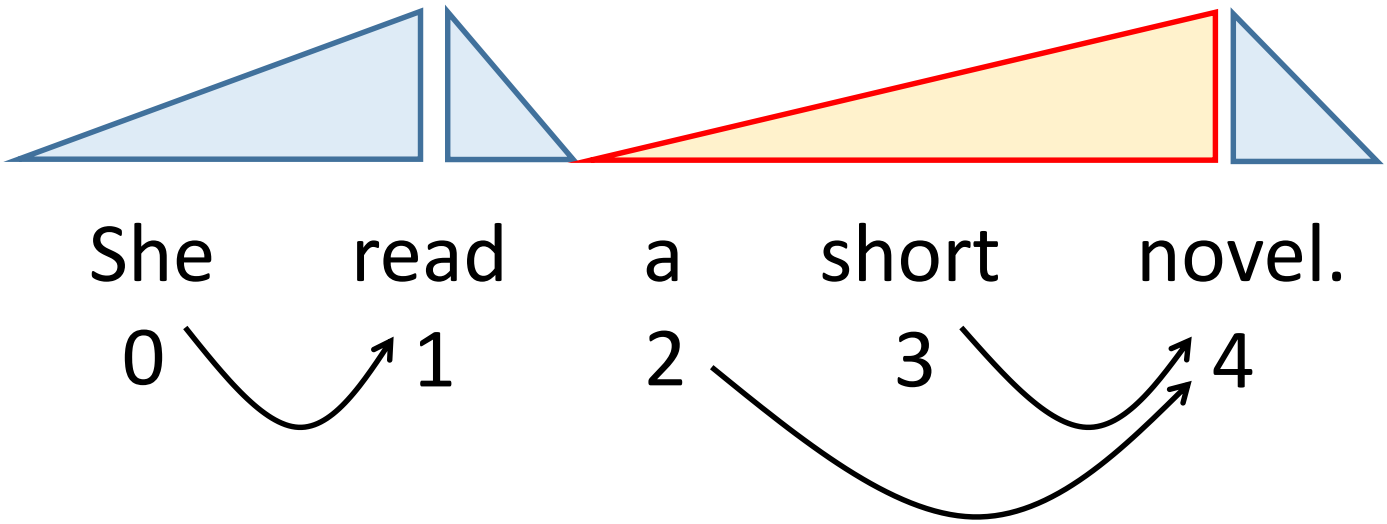
Eisner's Algorithm



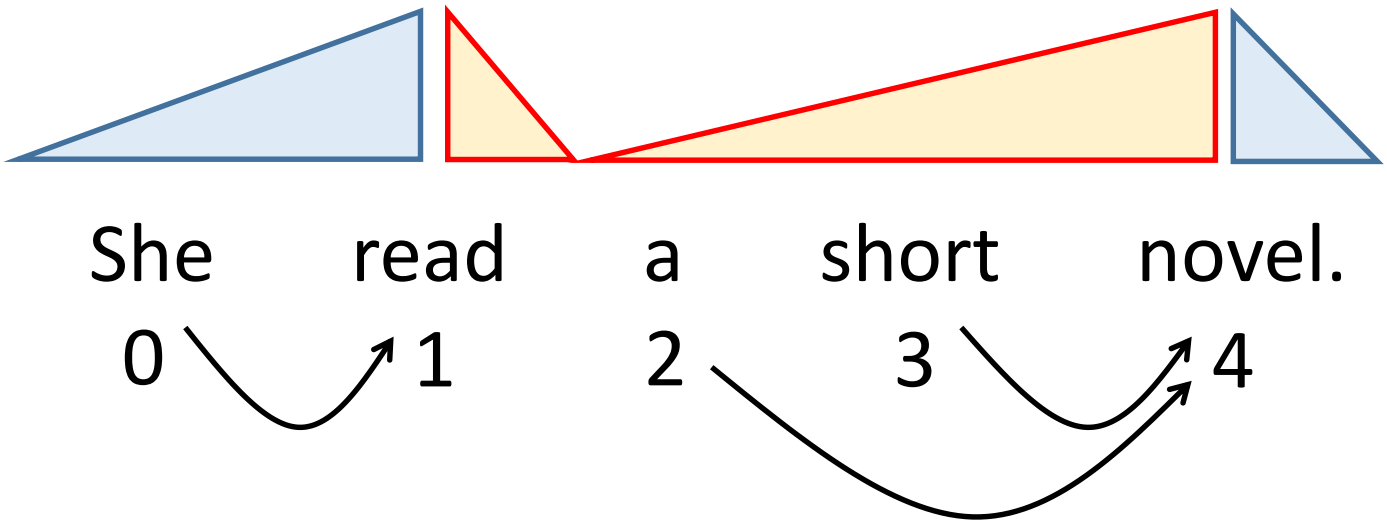
Eisner's Algorithm



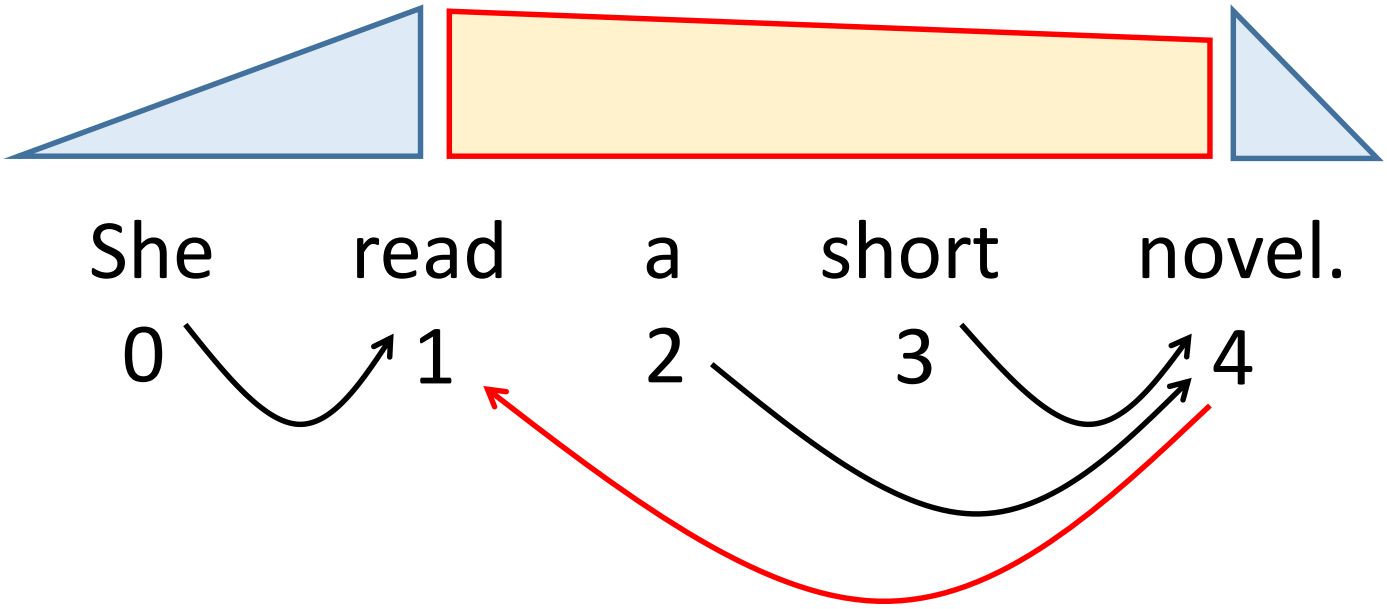
Eisner's Algorithm



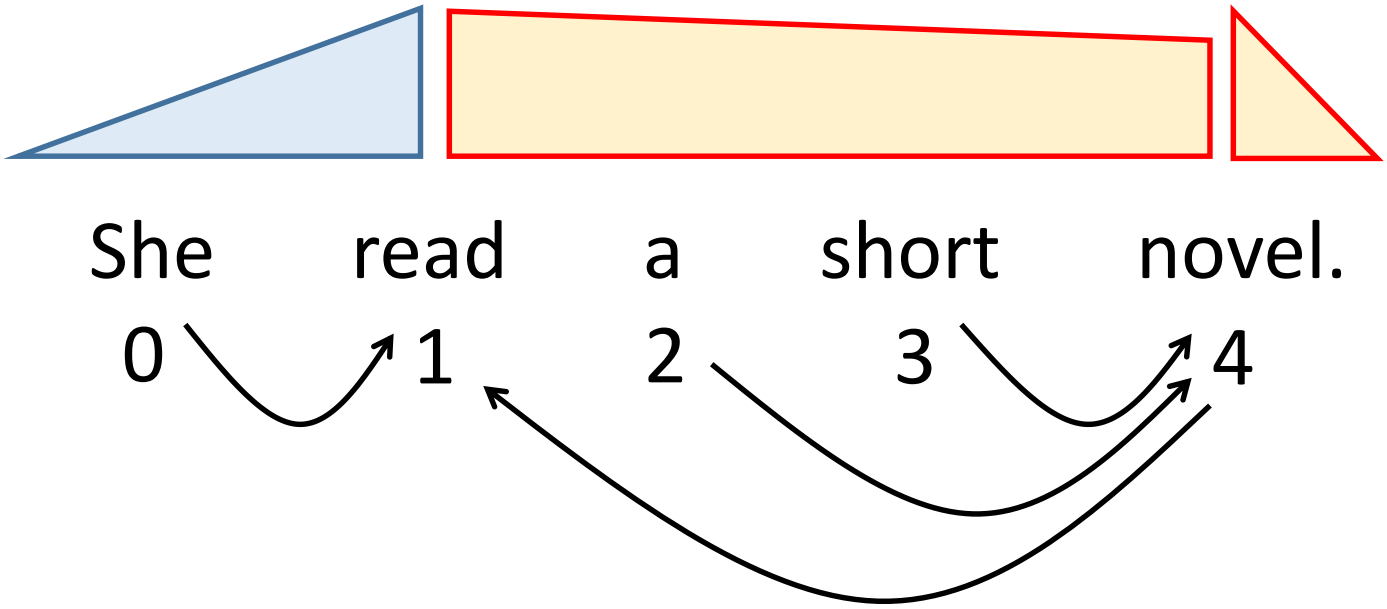
Eisner's Algorithm



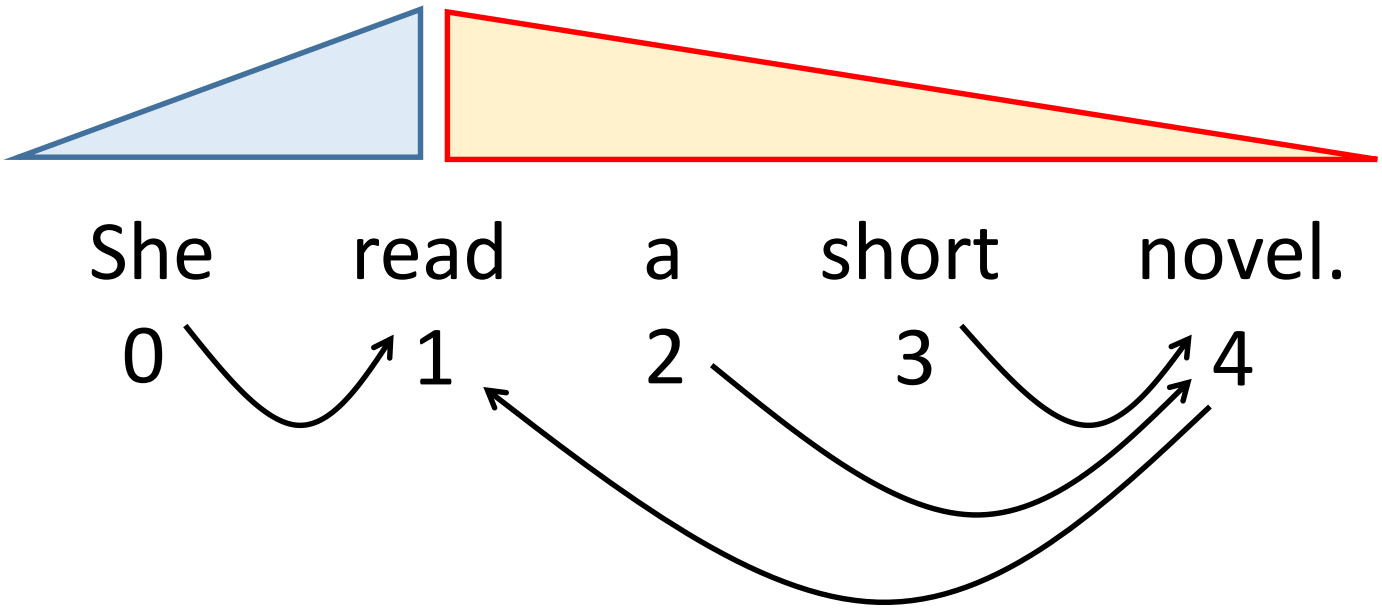
Eisner's Algorithm



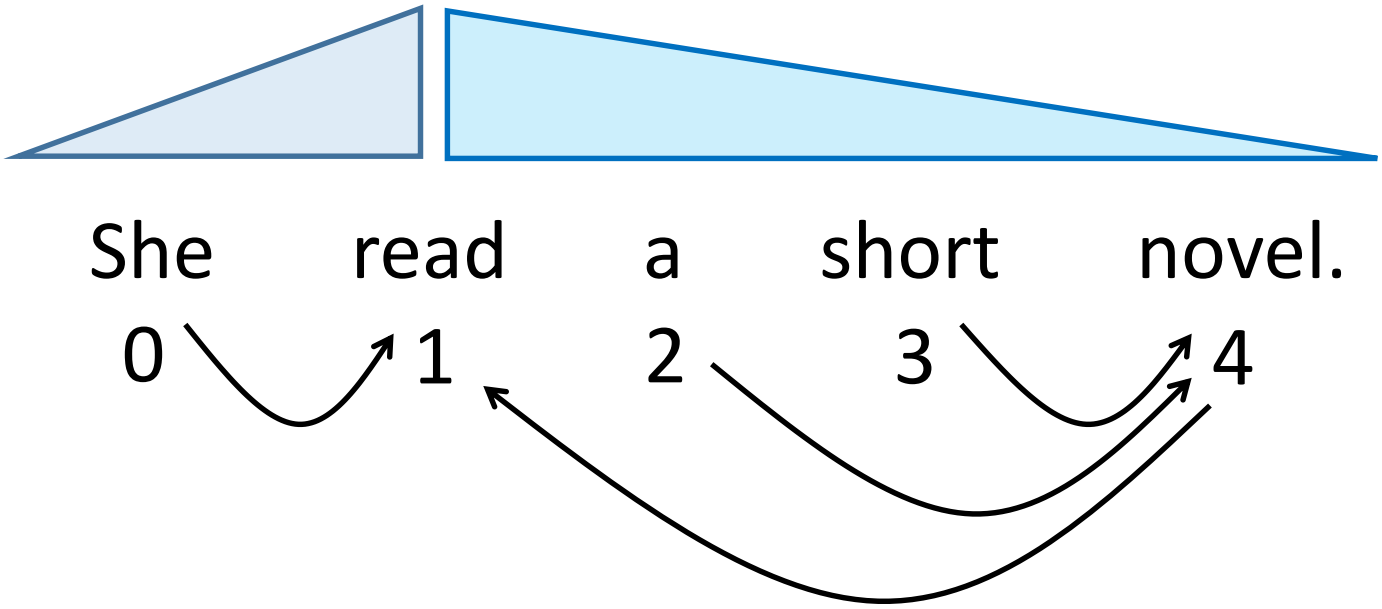
Eisner's Algorithm



Eisner's Algorithm



Eisner's Algorithm



スコア計算の例: Arc-factoredモデル

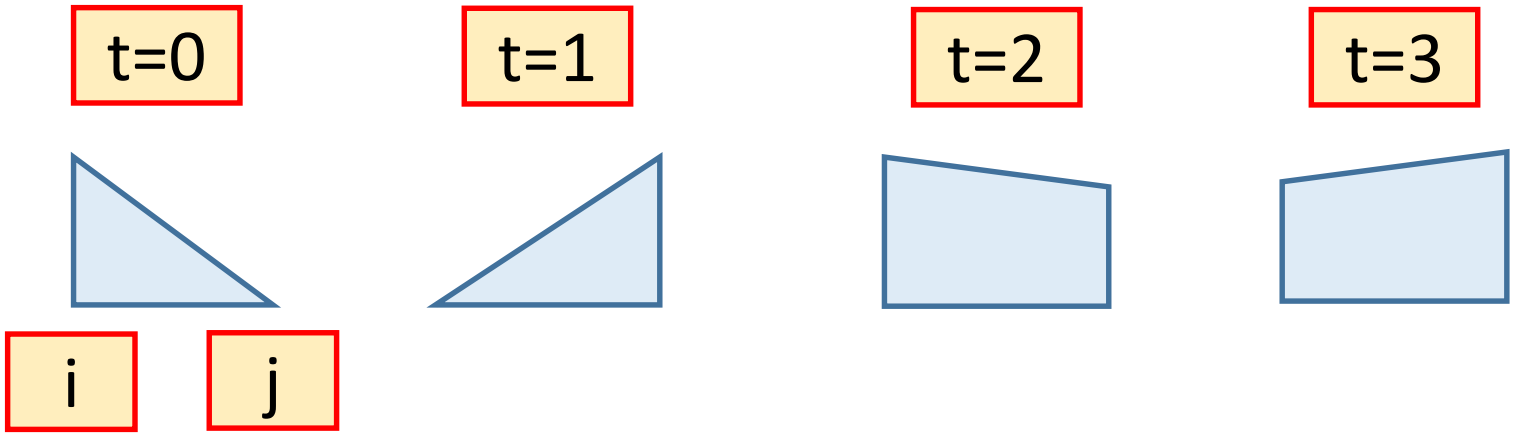
- 依存構造の良さを計るために, スコアを計算する.
- Arc-factoredモデルでは，左と右の部分木のスコアと，向きのスコアの和が全体のスコアとなる.

score(merged) =
score(left) + score(right) + score(left → right)

Eisner algorithm：スパンの表現

インデックス（i, j），タイプt，スコアによってスパンを表現する

t: type of Eisner's span

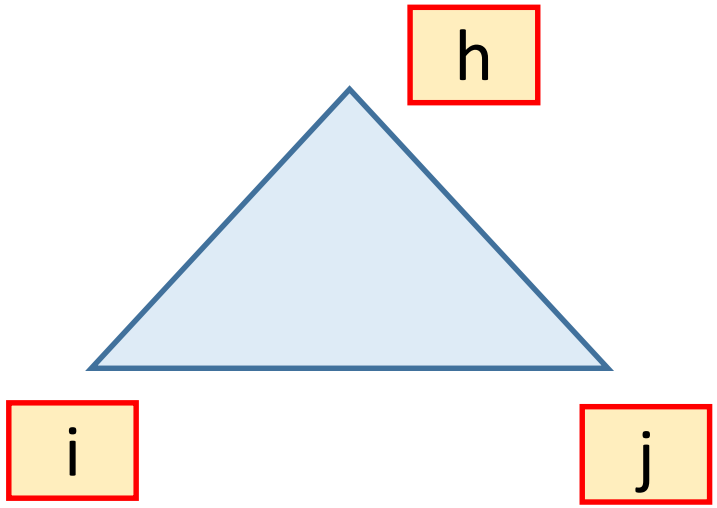


[i, j, t, score]

Collins algorithm：スパンの表現

インデックス（i, j, h）とスコアによってスパンを表現する

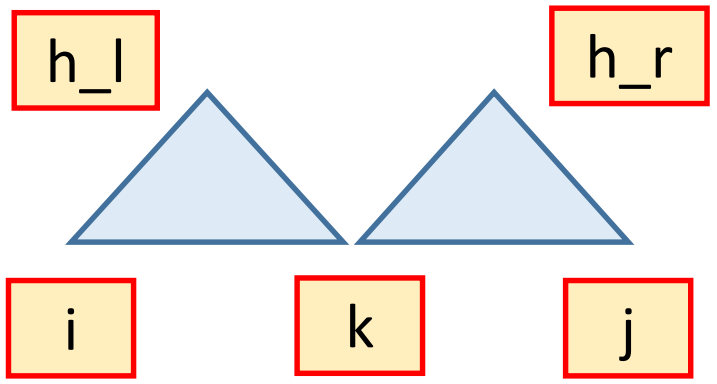
h: headの位置インデックス



[i, j, h, score]

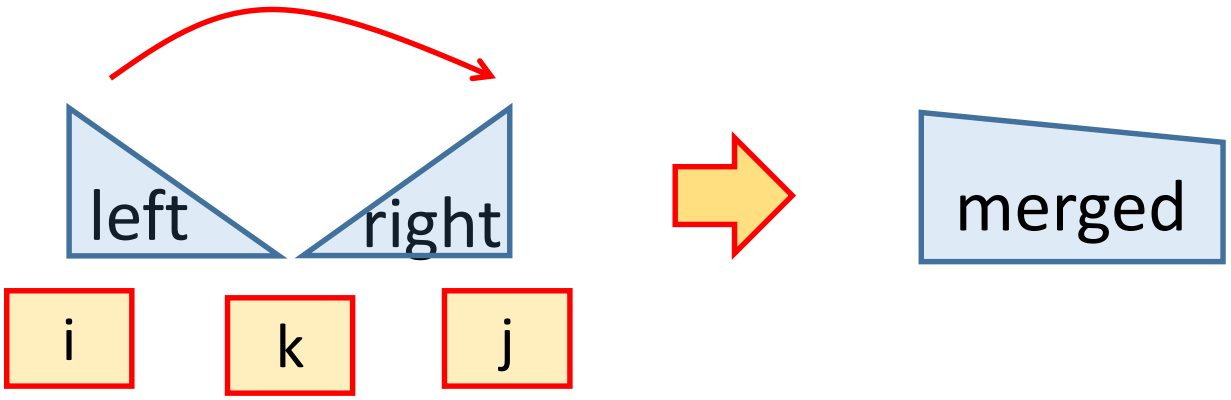
Collins Algorithmの疑似コード

- For each span: [i, j] in a bottom-up order...
  - For each mid-point: k (i < k < j)
    - Merge [i, k] with [k, j] → [i, j]
    - If score of [i, j] is the highest, keep it in the chart



Eisner Algorithmの疑似コード

- For each span: [i, j] in a bottom-up order...
  - For each mid-point: k (i < k < j)
    - Merge [i, k] with [k, j] → [i, j]
    - If score of [i, j] is the highest, keep it in the chart



Eisner's Algorithm

Computational complexity

- 空間計算量: $o(n^3)$
- 時間計算量: $o(n^3)$

スコア計算のモデル

- 線形モデル
- Support vector machine (SVM)
- ニューラルネットワーク

など

スコア計算のモデル

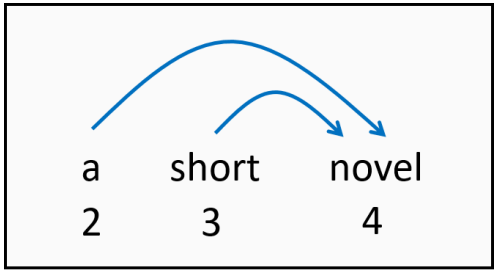
線形モデル

- 依存構造からfeature（素性，特徴量）を抽出する
- 各featureに対して重みwが割り当てられる.
- 依存構造のスコアは，featureの重みwの和とする.

score(t) = score(f1) + ... + score(fn)

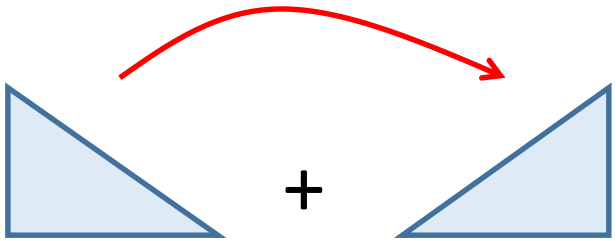
スコア計算のモデル

特徴量fの例



- f1: ‘ The head is a verb.’
- f2: ‘The dependent is a noun.’
- f3: ‘The head is a verb and the dependent is a noun.’
- f4: ‘The head is a verb and the predecessor of the head is a pronoun.’
- f5: ‘The arc goes from left to right.’
- f6: ‘The arc has length 2

特徴量の抽出



w_d w_h

特徴ベクトル

$$v \in R^d$$

$$d \approx 10^6 \sim 10^8$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \dots \\ 0 \\ 1 \end{pmatrix}$$

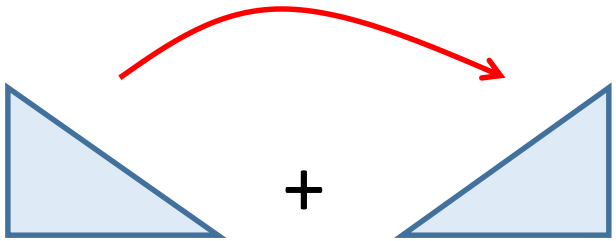
← w_h is a verb or not

← w_d is a noun or not

… … …

← Distance between w_d and w_h is 2 or not

線形モデルによるスコア計算



w_d w_h

重みベクトル

$w \in R^d$

特徴ベクトル

$v \in R^d$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \dots \\ 0 \\ 1 \end{pmatrix} \bullet \begin{pmatrix} 1.1 \\ -0.5 \\ 0.0 \\ -0.1 \\ \dots \\ 3.7 \\ -2.1 \end{pmatrix} = -1.1$$

スコア