Natural Language Processing (8)

Parsing (2): Dependency Parsing

Daisuke Kawahara

Department of Communications and Computer Engineering, Waseda University

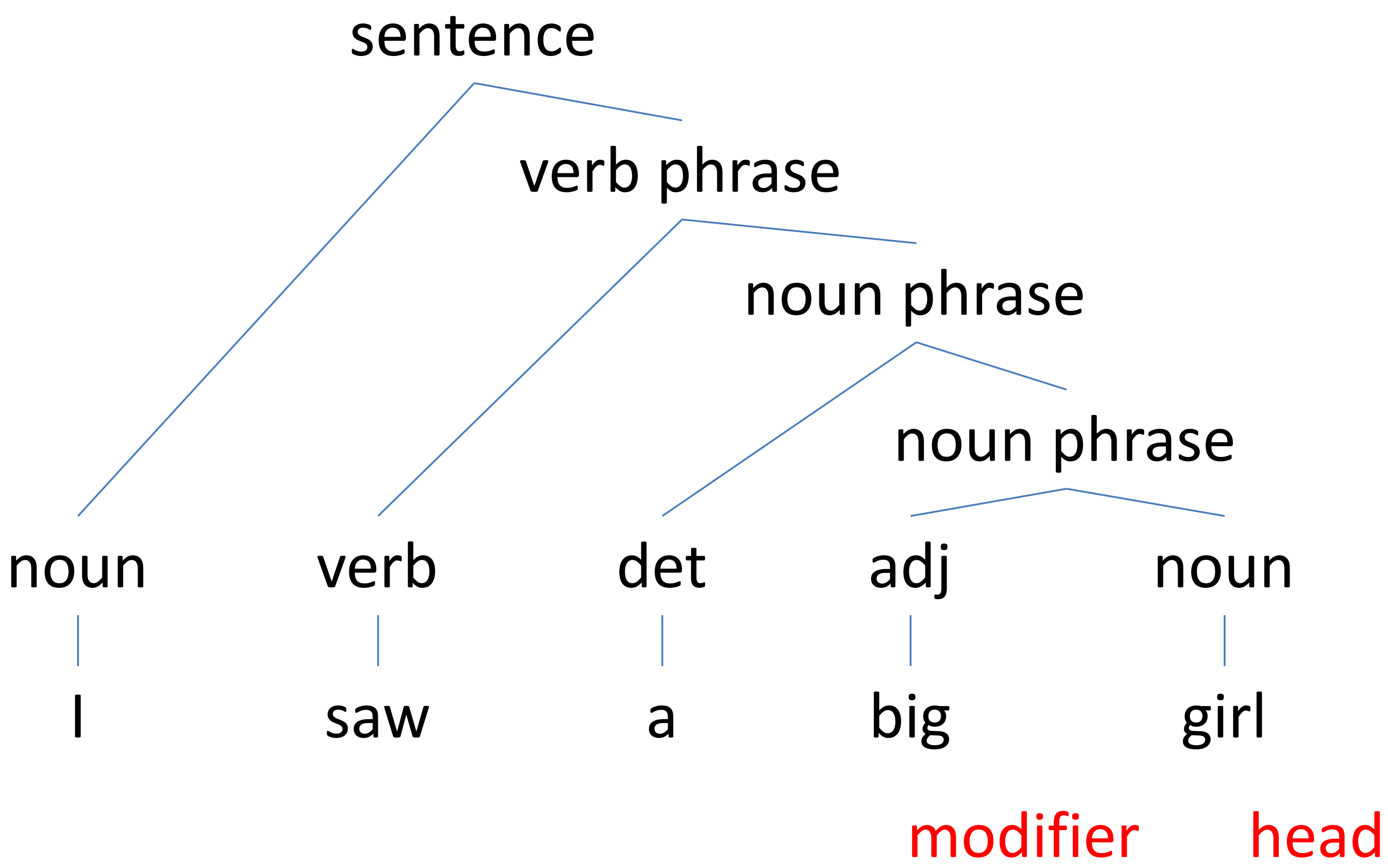
Lecture Plan

1. Overview of Natural Language Processing
2. Formal Language Theory
3. Word Senses and Embeddings
4. Topic Models
5. Collocations, Language Models, and Recurrent Neural Networks
6. Sequence Labeling and Morphological Analysis
7. Parsing (1)
8. Parsing (2)
9. Transfer Learning
10. Knowledge Acquisition
11. Information Retrieval, Question Answering, and Machine Translation
12. Guest Talk (1)
13. Guest Talk (2)
14. Project: Survey or Programming
15. Project Presentation

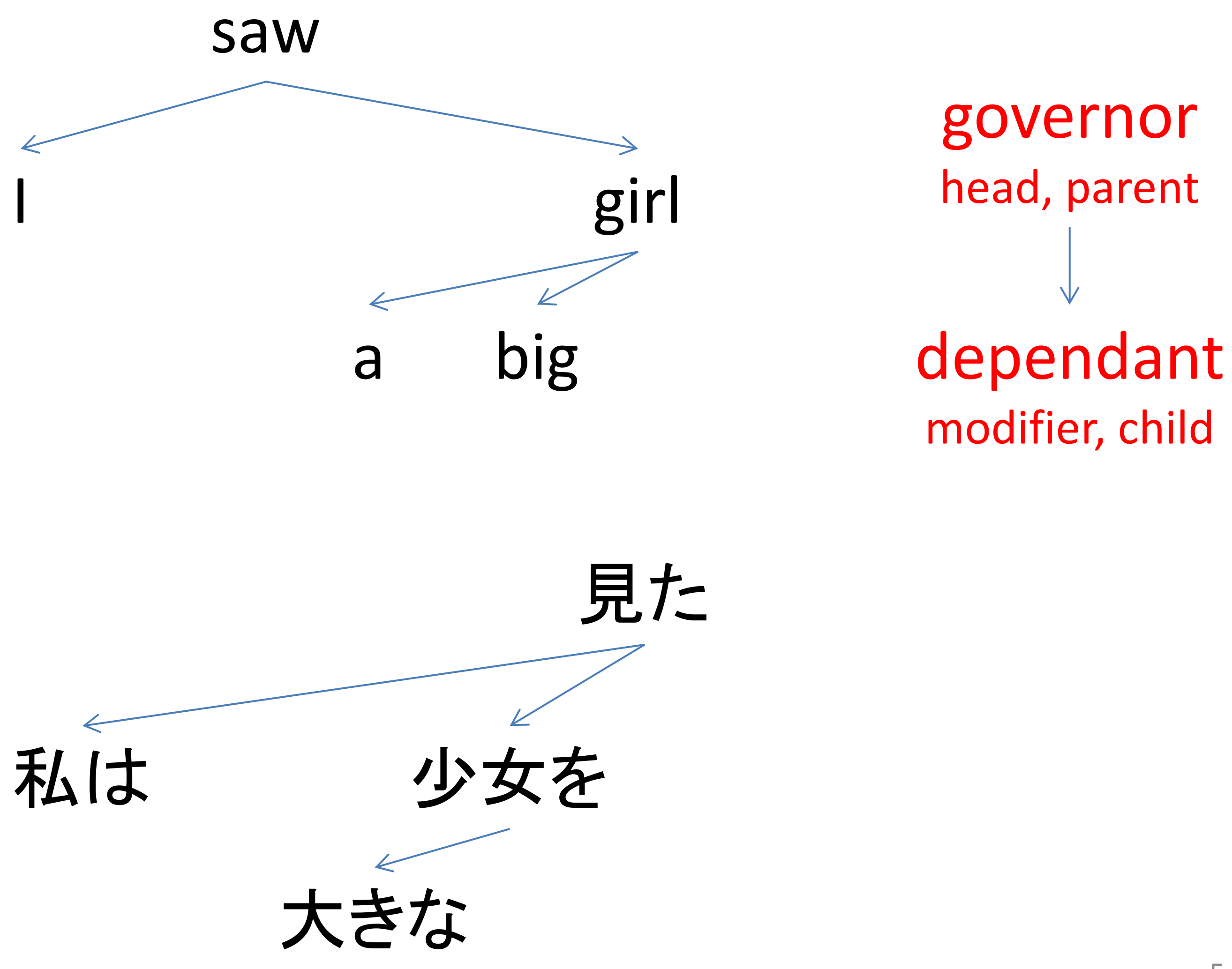
Table of Contents

- Dependency formalism
- Graph-based parsing
- Transition-based parsing
- Japanese dependency parsing

Review: Phrase Structure

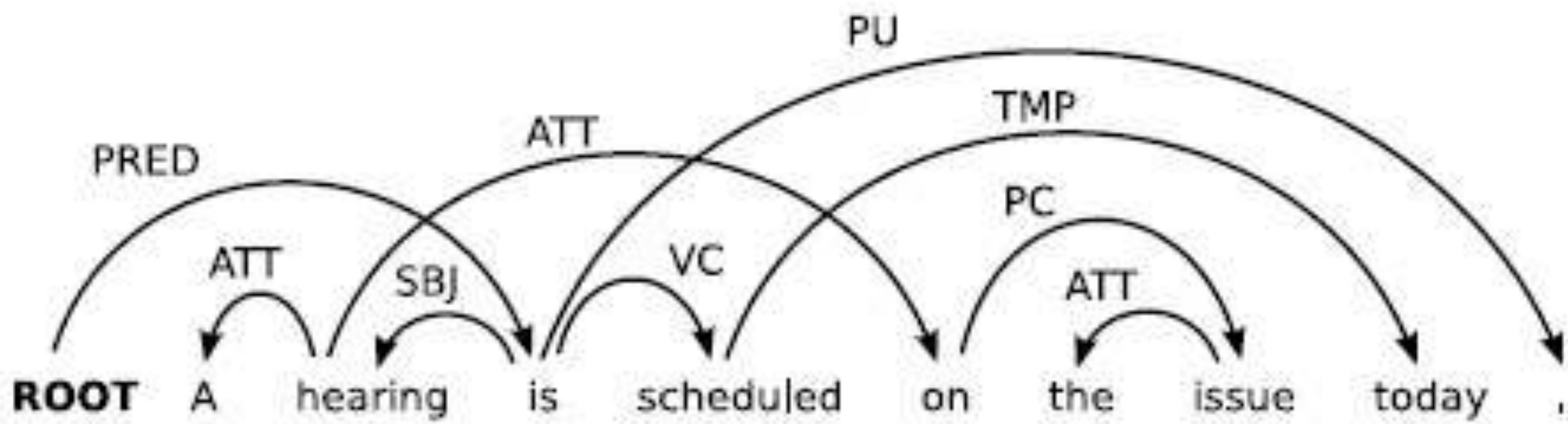


Dependency Structure



Dependency Parsing (1/2)

- Outputs a **dependency tree** from an input sentence where...
  - node of a (directed) graph: word
  - arc of a graph: dependency with a syntactic role



- Non-projective / Projective

Dependency Parsing (2/2)

- Successfully employed for…
  - machine translation
  - knowledge acquisition
  - …

- Research on data-driven dependency parsing is a boom
  - dependency treebanks
  - resources of the CoNLL shared tasks
  - Universal dependencies

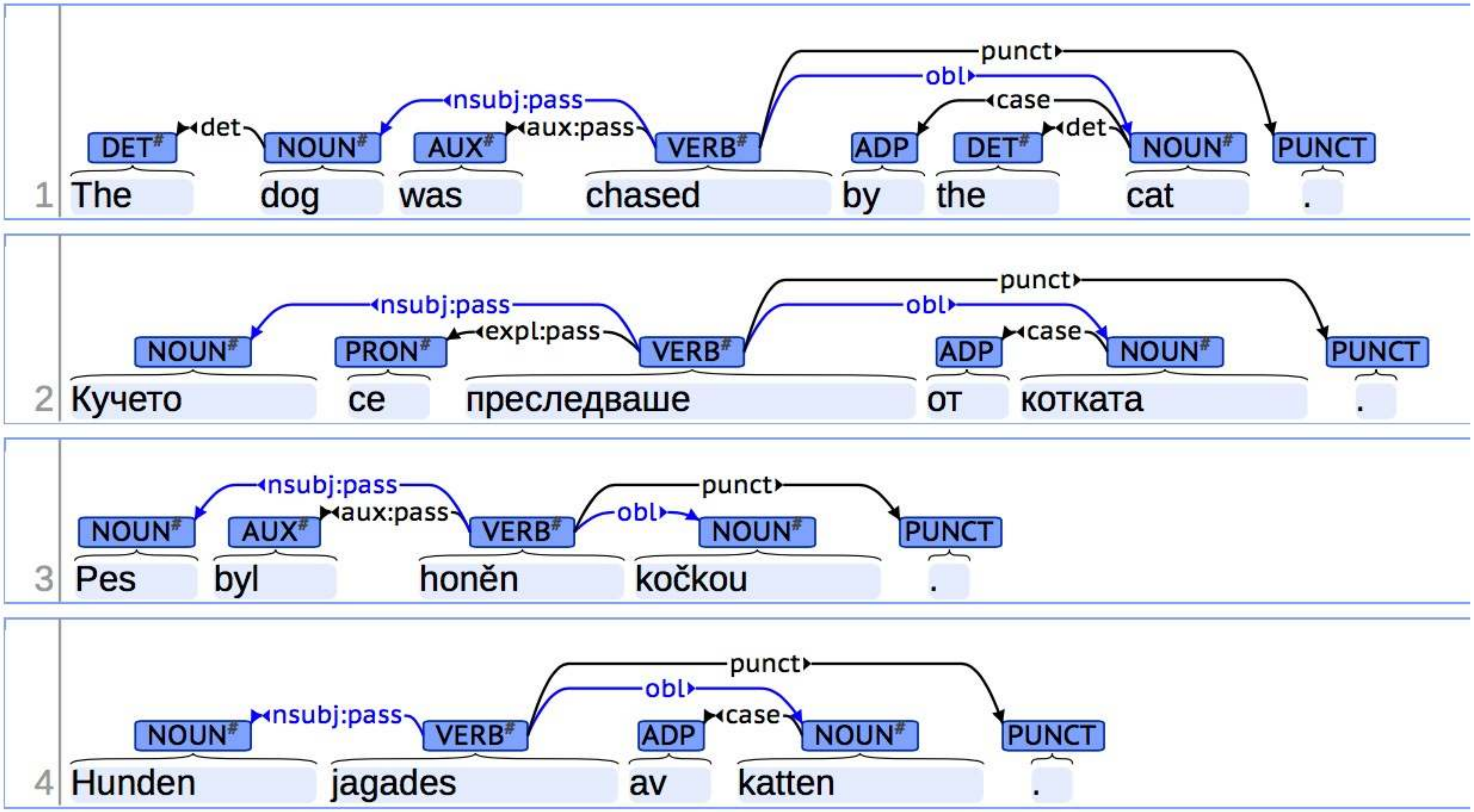
CoNLL-X shared task (2006)

Data sets. Tok = number of tokens (×1000); Sen = number of sentences (×1000); T/S = tokens per sentence (mean); Lem = lemmatization present; CPoS = number of coarse-grained part-of-speech tags; PoS = number of (fine-grained) part-of-speech tags; MSF = number of morphosyntactic features (split into atoms); Dep = number of dependency types; NPT = proportion of non-projective dependencies/tokens (%); NPS = proportion of non-projective dependency graphs/sentences (%).

Language	Tok	Sen	T/S	Lem	CPoS	PoS	MSF	Dep	NPT	NPS
Arabic	54	1.5	37.2	yes	14	19	19	27	0.4	11.2
Bulgarian	190	14.4	14.8	no	11	53	50	18	0.4	5.4
Chinese	337	57.0	5.9	no	22	303	0	82	0.0	0.0
Czech	1,249	72.7	17.2	yes	12	63	61	78	1.9	23.2
Danish	94	5.2	18.2	no	10	24	47	52	1.0	15.6
Dutch	195	13.3	14.6	yes	13	302	81	26	5.4	36.4
German	700	39.2	17.8	no	52	52	0	46	2.3	27.8
Japanese	151	17.0	8.9	no	20	77	0	7	1.1	5.3
Portuguese	207	9.1	22.8	yes	15	21	146	55	1.3	18.9
Slovene	29	1.5	18.7	yes	11	28	51	25	1.9	22.2
Spanish	89	3.3	27.0	yes	15	38	33	21	0.1	1.7
Swedish	191	11.0	17.3	no	37	37	0	56	1.0	9.8
Turkish	58	5.0	11.5	yes	14	30	82	25	1.5	11.6

[Nivre and McDonald 2011]

Universal Dependencies



http://universaldependencies.org/introduction.html

Two Approaches

- Graph-based parsing
  - finds an entire tree among all possible trees
  - with globally optimized models

- Transition-based parsing
  - greedily adds an arc step by step to make a tree
  - with locally optimized models

R. McDonald, J. Nivre, Computational Linguistics, 2011

- Analyze these two kinds of parsers
  - Actually, both obtain similar parsing accuracies

Language	Graph-based	Transition-based
Arabic	66.91	66.71
Bulgarian	87.57	87.41
Chinese	85.90	86.92
…	…	…
Average	**80.83**	**80.75**

Notation

- Let $L = \{l_1, \dots, l_{|L|}\}$ be arc labels

- Let $x = w_0, w_1, \dots, w_n$ be an input sentence
  - where $w_0 = \mathrm{ROOT}$

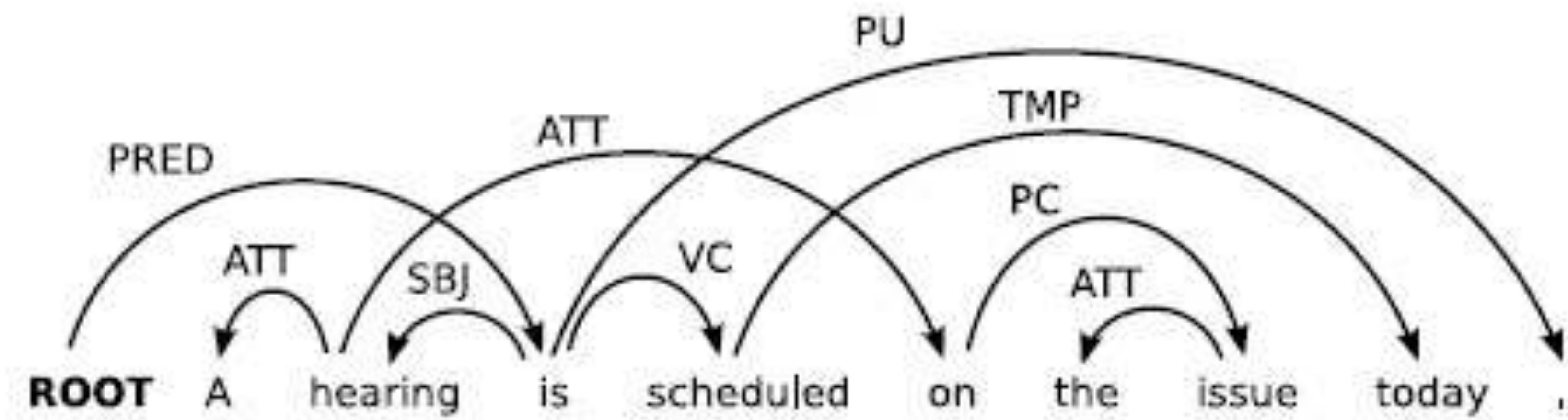


Table of Contents

- Dependency formalism
- Graph-based parsing
- Transition-based parsing
- Japanese dependency parsing

Notation

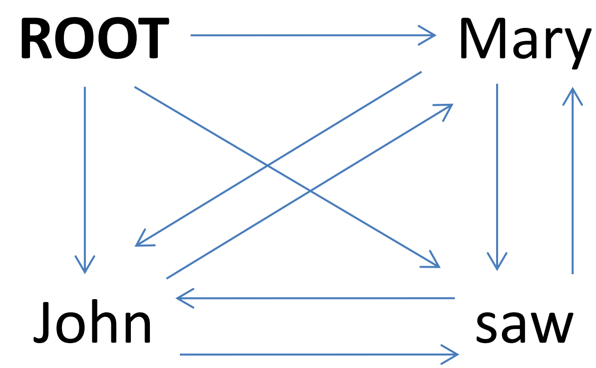
- Dependency graph/tree: $G = (V, A)$
  - V : a set of nodes (vertices)
  - A : a set of arcs (directed edges)
    - A dependency: $(i, j, l) \in A$
  - a linear precedence order $<$ on V (word order)
- Conditions on dependency graphs
  - G is connected
    - if $i, j \in V$ then $i \leftrightarrow^* j$
  - G is acyclic
    - if $i \rightarrow j$ then not $j \rightarrow^* i$
  - G obeys the single-head constraint
    - if $i \rightarrow j$ then not $i' \rightarrow j$ for any $i' \neq i$

Graph-based Parsing

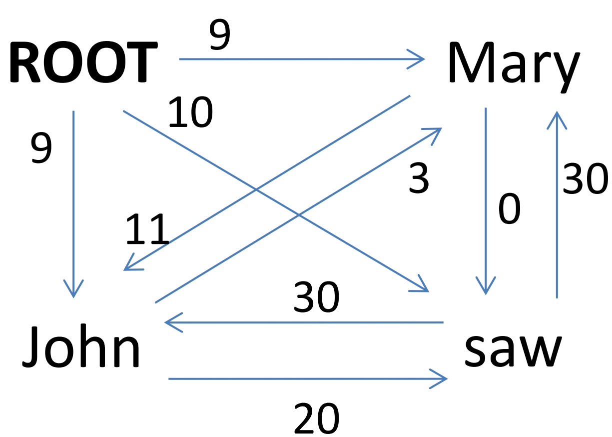
Input sentence

x =John saw Mary

Make dense graph

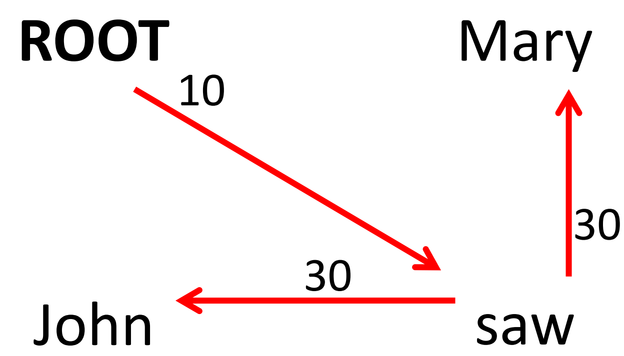


Score each arc
by some scoring function



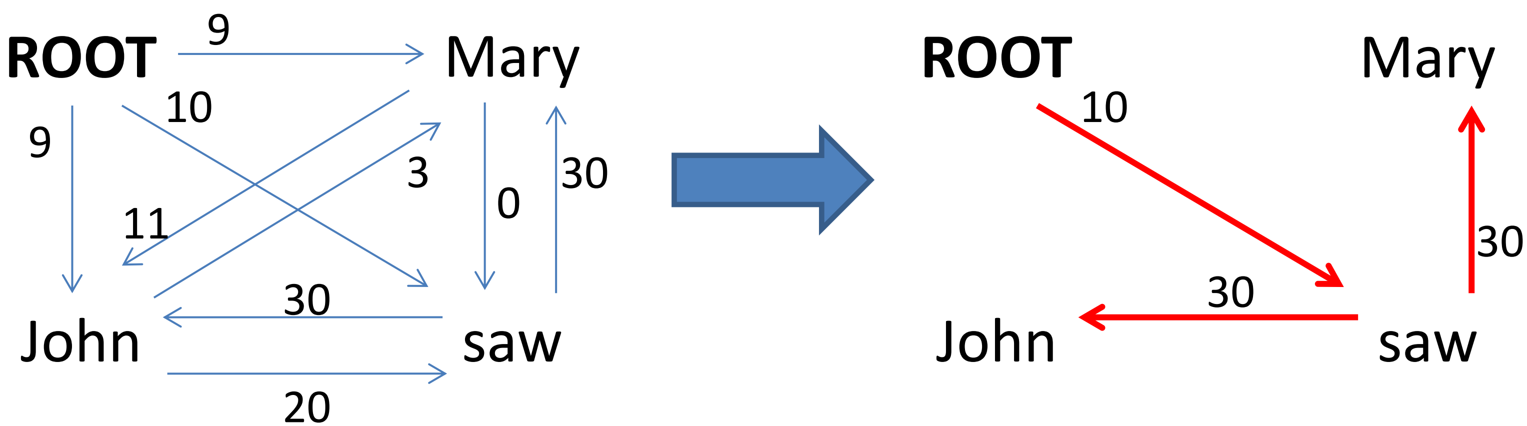
Find the tree
that maximizes the
sum of the arc scores

Output dependency tree



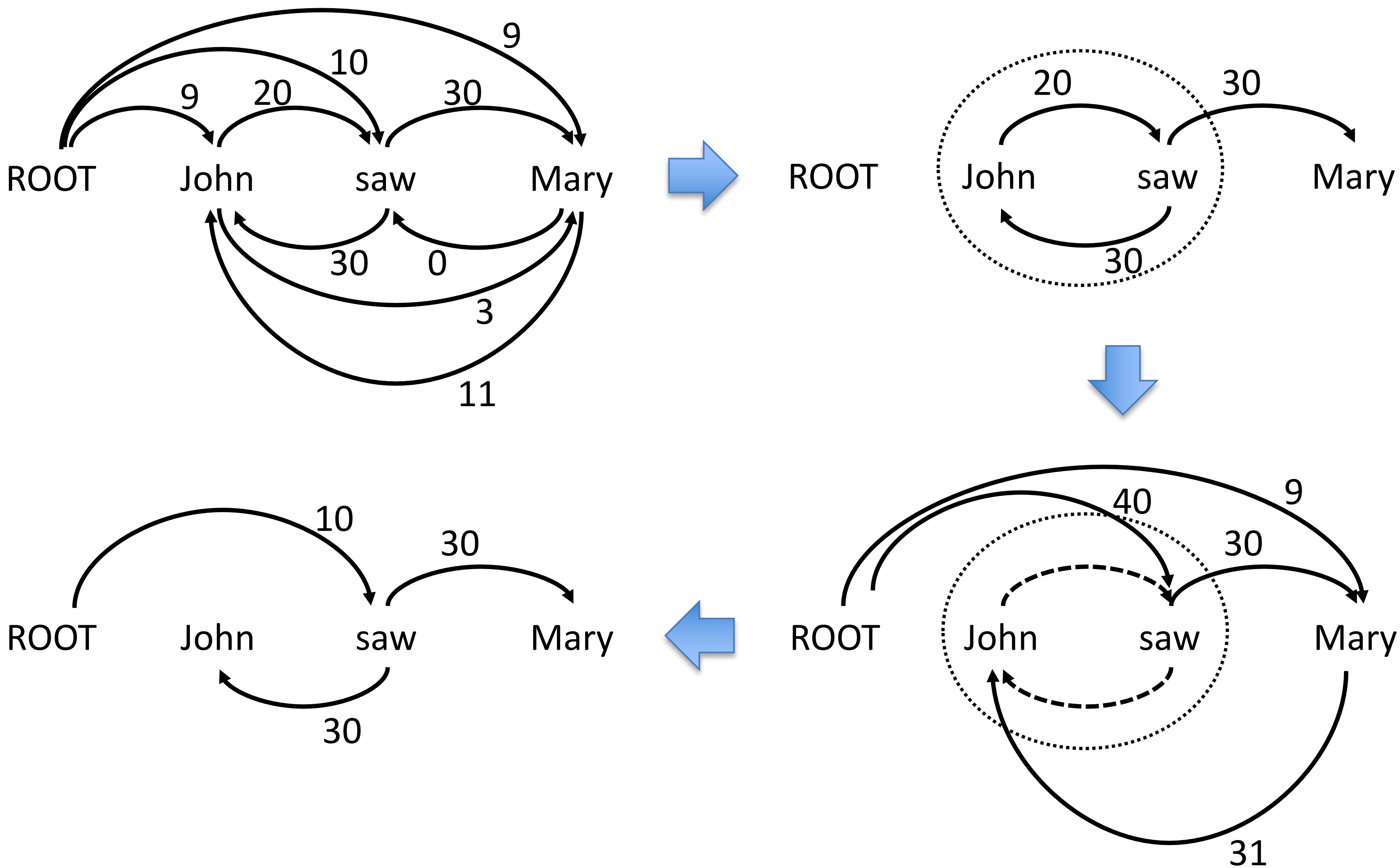
MST: Maximum Spanning Tree

- The last step is finding the tree that …
  - has all the nodes of the dense graph
  - maximizes the sum of the arc scores



- This is a **maximum spanning tree problem**
  - $O(n^2)$ algorithm by [Chu and Liu 1965] [Edmonds 1967]
  - Do exhaustive search quickly

Chu-Liu-Edmonds Algorithm



Practice

Use the Chu-Liu-Edmonds algorithm to find the dependency structure of “boys often play games” and its score.

Y

X

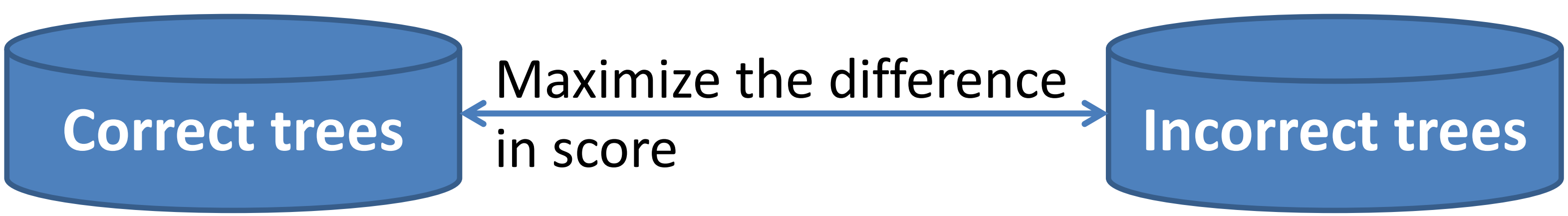
X→Y	boys	often	play	games
ROOT	8	5	8	6
boys		2	2	1
often	2		9	2
play	9	8		10
games	0	1	1	

Learning an Arc Scoring Function

- Target: dependency arc scoring function s
  - s: $(i, j, l) \rightarrow s(i, j, l) \in \mathbb{R}$
    - (i, j, l): arc of dependency $w_i \rightarrow w_j$ with label l
  - $s(i, j, l)$ is often defined as $\boldsymbol{w} * \boldsymbol{f}(i, j, l)$

Feature vector of arc

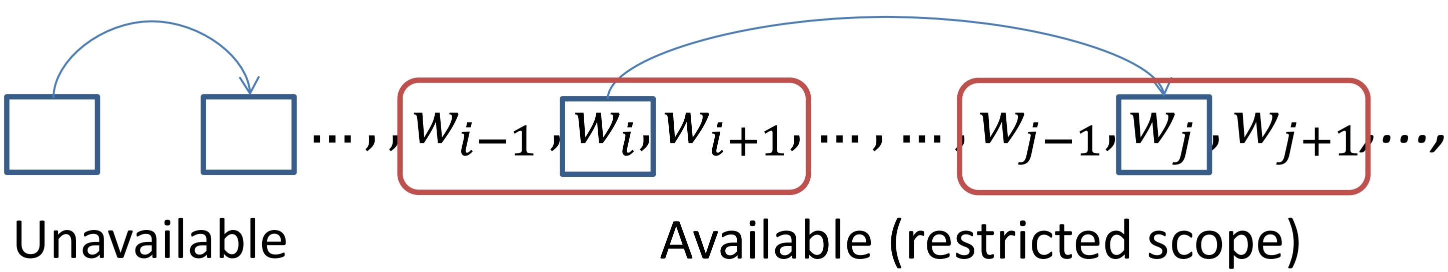
- Optimize parameters to maximize the difference in score between correct/incorrect trees



Characterization of Graph-based Approach

- The learning procedure is **<u>global</u>** because …
  - optimizing the global score of an entire tree
  - not just over single arc attachment decisions

- Restricted scope of feature sets for $\boldsymbol{f}(i, j, l)$
  - e.g., lexical and surface syntactic features



Training MST Parser

- Two-stage approach
  - First predict arcs, then arc labels

1. Arc score $s(i,j) = \boldsymbol{w} * \boldsymbol{f}(i,j)$
   - Labels are ignored
   - Online large-margin training algorithm

2. Label score $s(l|i,j) = \boldsymbol{w} * \boldsymbol{f}(i,j,l)$
   - A label is conditioned on a fixed arc (i, j)
   - Log-linear arc-labeler

Features of MSTParser

Features for MSTParser. $\wedge$ indicates a conjunction of features. † indicates that all back-off versions of a conjunction feature are included as well. A back-off version of a conjunction feature is one where one or more base features are disregarded. ‡ indicates that all back-off versions are included where a single base feature is disregarded.

Base features for sentence: $x = w_0, w_1, \ldots, w_n$

Lexical features: Identity of w_i, $w_i \in x$

Affix features: 3-gram lexical prefix/suffix identity of $\mathrm{Pref}(w_i)/\mathrm{Suff}(w_i)$, $w_i \in x$

Part-of-speech features: Identity of $\mathrm{PoS}(w_i)$, $w_i \in x$

Morphosyntactic features: For all morphosyntactic features MSF_k for a word w_i, identity of $\mathrm{MSF}_k(w_i)$, $w_i \in x$

Label features: Identity of l in some labeled arc (i, j, l)

(a) Head-modifier features for unlabeled arc (i, j)

$w_i \wedge \mathrm{PoS}(w_i) \wedge w_j \wedge \mathrm{PoS}(w_j)$ †

$\mathrm{Pref}(w_i) \wedge \mathrm{PoS}(w_i) \wedge \mathrm{Pref}(w_j) \wedge \mathrm{PoS}(w_j)$ †

$\mathrm{Suff}(w_i) \wedge \mathrm{PoS}(w_i) \wedge \mathrm{Suff}(w_j) \wedge \mathrm{PoS}(w_j)$ †

$\forall k, k' : \mathrm{MSF}_k(w_i) \wedge \mathrm{PoS}(w_i) \wedge \mathrm{MSF}_{k'}(w_j) \wedge \mathrm{PoS}(w_j)$ †

(b) PoS-context features for unlabeled arc (i, j)

$\forall k, i < k < j : \mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_k) \wedge \mathrm{PoS}(w_j)$

$\mathrm{PoS}(w_{i-1}) \wedge \mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_{j-1}) \wedge \mathrm{PoS}(w_j)$ ‡

$\mathrm{PoS}(w_{i-1}) \wedge \mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_j) \wedge \mathrm{PoS}(w_{j+1})$ ‡

$\mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_{i+1}) \wedge \mathrm{PoS}(w_{j-1}) \wedge \mathrm{PoS}(w_j)$ ‡

$\mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_{i+1}) \wedge \mathrm{PoS}(w_j) \wedge \mathrm{PoS}(w_{j+1})$ ‡

(c) Head-modifier features for unlabeled arc pair $(i, j \diamond k)$

$w_j \wedge w_k$

$w_j \wedge \mathrm{PoS}(w_k)$

$\mathrm{PoS}(w_j) \wedge w_k$

$\mathrm{PoS}(w_j) \wedge \mathrm{PoS}(w_k)$

$\mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_j) \wedge \mathrm{PoS}(w_k)$

(d) Arc-label features for labeled arc (i, j, l)

$w_i \wedge \mathrm{PoS}(w_i) \wedge w_j \wedge \mathrm{PoS}(w_j) \wedge l$ †

$\forall k, i < k < j : \mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_k) \wedge \mathrm{PoS}(w_j) \wedge l$

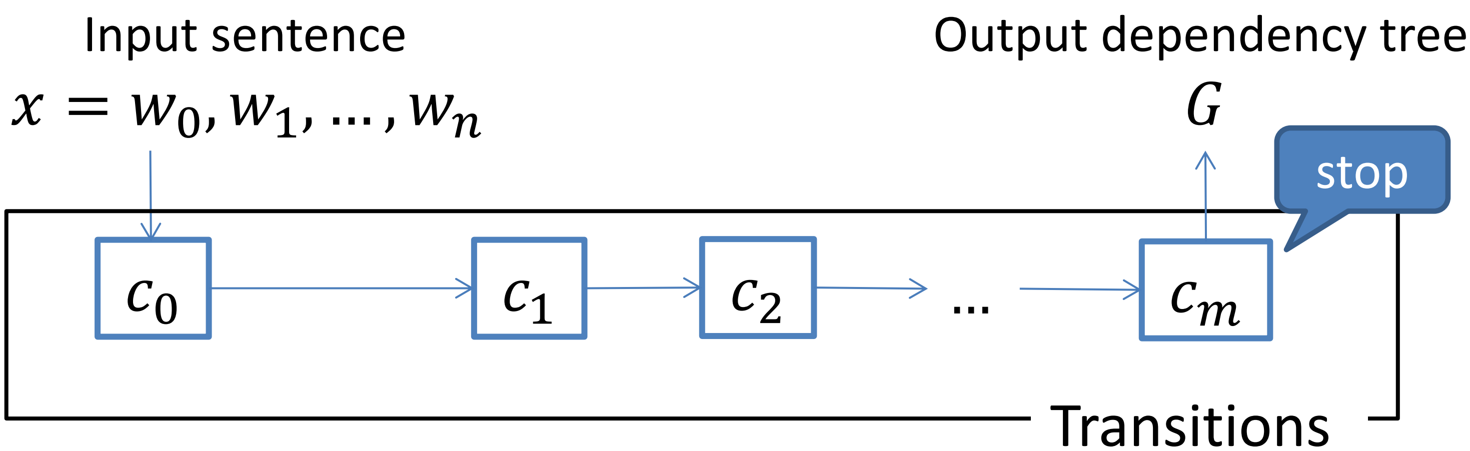
$\mathrm{PoS}(w_{j-1}) \wedge \mathrm{PoS}(w_j) \wedge \mathrm{PoS}(w_{j+1}) \wedge l$ †

$\mathrm{PoS}(w_{i-1}) \wedge \mathrm{PoS}(w_i) \wedge \mathrm{PoS}(w_{i+1}) \wedge l$ †

Table of Contents

Transition-based Parsing

- Parsing based on transitions

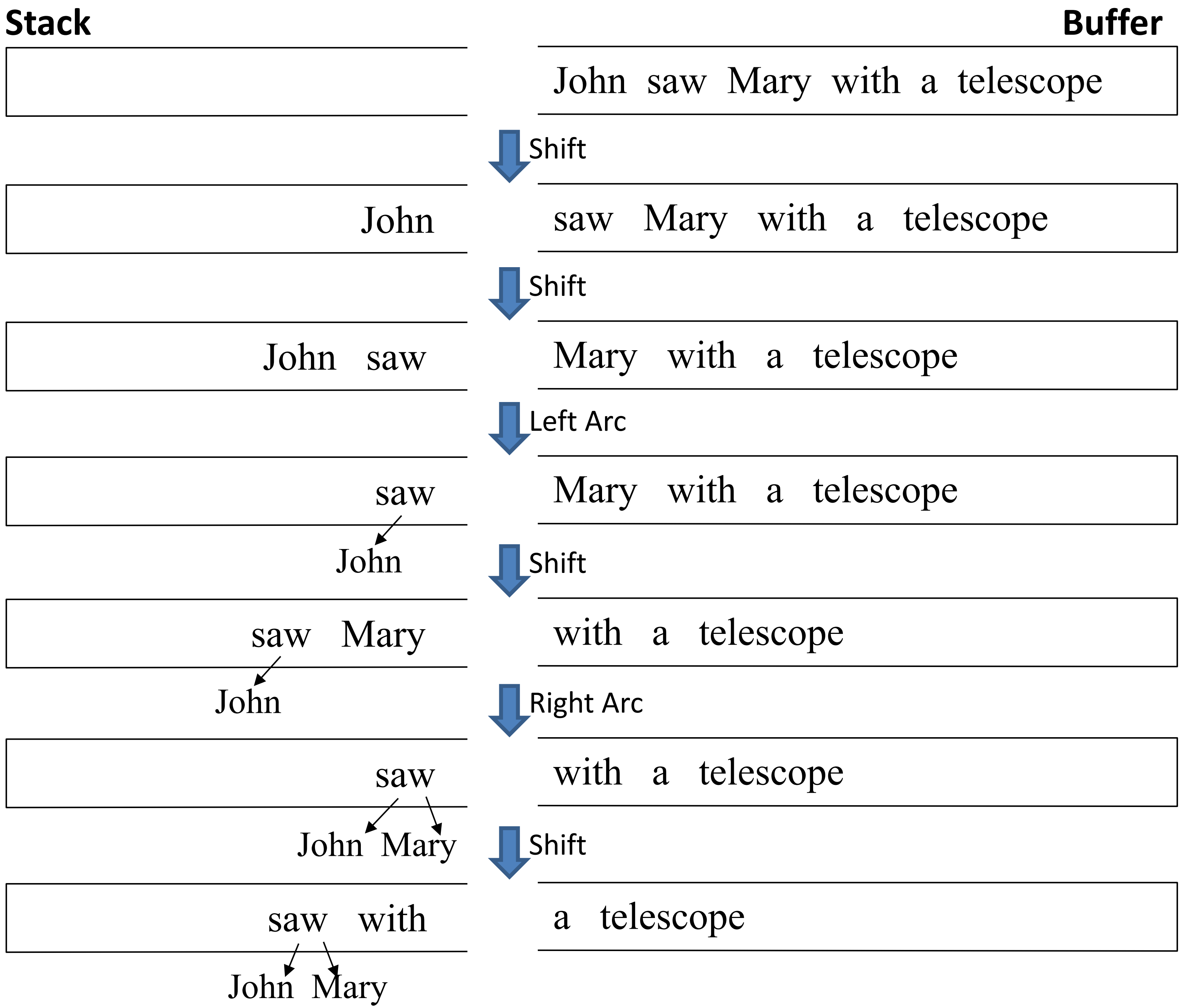


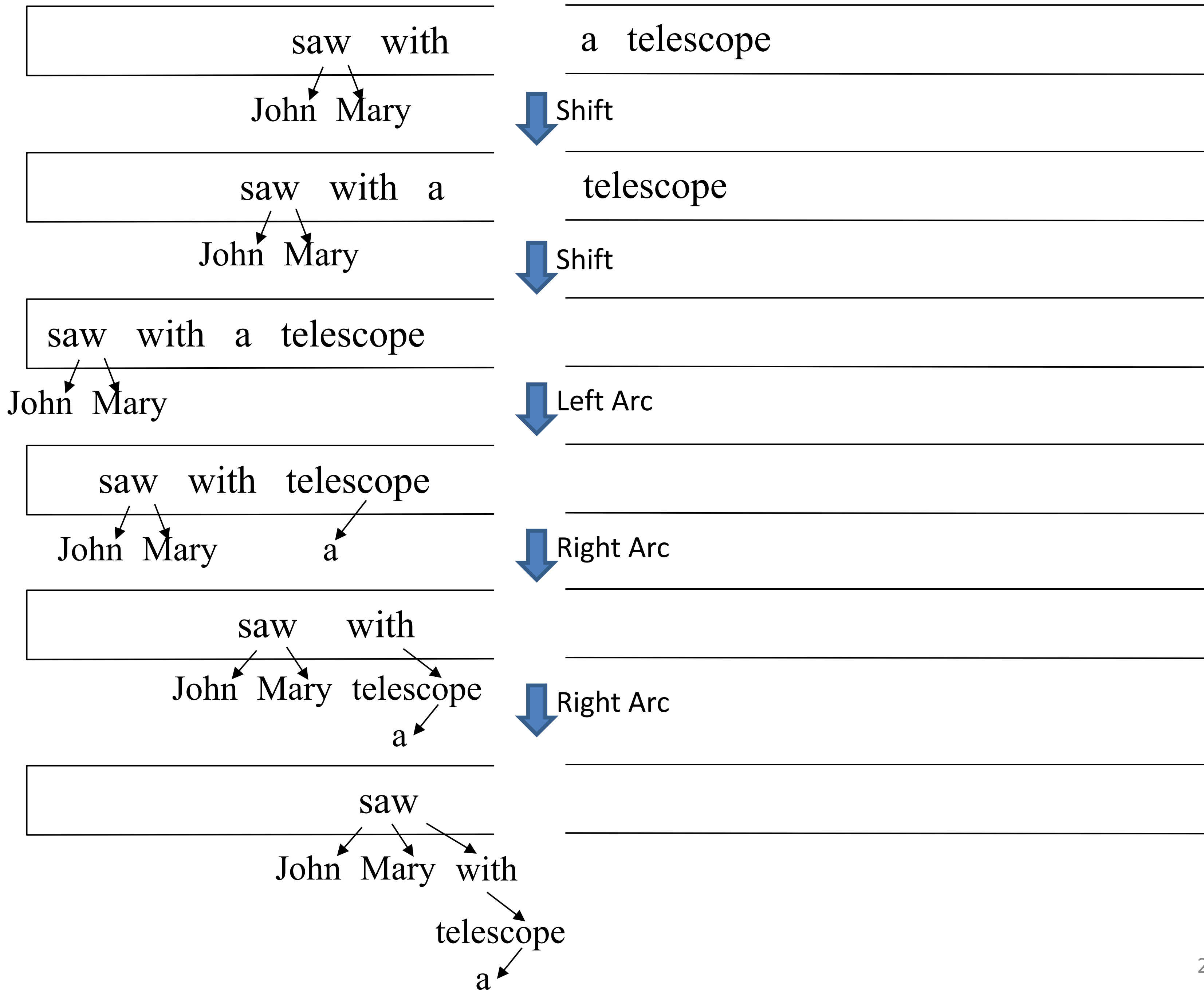
- Building the output dependency tree step by step
  - Each c_i defines a partially built dependency graph
  - The last c_m defines the output dependency tree

Transition-based Parsing

- State: triple
  - σ: stack of partially processed words
  - β: buffer of remaining input words
  - A: set of labeled dependency arcs
- Transitions
  - Shift: move the first word in the buffer to the stack
  - Left Arc: remove w_i from the stack, with the dependency relation from w_i to w_j ($w_i \leftarrow w_j$)
  - Right Arc: remove w_j from the stack, with the dependency relation from w_j to w_i ($w_i \rightarrow w_j$)

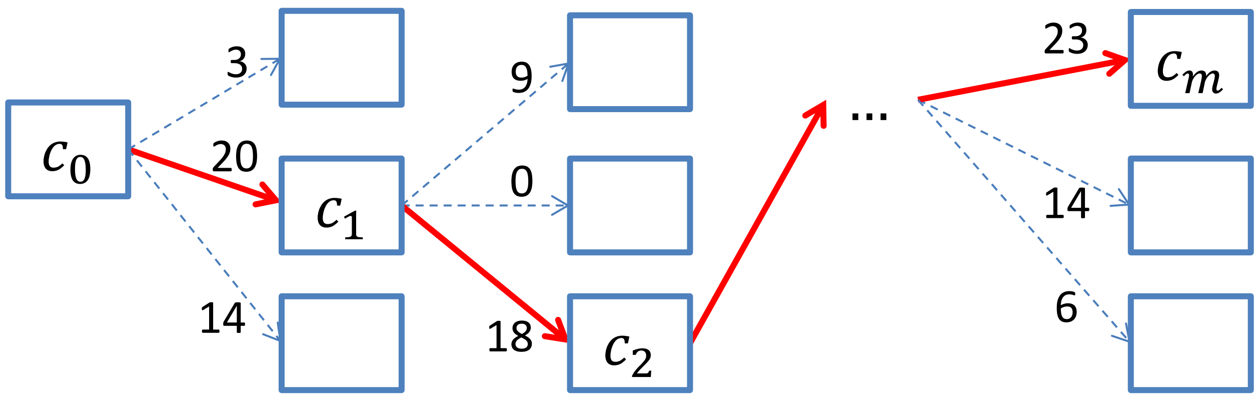
* w_i, w_j are the rightmost words in the stack


Stack
Buffer
John saw Mary with a telescope
Shift
John
saw Mary with a telescope
Shift
John saw
Mary with a telescope
Left Arc
saw
John
Mary with a telescope
Shift
saw Mary
John
with a telescope
Right Arc
saw
John Mary
with a telescope
Shift
saw with
John Mary
a telescope


saw with
a telescope
John Mary
Shift
saw with a
telescope
John Mary
Shift
saw with a telescope
John Mary
Left Arc
saw with telescope
John Mary a
Right Arc
saw with
John Mary telescope
a
Right Arc
saw
John Mary with
telescope
a

Transition-based Parsing

- Use some transition scoring function to choose next transition



- Repeat taking the optimal transition at each step
  - Greedy search of $O(n)$

Learning a Transition Scoring Function

- Target: transition scoring function s
  - $s\colon (c,t) \rightarrow s(c,t) \in \mathbb{R}$
    - c: current state
    - t: transition that will be scored
  - Transition set is finite → classification problem

- Discriminative learning methods (such as SVMs)
  - Training data: history of states and gold standard transitions

$$c_i \xrightarrow{t} c_{i+1}$$

Characterization of Transition-based Approach

- The learning procedure is **<u>local</u>**
  - only single transitions are scored
  - not entire transition sequences

- Rich feature sets
  - e.g., the entire dependency graph built so far

$$\ldots \rightarrow c_{i-1} \rightarrow \boxed{c_i} \xrightarrow{t} c_{i+1}$$

Available (dependency graph)

- Greedy search may lead to error propagation
  - False early predictions may eliminate correct trees

Training Malt Parser

- $c = (\sigma_c, \beta_c, A_c)$: current state
  - σ_c^i: i-th element from the top of stack σ_c
  - β_c^i: i-th element from the head of buffer β_c

- Features:
  - Pos(w), $w \in \{\sigma_c^0, \sigma_c^1, \beta_c^0, \beta_c^1, \beta_c^2, \beta_c^3\}$
  - $w, w \in \{\sigma_c^0, \beta_c^0, \beta_c^1\}$ or $(\sigma_c^0, w, l) \in A_c$
  - $l, (w, w', l) \in A_c$ and $w \in \{\sigma_c^0, \sigma_c^1\}$

Comparison

- Training algorithms

	MST (graph-based)	Malt (transition-based)
Algorithm	Large-margin learning (Online algorithm)	Large-margin learning (Support Vector Machines)
Model	Globally trained	Locally trained

- Feature representation

MST (graph-based)	Malt (transition-based)
Restricted, local features (Neighboring words and POS tags)	Rich, global features (History of previous decisions)

Comparison

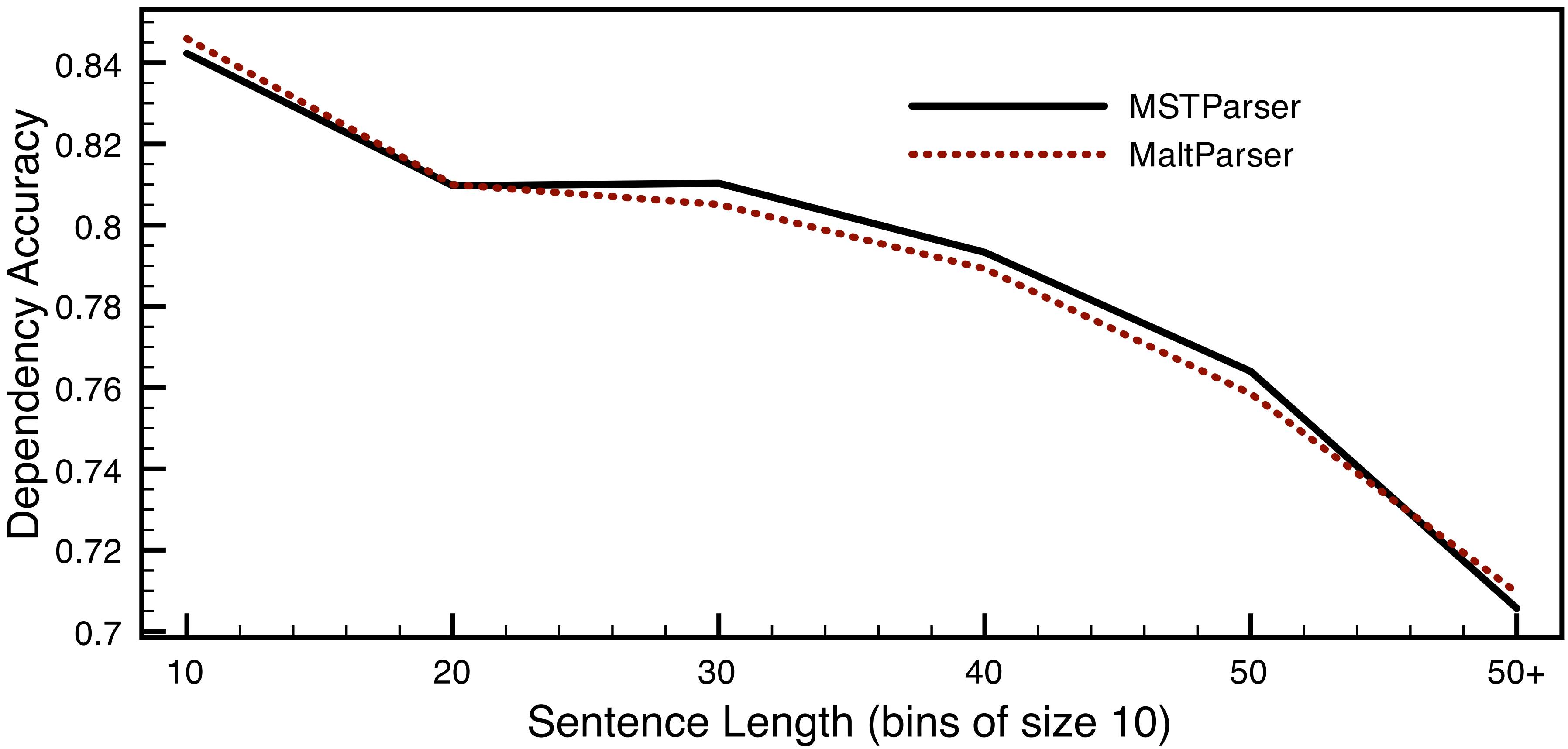
- Inference
  - Malt is far quicker: $O(n)$ vs. $O(n^2)$
  - Malt may cause error propagation

Exhaustive inference algorithm & global learning

Trade-off

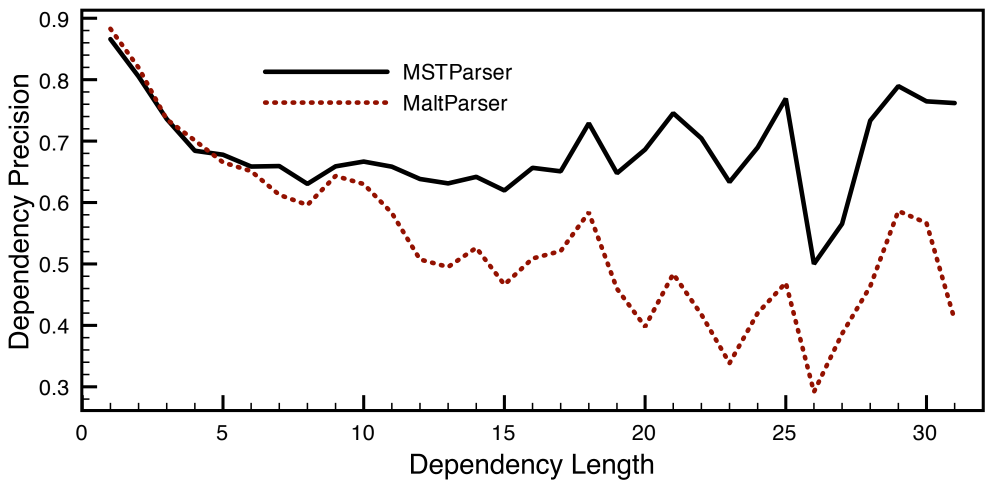
Expressiveness of feature representation

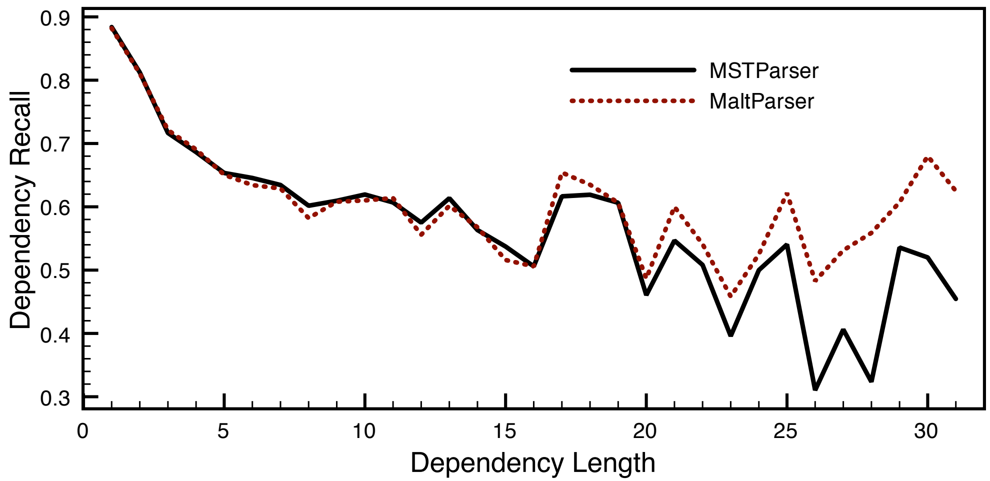
Sentence Length



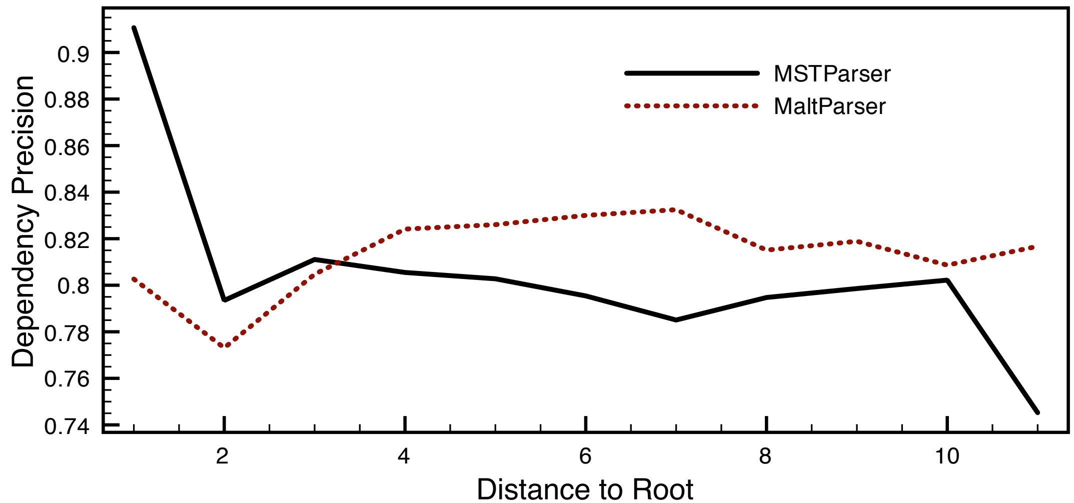
[McDonald and Nivre 2011]

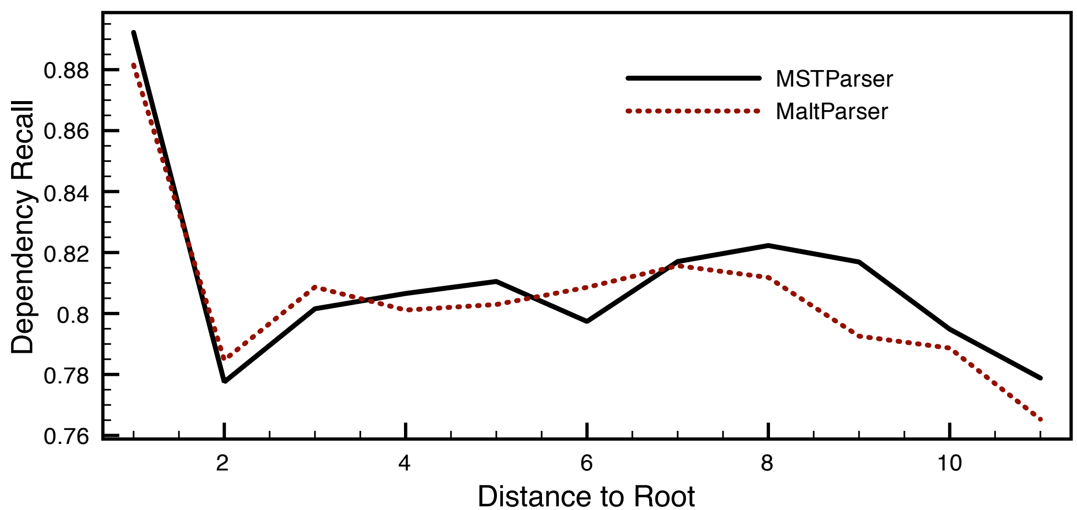
Dependency Length



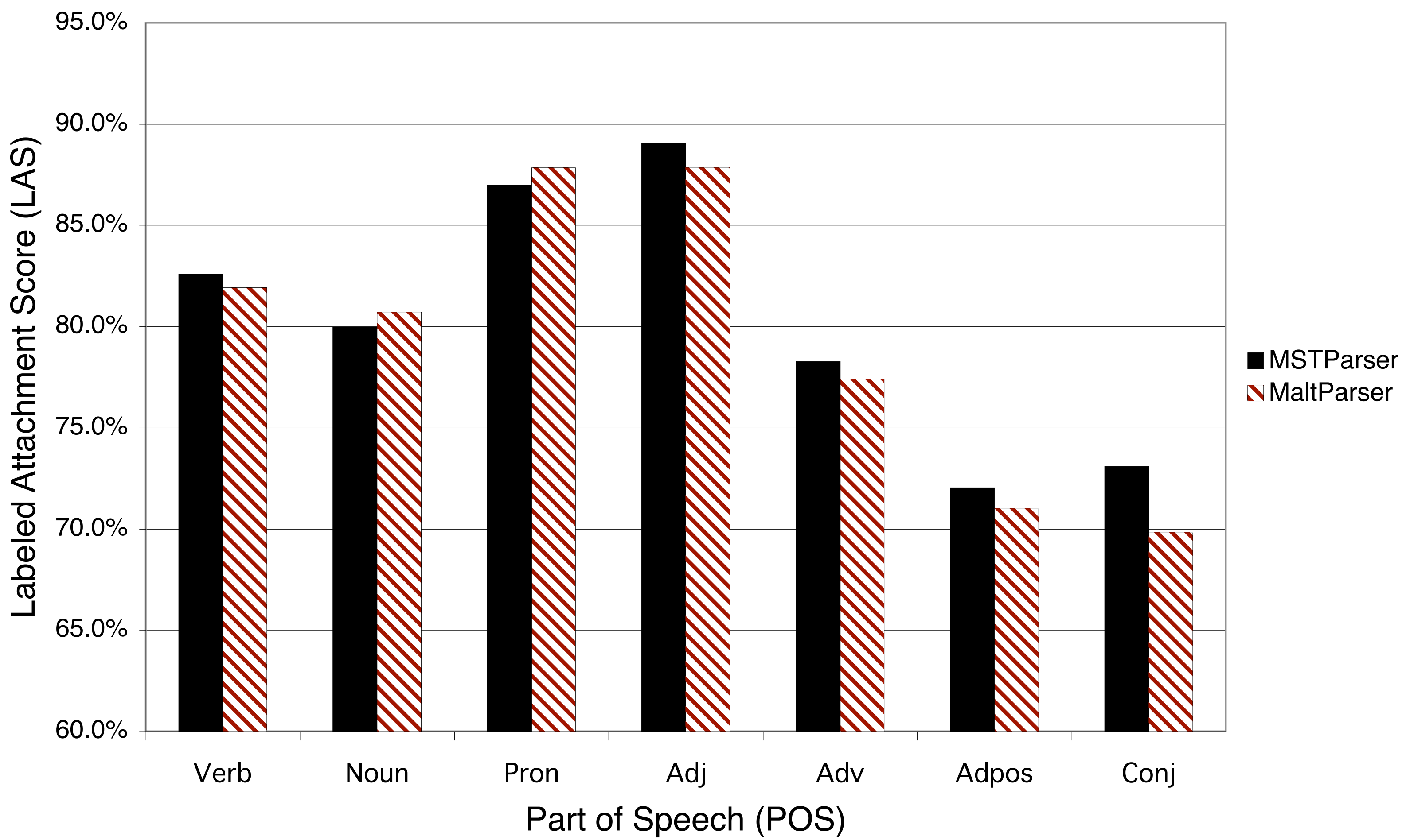


Tree Depth (Distance to Root)

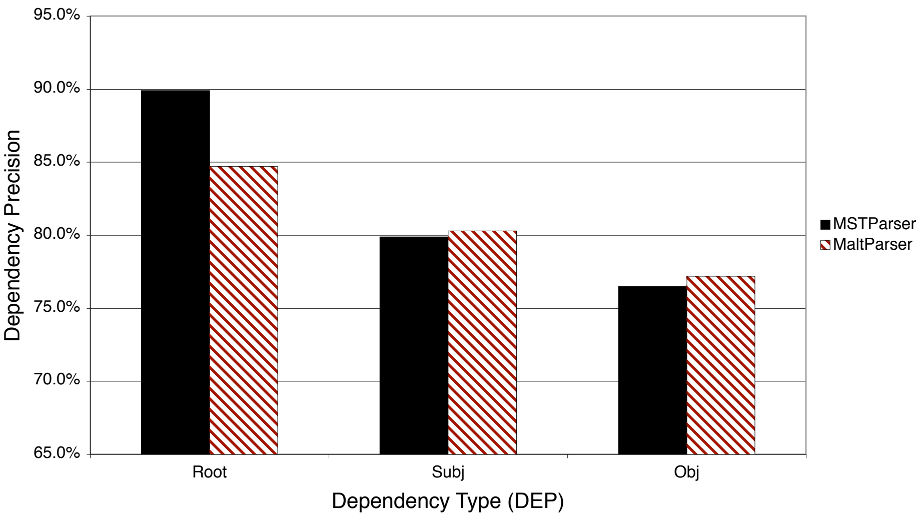


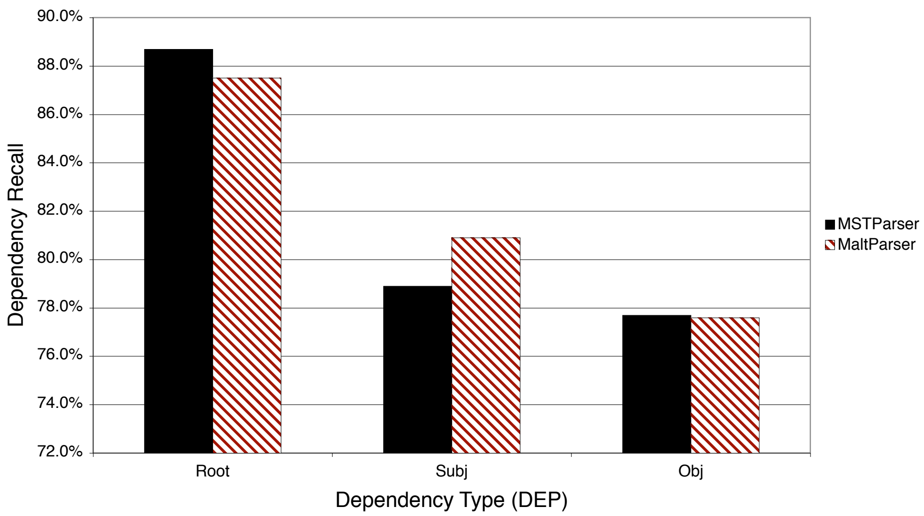


Part of Speech of Dependents



Dependency Type: Root, Subject, Object



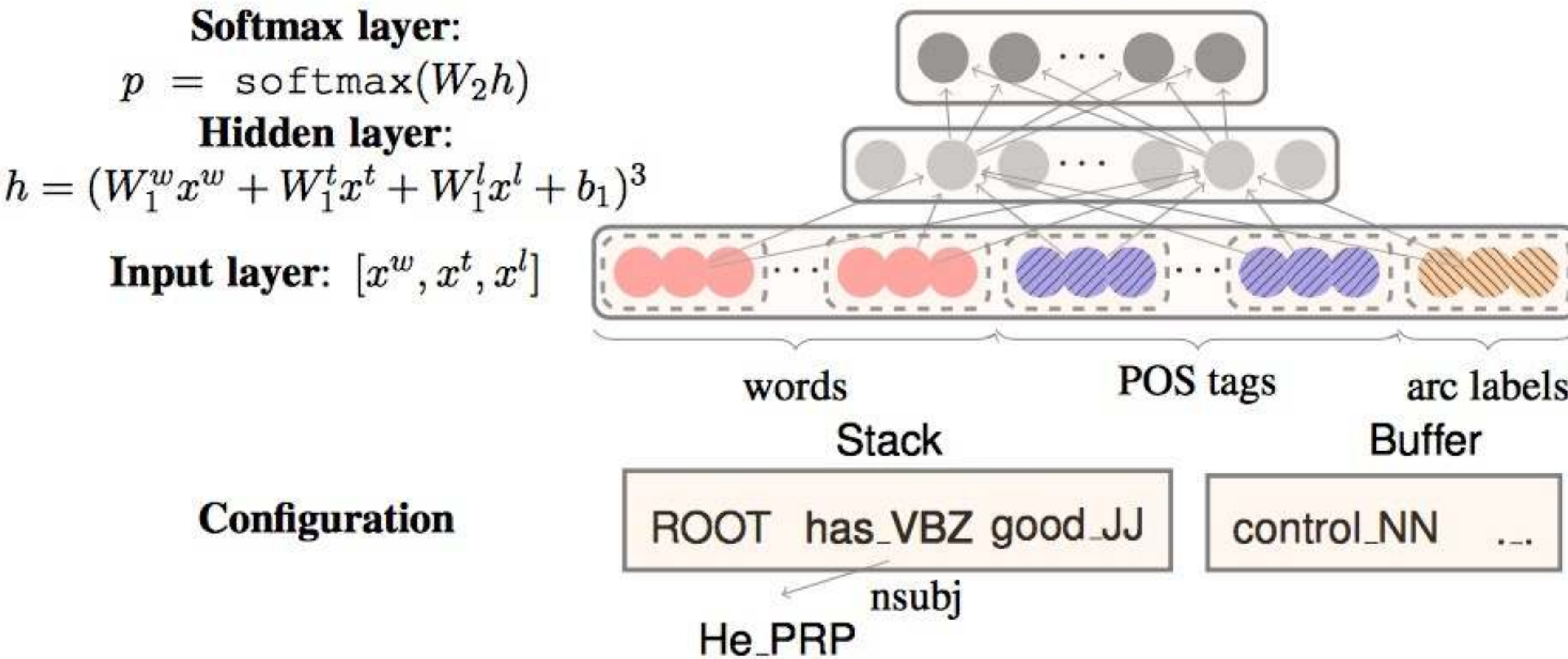


Phrase Structure vs. Dependency Structure

- Phrase structure
  - Phrases (nonterminal nodes)
  - Functional categories (functional labels)
  - Structural categories (nonterminal labels)

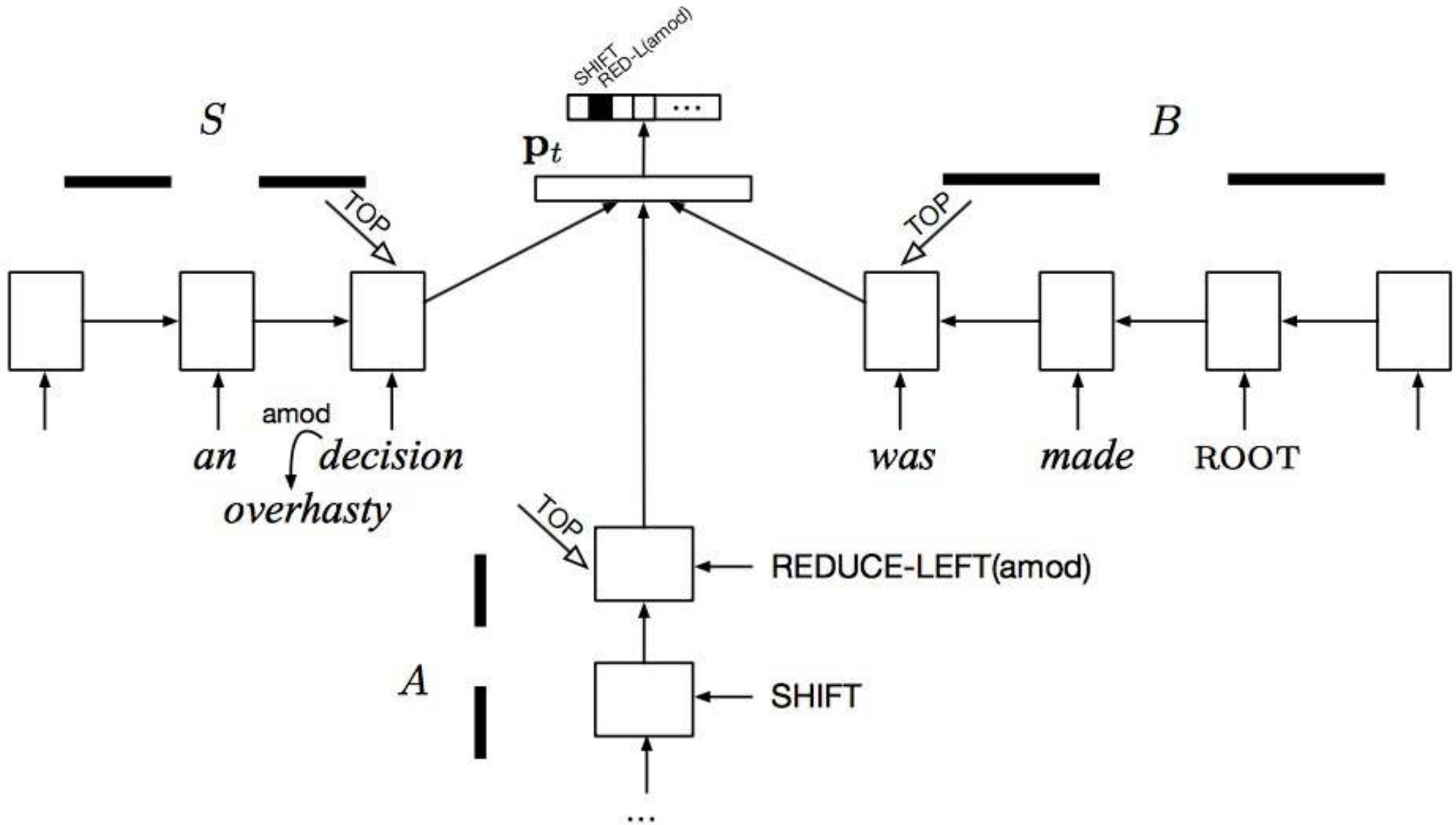
- Dependency structure
  - Head-modifier relations (directed arcs)
  - Functional categories (arc labels)
  - No structural categories
  - Easy to convert to predicate-argument structures

Neural Network-based Dependency Parsing



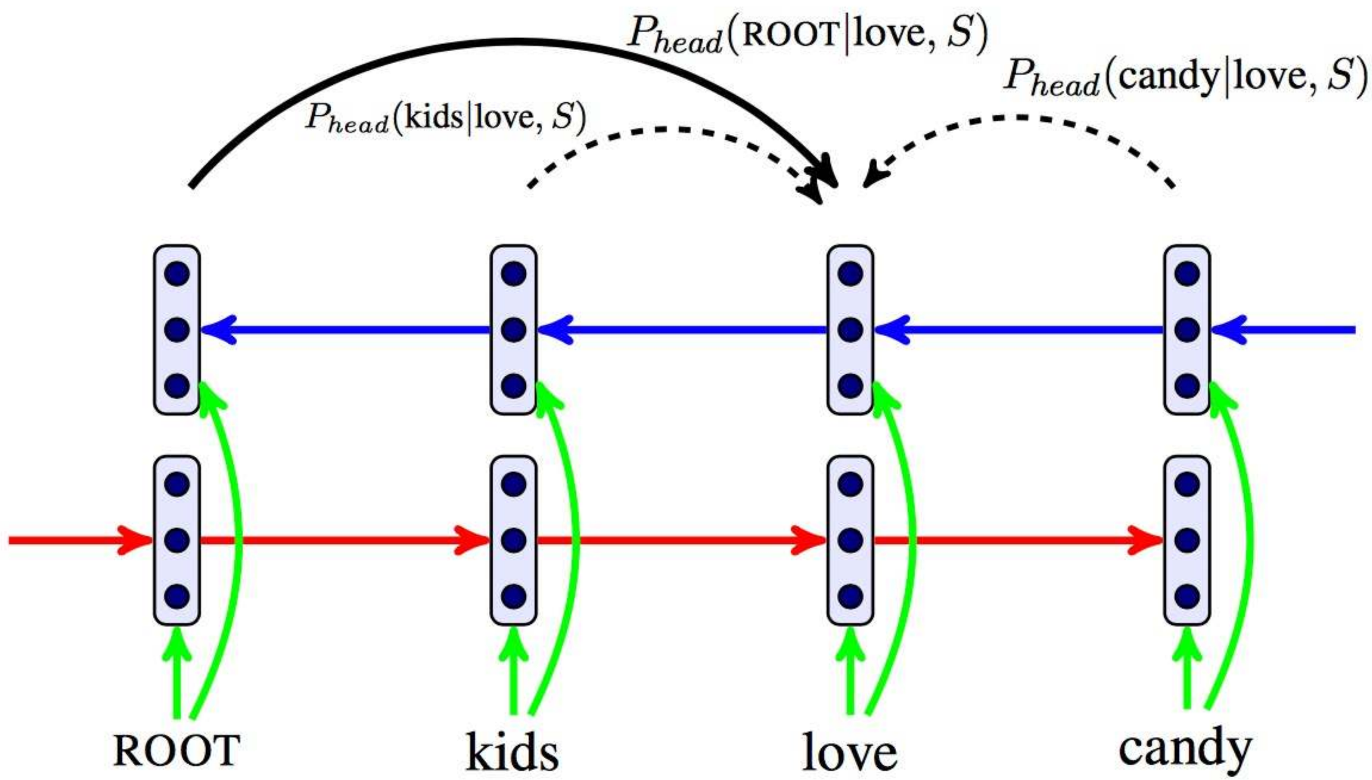
[Chen and Manning 2014]

Transition-based Model with Stack LSTM



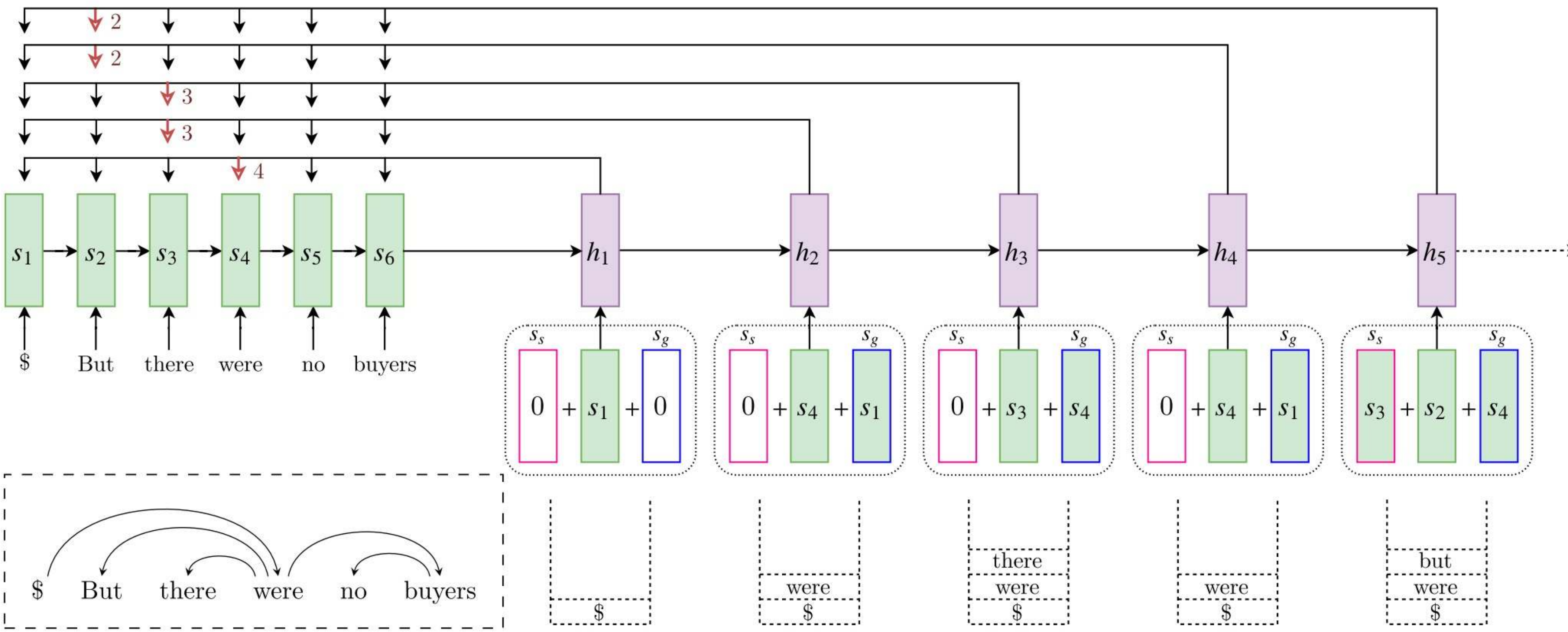
[Dyer+ 2015]

Head Selection



[Zhang+ 2017]

Stack-Pointer Networks



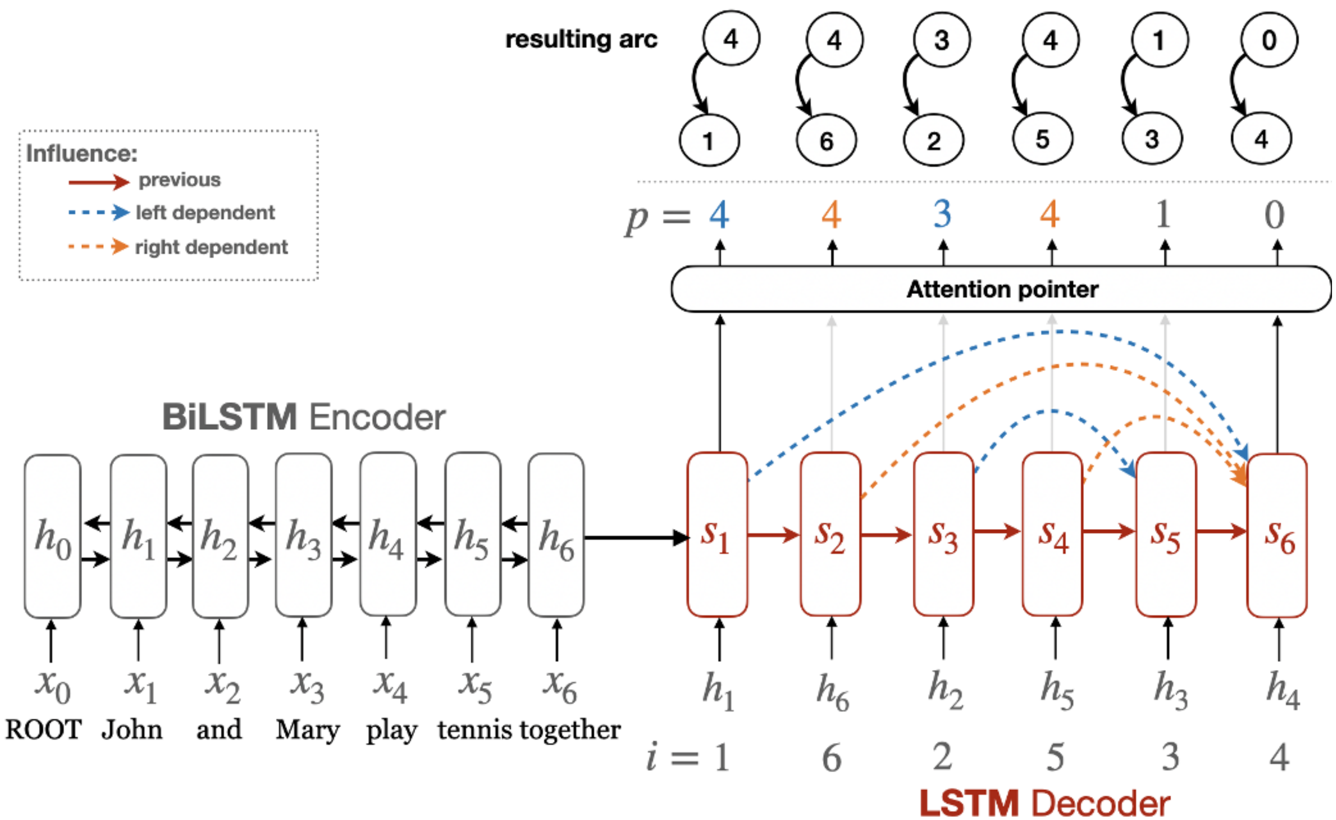
[Ma+ 2018]

Stack-Pointer Networks

System		English		Chinese		German	
		UAS	LAS	UAS	LAS	UAS	LAS
Chen and Manning (2014)	T	91.8	89.6	83.9	82.4	–	–
Ballesteros et al. (2015)	T	91.63	89.44	85.30	83.72	88.83	86.10
Dyer et al. (2015)	T	93.1	90.9	87.2	85.7	–	–
Bohnet and Nivre (2012)	T	93.33	91.22	87.3	85.9	91.4	89.4
Ballesteros et al. (2016)	T	93.56	91.42	87.65	86.21	–	–
Kiperwasser and Goldberg (2016)	T	93.9	91.9	87.6	86.1	–	–
Weiss et al. (2015)	T	94.26	92.41	–	–	–	–
Andor et al. (2016)	T	94.61	92.79	–	–	90.91	89.15
Kiperwasser and Goldberg (2016)	G	93.1	91.0	86.6	85.1	–	–
Wang and Chang (2016)	G	94.08	91.82	87.55	86.23	–	–
Cheng et al. (2016)	G	94.10	91.49	88.1	85.7	–	–
Kuncoro et al. (2016)	G	94.26	92.06	88.87	87.30	91.60	89.24
Ma and Hovy (2017)	G	94.88	92.98	89.05	87.74	92.58	90.54
BIAF: Dozat and Manning (2017)	G	95.74	94.08	89.30	88.23	93.46	91.44
BIAF: re-impl	G	95.84	**94.21**	90.43	89.14	**93.85**	**92.32**
STACKPTR: Org	T	95.77	94.12	90.48	89.19	93.59	92.06
STACKPTR: +gpar	T	95.78	94.12	90.49	89.19	93.65	92.12
STACKPTR: +sib	T	95.85	94.18	90.43	89.15	93.76	92.21
STACKPTR: Full	T	**95.87**	94.19	**90.59**	**89.29**	93.65	92.11

[Ma+ 2018]

Bottom-up Hierarchical Pointer Networks



Parser	PTB UAS	PTB LAS
Zhang et al. (2017)	94.10	91.90
Ma and Hovy (2017)	94.88	92.96
Dozat and Manning (2017)	95.74	94.08
Li et al. (2018)	94.11	92.08
Ma et al. (2018)	95.87	94.19
Ji et al. (2019)†	95.97	94.31
Fdez-G & Gómez-R (2019)	96.04	94.43
Li et al. (2020)	95.83	94.54
Fdez-G & Gómez-R (2020)	96.06	94.50
Zhang et al. (2020b)†	96.14	94.49
Wang and Tu (2020)	95.98	94.34
Hier. Ptr. Net. L2R	**96.18**	**94.59**
Hier. Ptr. Net. R2L	96.14	94.53
Hier. Ptr. Net. O-I	96.07	94.48
+BERT		
Li et al. (2020)	96.44	94.63
Li et al. (2020)*	96.57	95.05
Moham. & Hend. (2020)*	96.66	95.01
Wang and Tu (2020)*	96.91	95.34
Fdez-G & Gómez-R (2020)	96.91	95.35
Hier. Ptr. Net. L2R	**97.05**	95.47
Hier. Ptr. Net. R2L	97.01	**95.48**
Hier. Ptr. Net. O-I	96.95	95.36

Hierarchical Pointer Network with Outside-in Order

[Fernández-González and Gómez-Rodríguez 2021]

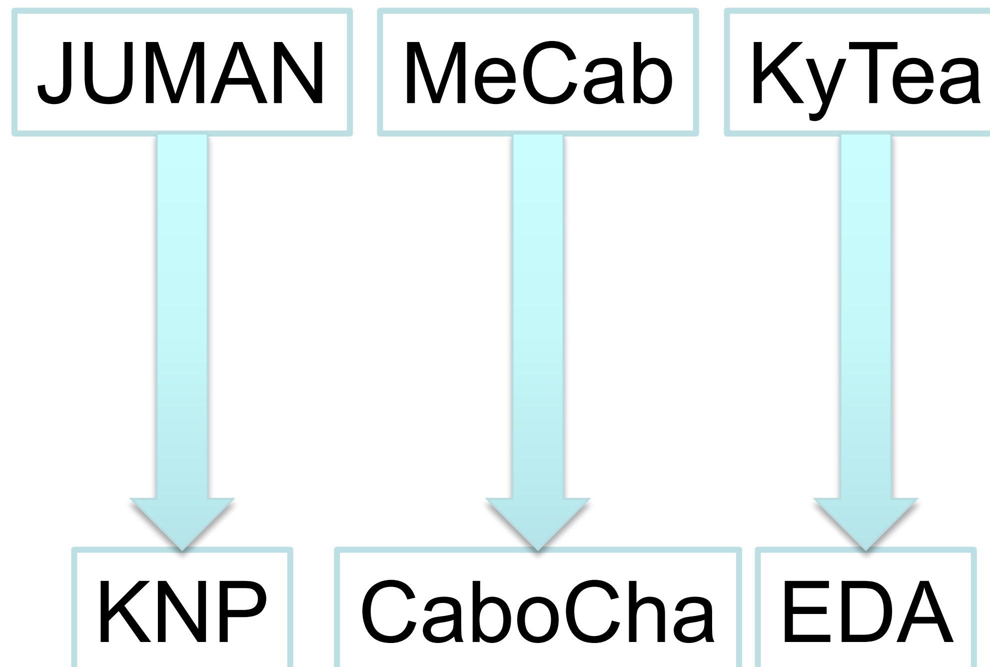
Table of Contents

Japanese Dependency Parsers

- KNP http://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP (In Japanese)
  - A probabilistic model based on case frames
  - Phrase dependency
- CaboCha http://code.google.com/p/cabocha/ (In Japanese)
  - Transition-based
  - Phrase dependency
  - SVMs
- EDA http://plata.ar.media.kyoto-u.ac.jp/tool/EDA/home_en.html
  - MST with pointwise edge score estimation
  - Word dependency

Japanese Dependency Parsers

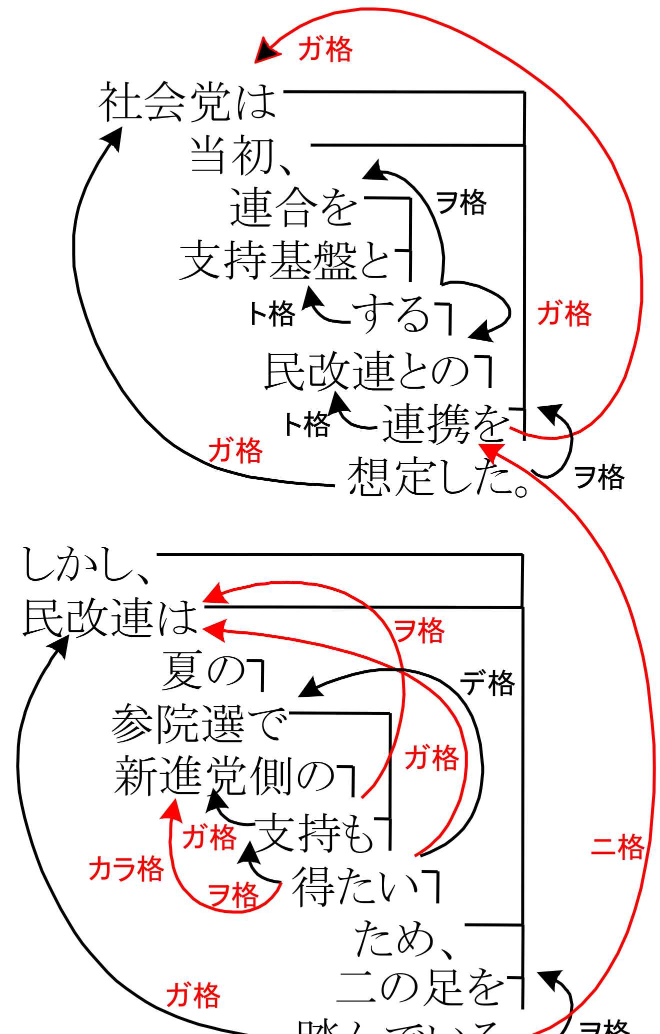
1. Word segmentation
2. POS tagging
3. Phrase chunking
4. Parsing



Kyoto University Text Corpus

[Kurohashi&Nagao 1998]

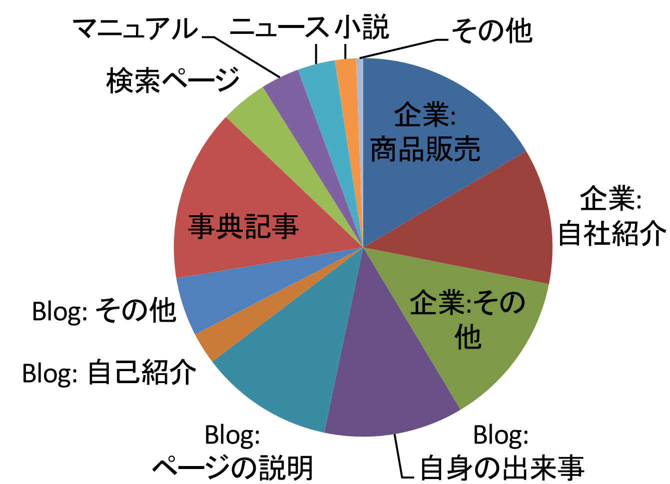
- 40K Mainichi newspaper articles annotated with syntactic information
  - Word segmentation
  - POS
  - Dependency
- 10K articles annotated with relation information
  - Predicate-argument structures
  - Relations between nouns
  - Anaphora and coreference



KU Web Document Leads Corpus

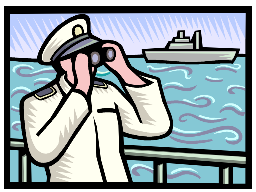
[Hangyo+ 2014]

- Lead 3 sentences of 5K web documents annotated with various linguistic information
  - Annotated by linguists
    - Word segmentation
    - POS
    - Dependency
    - Predicate-argument structures
    - Anaphora and coreference
  - Annotated by crowdworkers
    - Discourse relations



今回は様々な保険について([著者]ガ)([読者]ニ)説明しています。丁寧に([著者]ガ)([読者]ニ)(保険ヲ)解説したつもりですが、逆接([読者]ガ)分からない部分もあるかもしれません。原因・理由 疑問点はどんどん([読者]ガ) ([著者]ニ)コメントしてください。

Dependency Parsing based on Case Frames (KNP)



？

クロールで 泳いでいる女の子を見た
望遠鏡で　泳いでいる女の子を見た

Case frames

{人,子,…}が {クロール,平泳ぎ,…}で {海,大海,…}を**泳ぐ**	{人,者,…}が {双眼鏡,望遠鏡,…}で {姿,人,…}を**見る**

Probabilistic Model (KNP)

弁当は

食べ

弁当は食べて帰った

弁当は

OS

$P(\text{帰った} \mid \text{EOS})$ … $\text{EOS})$

$\times P(\text{弁当は食べ}$ … $\text{た})$

食べる1 が を　弁当	食べる1 が　弁当 を	いずれか
食べる2 が を　弁当 に ⋮	食べる2 が　弁当 を に ⋮	食べる2 が を に　弁当 ⋮

$P(CS(\text{帰る}) \mid \text{タ形}, \text{E}$ … $\text{タ形}, \text{EOS})$

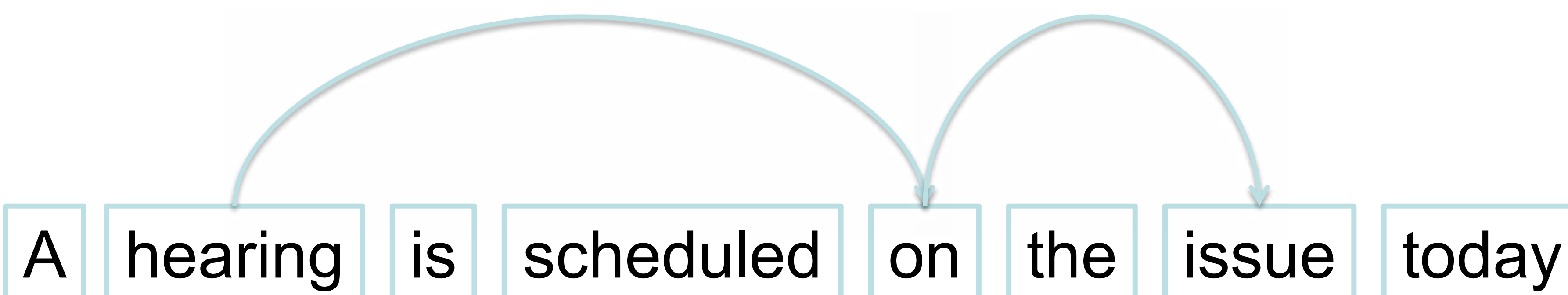
$\times P(\text{タ形} \mid \text{EOS})$ $\quad \times P(\text{タ形} \mid \text{EOS})$

$\times P(CS(\text{弁当は食べる}) \mid \text{テ形}, \text{帰る})$ $\quad \times P(CS(\text{食べる}) \mid \text{テ形}, \text{帰る})$

$\times P(\text{テ形} \mid \text{タ形})$ $\quad \times P(\text{テ形} \mid \text{タ形})$

Pointwise Edge Score Estimation (EDA)

- Trainable from partially annotated sentences
  - Only some words are annotated
  - Practical for domain adaptation



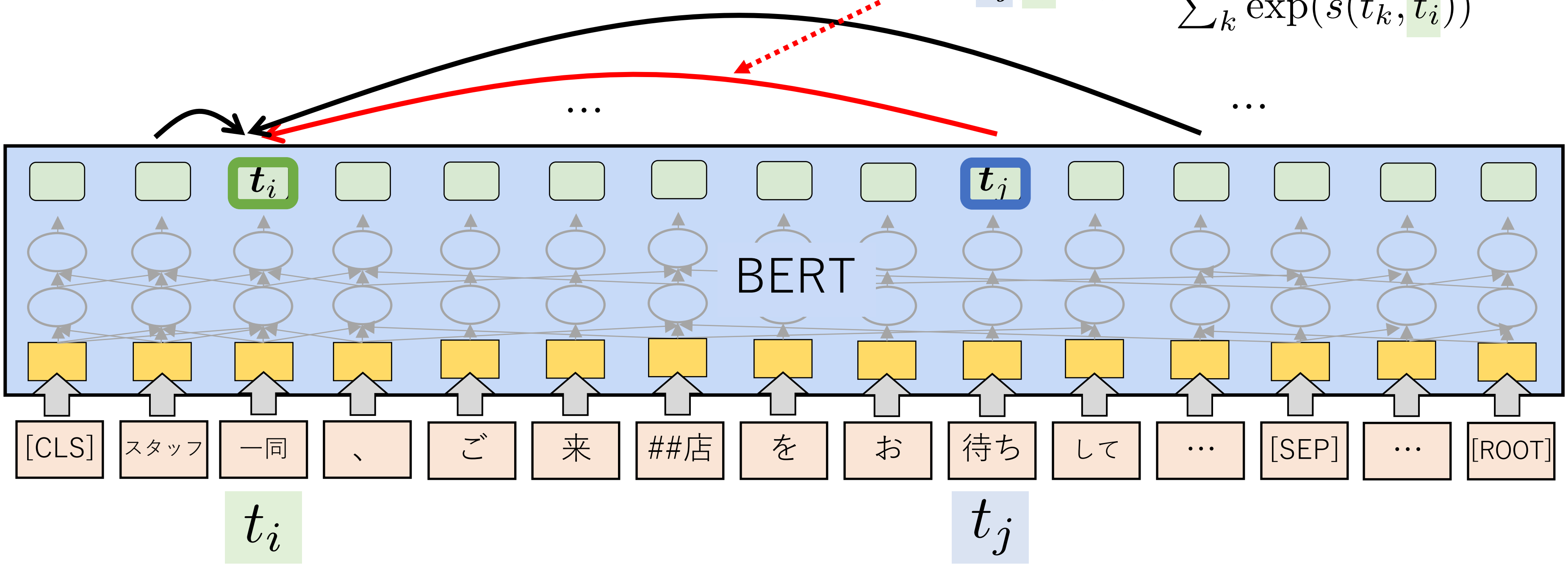
BERT-based Dependency Parsing

[柴田+ 2019]

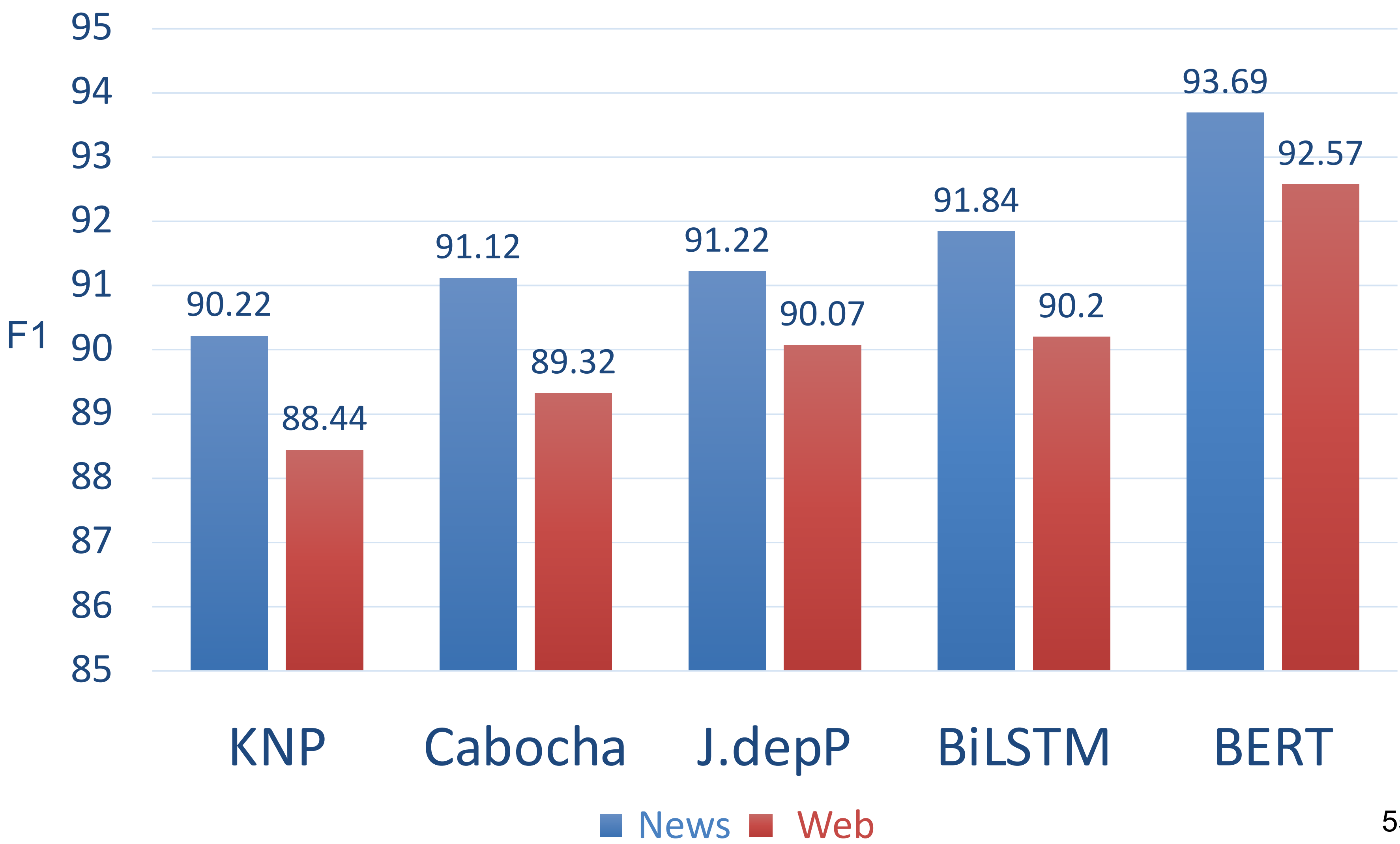
Based on
head selection [Zhang+ 2017]

$$s(t_j, t_i) = \boldsymbol{v}_h^{\mathrm{T}} \tanh(U_h \boxed{\boldsymbol{t}_j} + W_h \boxed{\boldsymbol{t}_i})$$

$$P_{head}(t_j | t_i, S) = \frac{\exp(s(t_j, t_i))}{\sum_k \exp(s(t_k, t_i))}$$



Phrase-based Performance (F1)



Error Analysis

Tree: Gold tree
▼: System output

Coordination

当社nのp
購買n
方針nとp ▼ P
おp取引先nのp
選定nをp
するv
際nのp
基準nをp
公開nしてvいsますs。 *

Others

岩盤n
浴nならp ▼
サウナnがp
苦手なj
方nもp
低温nでもp
多量のj
汗nがp
出vますsのでx
本当にj
驚きvますsよp。 *

Compound nouns

担当n▼
責任n
者sがp、 *
何n回sにpもp
分けてv
実施nしてvいるs。 *

またa、 *
既存nのp
おp客nさまsのp ▼
おp申込みn
受付nもp
２０１１n年s
２n月s
２８n日sにてp
終了nいたしvますs。 *

Topic markers (*wa*)

備えn付けnのp
食器n
洗浄n
器nはp ▼
ほとんどa
使用nされてsおらsずx、 * P
新品n
同様のj
状態nですc。 *

Summary

- Dependency formalism
- Graph-based parsing
- Transition-based parsing
- Japanese dependency parsing