IN2110: Methods in Language Technology

Dependency Parsing

Stephan Oepen

Language Technology Group (LTG)

April 30, 2019

Topics for Today

- Short recap:
  - Phrase Structure vs. Dependency syntax
  - Formal properties of dependency graphs
- Universal Dependencies
- Data-driven dependency parsing
  - Variations on shift–reduce parsing
  - The arc-eager transition system
  - Thorough walk-through example
- Transition oracles and features
- Dependency Parser Evaluation
- Sample exam questions

Recent Advances in Dependency Parsing

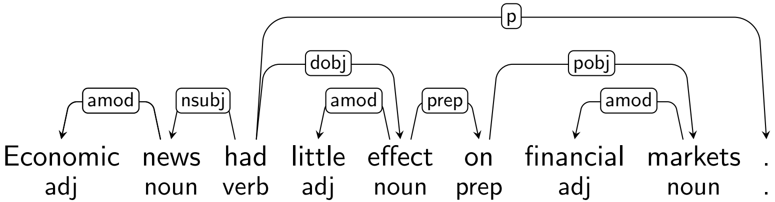
Tutorial, EACL, April 27th, 2014

Ryan McDonald[1] Joakim Nivre[2]

[1]Google Inc., USA/UK
E-mail: ryanmcd@google.com

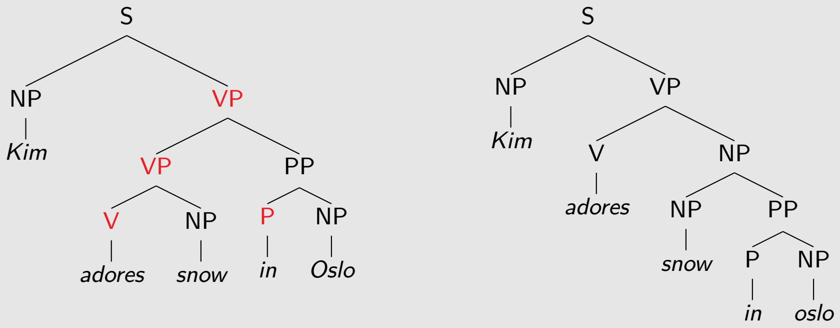
[2]Uppsala University, Sweden
E-mail: joakim.nivre@lingfil.uu.se

Dependency Structure



Terminology

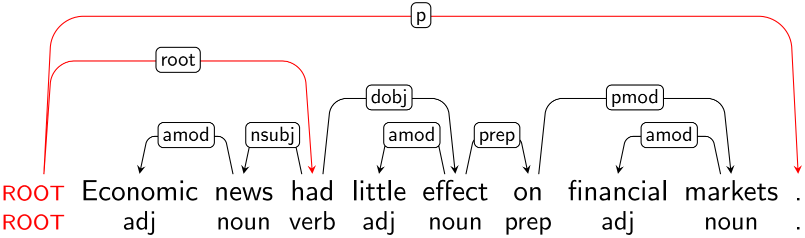
Superior	Inferior
Head	Dependent
Governor	Modifier
Regent	Subordinate
⋮	⋮



(4) Draw the dependency trees for the two readings. Where does the attachment ambiguity manifest itself?

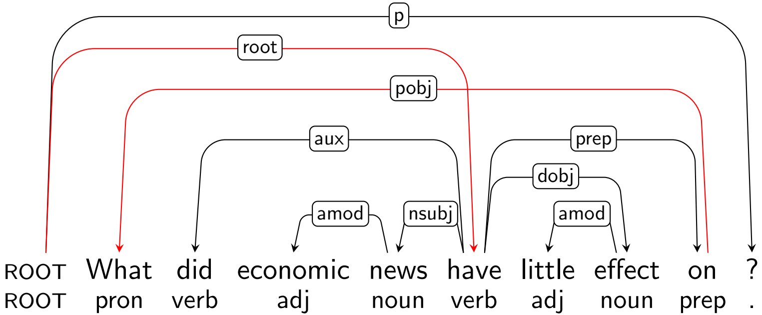
Connectedness, Acyclicity and Single-Head

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.



Projectivity

- Most theoretical frameworks do not assume projectivity.
- Non-projective structures are needed to account for
  - long-distance dependencies,
  - free word order.



Treebanks: Universal Dependencies

universaldependencies.org

This page pertains to UD version 2.

Universal Dependencies

Universal Dependencies (UD) is a framework for cross-linguistically consistent grammatical annotation and an open community effort with over 200 contributors producing more than 100 treebanks in over 70 languages.

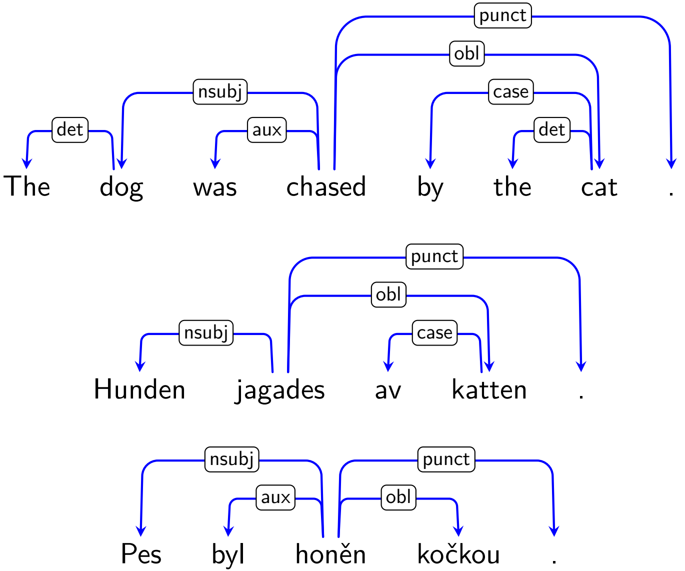
- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

If you want to receive news about Universal Dependencies, you can subscribe to the UD mailing list. If you want to discuss individual annotation

Example 'Universal' Dependency Types

nsubj	nominal subject	She arrived.
csubj	clausal subject	That she arrived surprised me.
obj	(direct) object	My mother called me.
iobj	indirect object	She teaches my daughter maths.
ccomp	clausal complement	She knew that she arrived.
xcomp	open clausal complement	She promised to sing.
obl	oblique nominal	She arrived on Monday
obl	oblique nominal	She depends on me.
nmod	nominal modifier	the office of the chair is empty.
amod	adjectival modifier	the fierce dog barks.
acl	adjectival clause	the dog that barks arrived.
conj	conjunct	Kim and Sandy arrived.
cc	coordinating conjunction	Kim and Sandy arrived.

(Degrees of) Cross-Linguistic Consistency



- Capitalize on content words, e.g. demote case-marking prepositions.

Data-Driven Dependency Parsing

- Need to define a function $f : \mathcal{X} \rightarrow \mathcal{G}$
  - From sentences $x \in \mathcal{X}$ to valid dependency graphs $G \in \mathcal{G}$
- Most common approach is to learn from training data $\mathcal{T}$,
  - where $\mathcal{T} = \{(x_1, G_1), (x_2, G_2), \ldots, (x_n, G_n)\}$,
  - and (x_i, G_i) are labeled sentence and dependency graph pairs that make up the treebank.
- Supervised learning: Fully annotated training examples
- Semi-supervised learning: Annotated data plus constraints and features drawn from unlabeled resources
- Weakly-supervised learning: Constraints drawn from ontologies, structural and lexical resources
- Unsupervised learning: Learning only from unlabeled data

The Basic Idea

- Define a transition system for dependency parsing
- Learn a model for scoring possible transitions
- Parse by searching for the optimal transition sequence

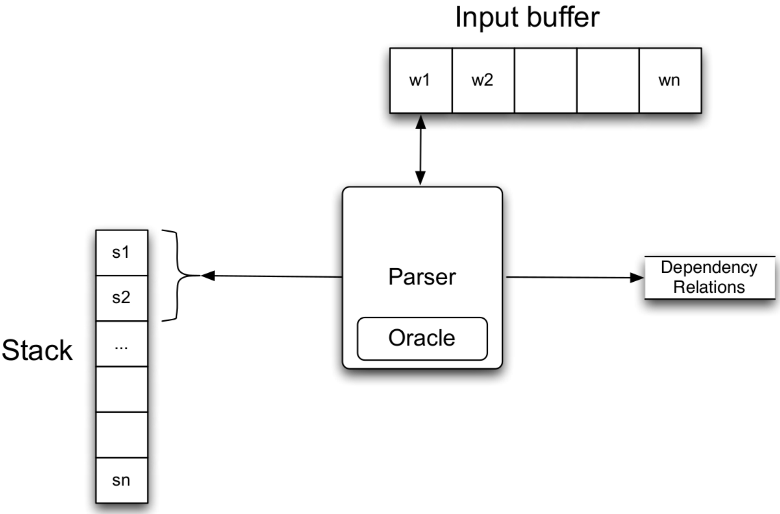
An Adaptation of Shift–Reduce Parsing

- Originally developed for non-ambiguous languages: deterministic.
- Shift ('read') tokens from input buffer, one at a time, left-to-right;
- compare top n symbols on stack against rule RHS: reduce to LHS.
- Dependencies: create arcs between top of stack and front of buffer.

SHIFT	move from front of buffer to top of stack
REDUCE	pop the top of stack (requires existing head)
LEFT-ARC(K)	leftward dependency of type k; reduce
RIGHT-ARC(K)	rightward dependency of type k; shift

- At REDUCE, token must be fully processed (head and dependents).
- LEFT-ARC must respect single-head constraint and unique root node.

Architecture: Stack and Buffer Configurations



Arc-Eager Transition System [Nivre 2003]

Configuration: (S, B, A) [S = Stack, B = Buffer, A = Arcs]

Initial: $([\,], [0, 1, \ldots, n], \{\,\})$

Terminal: $(S, [\,], A)$

Shift:	$(S, i\vert B, A)$	$\Rightarrow$	$(S\vert i, B, A)$	
Reduce:	$(S\vert i, B, A)$	$\Rightarrow$	(S, B, A)	$h(i, A)$
Right-Arc(k):	$(S\vert i, j\vert B, A)$	$\Rightarrow$	$(S\vert i\vert j, B, A \cup \{(i, j, k)\})$	
Left-Arc(k):	$(S\vert i, j\vert B, A)$	$\Rightarrow$	$(S, j\vert B, A \cup \{(j, i, k)\})$	$\neg h(i, A) \wedge i \neq 0$

Notation:
$S|i$ = stack with top i and remainder S
$j|B$ = buffer with head j and remainder B
$h(i, A)$ = i has a head in A

Example Transition Sequence

$[\text{ROOT}]_S$ $[\text{Economic, news, had, little, effect, on, financial, markets, .}]_B$

ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

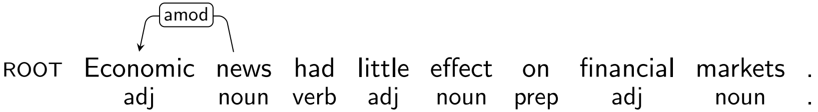
Example Transition Sequence

[ROOT, Economic]$_S$ [news, had, little, effect, on, financial, markets, .]$_B$

ROOT	Economic	news	had	little	effect	on	financial	markets	.
	adj	noun	verb	adj	noun	prep	adj	noun	.

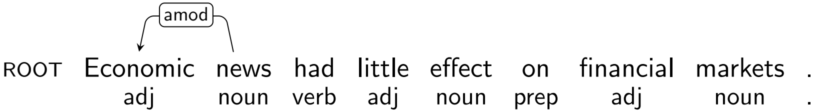
Example Transition Sequence

$[\text{ROOT}]_S$ $[\text{news, had, little, effect, on, financial, markets, .}]_B$



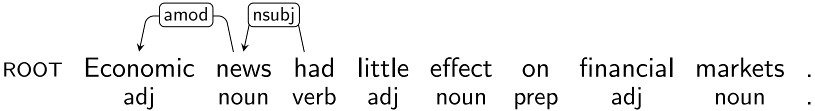
Example Transition Sequence

$[\text{ROOT, news}]_S$ $[\text{had, little, effect, on, financial, markets, .}]_B$



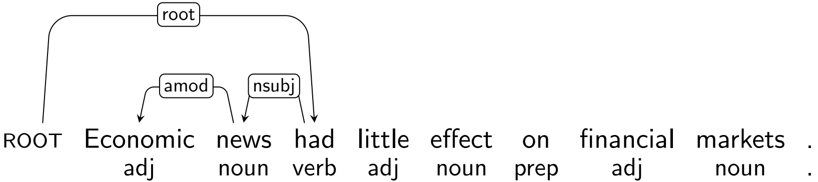
Example Transition Sequence

[ROOT]$_S$ [had, little, effect, on, financial, markets, .]$_B$



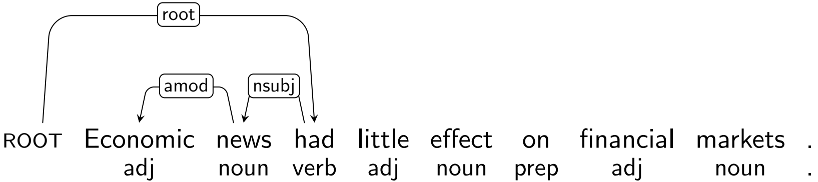
Example Transition Sequence

$[\text{ROOT, had}]_S \ \ [\text{little, effect, on, financial, markets, .}]_B$



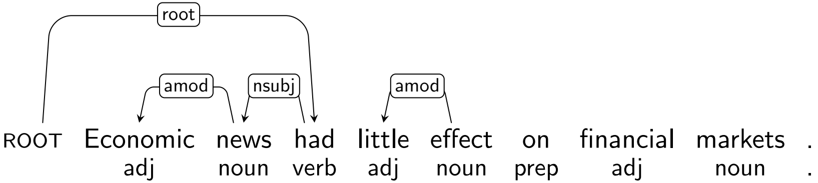
Example Transition Sequence

$[\text{ROOT, had, little}]_S \quad [\text{effect, on, financial, markets, .}]_B$



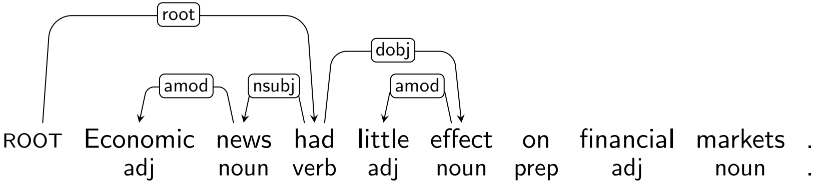
Example Transition Sequence

$[\text{ROOT, had}]_S$ $[\text{effect, on, financial, markets, .}]_B$



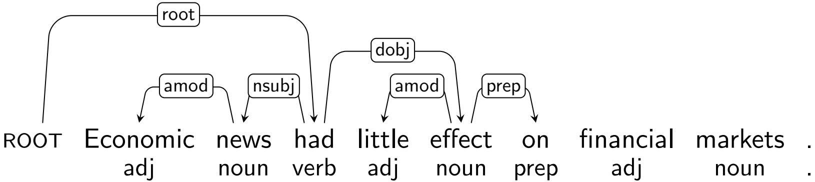
Example Transition Sequence

[ROOT, had, effect]$_S$ [on, financial, markets, .]$_B$



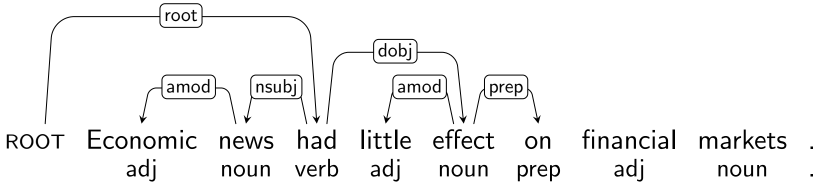
Example Transition Sequence

[ROOT, had, effect, on]$_S$ [financial, markets, .]$_B$



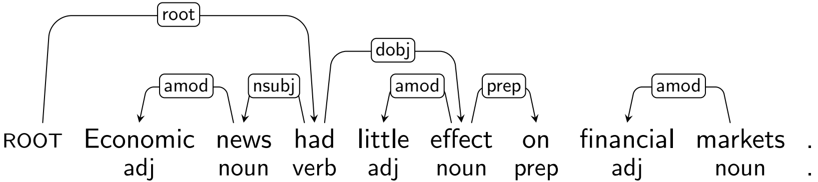
Example Transition Sequence

[ROOT, had, effect, on, financial]$_S$ [markets, .]$_B$



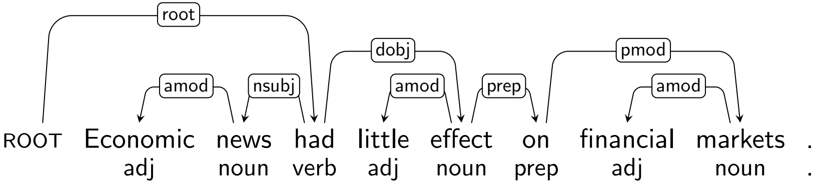
Example Transition Sequence

[ROOT, had, effect, on]$_S$ [markets, .]$_B$



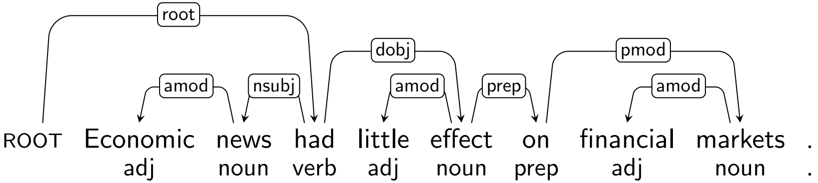
Example Transition Sequence

$[\text{ROOT, had, effect, on, markets}]_S \quad [.]_B$



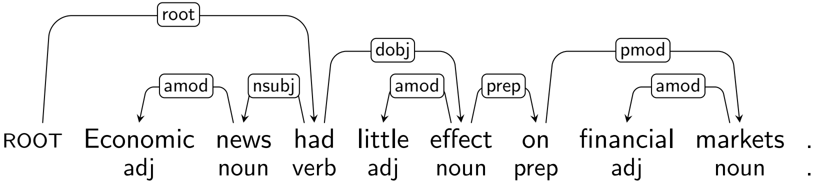
Example Transition Sequence

$[\text{ROOT, had, effect, on}]_S \quad [.]_B$



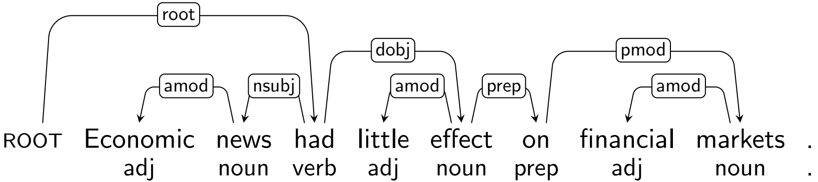
Example Transition Sequence

$[\text{ROOT, had, effect}]_S \quad [.]_B$



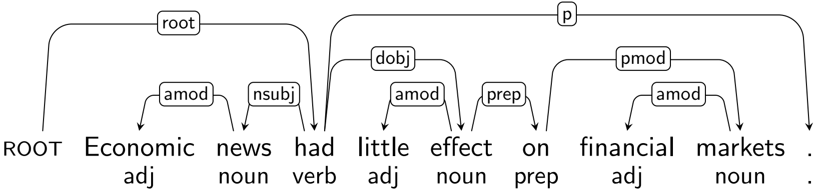
Example Transition Sequence

$[\text{ROOT, had}]_S \quad [.]_B$



Example Transition Sequence

$[\text{ROOT, had, .}]_S \quad [\;]_B$



What Just Happened

SHIFT LEFT-ARC(AMOD)

SHIFT LEFT-ARC(NSUBJ)

RIGHT-ARC(ROOT)

SHIFT LEFT-ARC(AMOD)

RIGHT-ARC(DOBJ)

RIGHT-ARC(PREP)

SHIFT LEFT-ARC(AMOD)

RIGHT-ARC(PMOD)

REDUCE REDUCE REDUCE

RIGHT-ARC(P)

REDUCE REDUCE

Navigating the Parser Search Space

The Search Space

- Transition system ensures formal wellformedness of dependency trees;
- The arc-eager system can generate all projective trees (and only those);
- A specific sequence of transitions determines the final parsing result.
- For a given tree, there can be multiple equivalent transition sequences.

Towards a Parsing Algorithm

- Abstract goal: Find transition sequence that yields the 'correct' tree.
- Learn from treebanks: output dependency tree with high probability.
- Probability distributions over transitions sequences (rather than trees).

Greedy Inference

- Given an oracle o that correctly predicts the next transition $o(c)$, parsing is deterministic:

Parse$(w_1, \ldots, w_n)$

$c \leftarrow ([\ ]_S, [0, 1, \ldots, n]_B, \{\ \})$

while $B_c \neq [\ ]$

$\quad t \leftarrow o(c)$

$\quad c \leftarrow t(c)$

return $G = (\{0, 1, \ldots, n\}, A_c)$

- Complexity given by upper bound on number of transitions
- Parsing in $O(n)$ time for the arc-eager transition system

From Oracles to Classifiers

- An oracle can be approximated by a (linear) classifier:

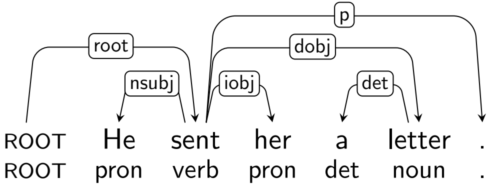
$$o(c) = \operatorname*{argmax}_t \mathbf{w} \cdot \mathbf{f}(c, t)$$

- History-based feature representation $\mathbf{f}(c, t)$
- Weight vector $\mathbf{w}$ learned from treebank data

Oracle Parse

Transitions:

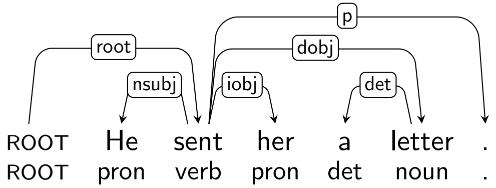
Stack	**Buffer**	**Arcs**
[ ]	[ROOT, He, sent, her, a, letter, .]	



Oracle Parse

Transitions: SH

Stack	**Buffer**	**Arcs**
[ROOT]	[He, sent, her, a, letter, .]	



Oracle Parse

Transitions: SH-RA

Stack

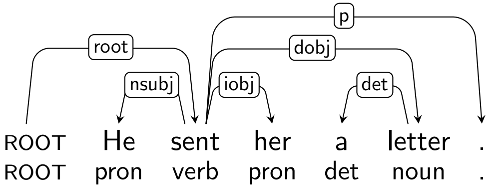
[ROOT, He]

Buffer

[sent, her, a, letter, .]

Arcs

$\text{ROOT} \xrightarrow{\text{root}} \text{sent}$



Oracle Parse

Transitions: SH-RA-LA

Stack

[ROOT]

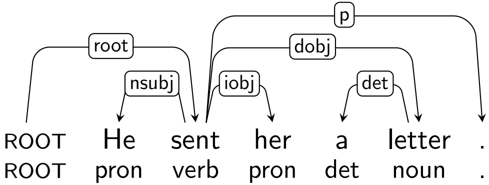
Buffer

[sent, her, a, letter, .]

Arcs

ROOT $\xrightarrow{\text{root}}$ sent

He $\xleftarrow{\text{sbj}}$ sent



Oracle Parse

Transitions: SH-RA-LA-SH

Stack

[ROOT, sent]

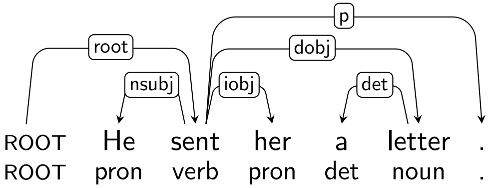
Buffer

[her, a, letter, .]

Arcs

$\text{ROOT} \xrightarrow{\text{root}} \text{sent}$

$\text{He} \xleftarrow{\text{sbj}} \text{sent}$



Oracle Parse

Transitions: SH-RA-LA-SH-RA

Stack

[ROOT, sent, her]

Buffer

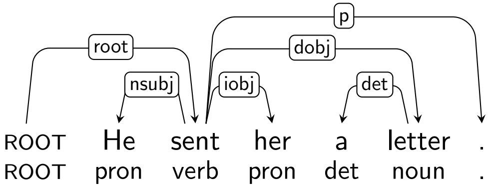
[a, letter, .]

Arcs

ROOT $\xrightarrow{\text{root}}$ sent

He $\xleftarrow{\text{sbj}}$ sent

sent $\xrightarrow{\text{iobj}}$ her



Oracle Parse

Transitions: SH-RA-LA-SH-RA-SH

Stack

[ROOT, sent, her, a]

Buffer

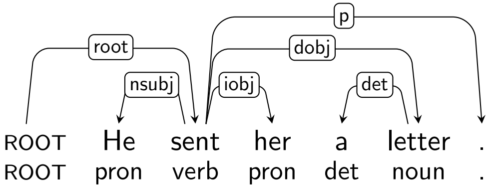
[letter, .]

Arcs

$\text{ROOT} \xrightarrow{\text{root}} \text{sent}$

$\text{He} \xleftarrow{\text{sbj}} \text{sent}$

$\text{sent} \xrightarrow{\text{iobj}} \text{her}$



Oracle Parse

Transitions: SH-RA-LA-SH-RA-SH-LA

Stack

[ROOT, sent, her]

Buffer

[letter, .]

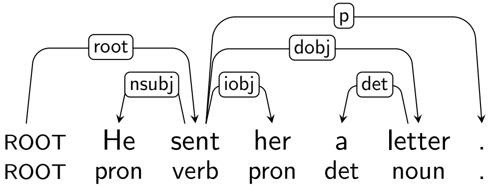
Arcs

ROOT $\xrightarrow{\text{root}}$ sent

He $\xleftarrow{\text{sbj}}$ sent

sent $\xrightarrow{\text{iobj}}$ her

a $\xleftarrow{\text{det}}$ letter



Oracle Parse

Transitions: SH-RA-LA-SH-RA-SH-LA-RE

Stack

[ROOT, sent]

Buffer

[letter, .]

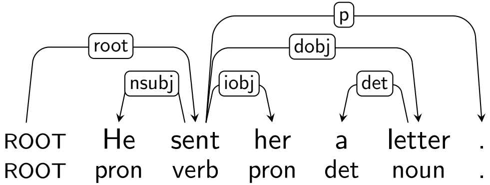
Arcs

ROOT $\xrightarrow{\text{root}}$ sent

He $\xleftarrow{\text{sbj}}$ sent

sent $\xrightarrow{\text{iobj}}$ her

a $\xleftarrow{\text{det}}$ letter



Oracle Parse

Transitions: SH-RA-LA-SH-RA-SH-LA-RE-RA

Stack

[ROOT, sent, letter]

Buffer

[.]

Arcs

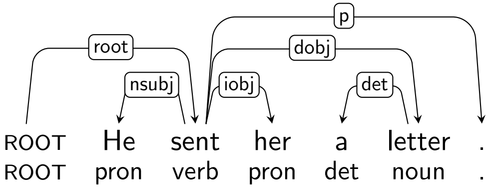
$\text{ROOT} \xrightarrow{\text{root}} \text{sent}$

$\text{He} \xleftarrow{\text{sbj}} \text{sent}$

$\text{sent} \xrightarrow{\text{iobj}} \text{her}$

$\text{a} \xleftarrow{\text{det}} \text{letter}$

$\text{sent} \xrightarrow{\text{dobj}} \text{letter}$



Oracle Parse

Transitions: SH-RA-LA-SH-RA-SH-LA-RE-RA-RE

Stack

[ROOT, sent]

Buffer

[.]

Arcs

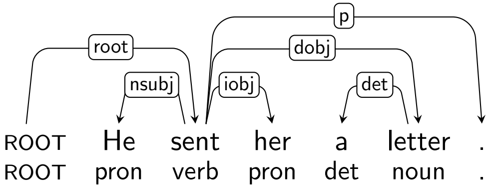
$\text{ROOT} \xrightarrow{\text{root}} \text{sent}$

$\text{He} \xleftarrow{\text{sbj}} \text{sent}$

$\text{sent} \xrightarrow{\text{iobj}} \text{her}$

$\text{a} \xleftarrow{\text{det}} \text{letter}$

$\text{sent} \xrightarrow{\text{dobj}} \text{letter}$



Oracle Parse

Transitions: SH-RA-LA-SH-RA-SH-LA-RE-RA-RE-RA

Stack

[ROOT, sent, .]

Buffer

[ ]

Arcs

ROOT $\xrightarrow{\text{root}}$ sent

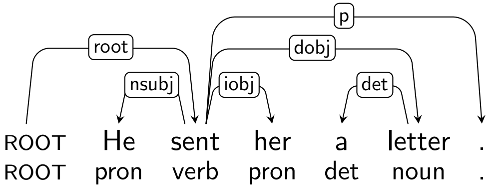
He $\xleftarrow{\text{sbj}}$ sent

sent $\xrightarrow{\text{iobj}}$ her

a $\xleftarrow{\text{det}}$ letter

sent $\xrightarrow{\text{dobj}}$ letter

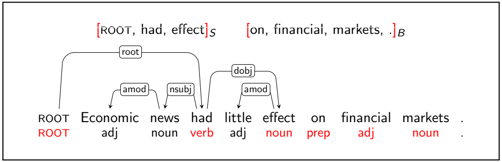
sent $\xrightarrow{\text{p}}$.



Feature Representation

- Features over input tokens relative to S and B

Configuration



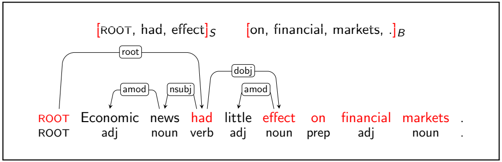
Features

$$\begin{aligned}
pos(S_2) &= \text{ROOT}\\
pos(S_1) &= \text{verb}\\
pos(S_0) &= \text{noun}\\
pos(B_0) &= \text{prep}\\
pos(B_1) &= \text{adj}\\
pos(B_2) &= \text{noun}
\end{aligned}$$

Feature Representation

- Features over input tokens relative to S and B

Configuration



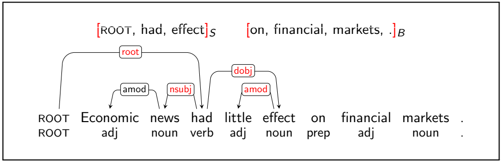
Features

$$
\begin{aligned}
\text{word}(S_2) &= \text{ROOT} \\
\text{word}(S_1) &= \text{had} \\
\text{word}(S_0) &= \text{effect} \\
\text{word}(B_0) &= \text{on} \\
\text{word}(B_1) &= \text{financial} \\
\text{word}(B_2) &= \text{markets}
\end{aligned}
$$

Feature Representation

- Features over input tokens relative to S and B
- Features over the (partial) dependency graph defined by A

Configuration



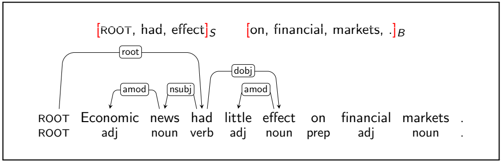
Features

$$\begin{aligned}
\text{dep}(S_1) &= \text{root}\\
\text{dep}(\text{lc}(S_1)) &= \text{nsubj}\\
\text{dep}(\text{rc}(S_1)) &= \text{dobj}\\
\text{dep}(S_0) &= \text{dobj}\\
\text{dep}(\text{lc}(S_0)) &= \text{amod}\\
\text{dep}(\text{rc}(S_0)) &= \text{NIL}
\end{aligned}$$

Feature Representation

- Features over input tokens relative to S and B
- Features over the (partial) dependency graph defined by A
- Features over the (partial) transition sequence

Configuration



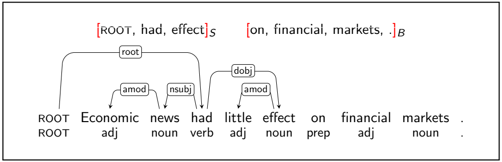
Features

t_{i-1} = Right-Arc(dobj)
t_{i-2} = Left-Arc(amod)
t_{i-3} = Shift
t_{i-4} = Right-Arc(root)
t_{i-5} = Left-Arc(nsubj)
t_{i-6} = Shift

Feature Representation

- Features over input tokens relative to S and B
- Features over the (partial) dependency graph defined by A
- Features over the (partial) transition sequence

Configuration



Features

t_{i-1} = Right-Arc(dobj)
t_{i-2} = Left-Arc(amod)
t_{i-3} = Shift
t_{i-4} = Right-Arc(root)
t_{i-5} = Left-Arc(nsubj)
t_{i-6} = Shift

- Feature representation unconstrained by parsing algorithm

In Conclusion

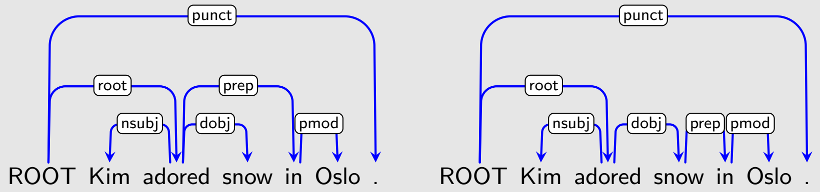
Data-Driven Dependency Parsing

- No notion of grammaticality (no rules): more or less probable trees.
- Much room for experimentation: Feature models and types of classifiers;
- decent results with Maximum Entropy or Support Vector Machines.
- In recent years, further advances with deep neural network classifiers.

Variants on Data-Driven Dependency Parsing

- Other transition systems (e.g. arc-standard; like 'classic' shift-reduce);
- different techniques for non-projective trees; e.g. swap transitions;
- can relax transition system further, to output general, non-tree graphs.
- Beam search: exploring the top-n transitions out of each configuration.

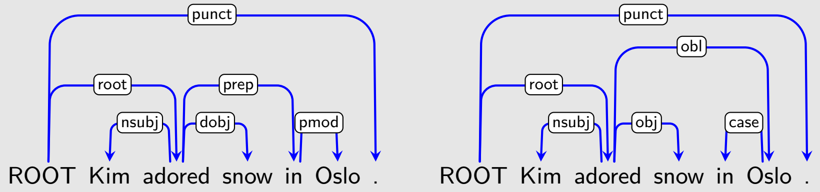
Exercise (5): Dependency Evaluation



(5) What are the LAS and UAS scores for the two trees?

Gold standard on the left, system prediction on the right.

Exercise (6): More Dependency Evaluation



(6) What are the LAS and UAS scores for the two trees?

Next Week: Applications of Syntactic Parsing