

IN2110: Methods in Language Technology

Grammatical Structure Wrap-Up

Stephan Oepen

Language Technology Group (LTG)

May 7, 2019





- ▶ Short recap:
 - ▶ Universal Dependencies
 - ▶ Transition-based dependency parsing



- ▶ Short recap:
 - ▶ Universal Dependencies
 - ▶ Transition-based dependency parsing
- ▶ Dependency Parser Evaluation



- ▶ Short recap:
 - ▶ Universal Dependencies
 - ▶ Transition-based dependency parsing
- ▶ Dependency Parser Evaluation
- ▶ Variants on data-driven dependency parsing
 - ▶ Graph-based dependency parsing
 - ▶ Arc-standard transition system
 - ▶ Semantic dependency graphs



- ▶ Short recap:
 - ▶ Universal Dependencies
 - ▶ Transition-based dependency parsing
- ▶ Dependency Parser Evaluation
- ▶ Variants on data-driven dependency parsing
 - ▶ Graph-based dependency parsing
 - ▶ Arc-standard transition system
 - ▶ Semantic dependency graphs
- ▶ Syntactic structure in negation resolution



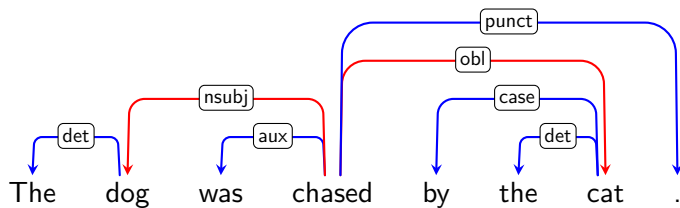
- ▶ Short recap:
 - ▶ Universal Dependencies
 - ▶ Transition-based dependency parsing
- ▶ Dependency Parser Evaluation
- ▶ Variants on data-driven dependency parsing
 - ▶ Graph-based dependency parsing
 - ▶ Arc-standard transition system
 - ▶ Semantic dependency graphs
- ▶ Syntactic structure in negation resolution
- ▶ Sample exam questions

Recap: 'Universal' Dependency Types

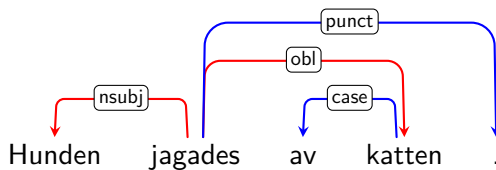
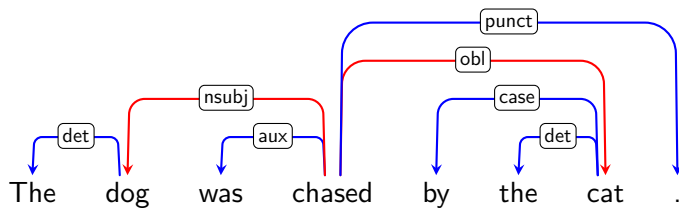


nsubj	nominal subject	<u>She</u> arrived.
csubj	clausal subject	That she arrived <u>surprised</u> me.
obj	(direct) object	My mother <u>called</u> me .
iobj	indirect object	She <u>teaches</u> my daughter maths.
ccomp	clausal complement	She <u>knew</u> that she arrived .
xcomp	open clausal complement	She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

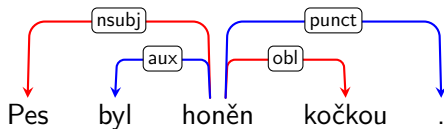
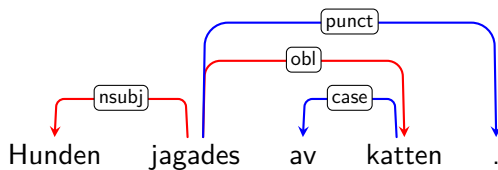
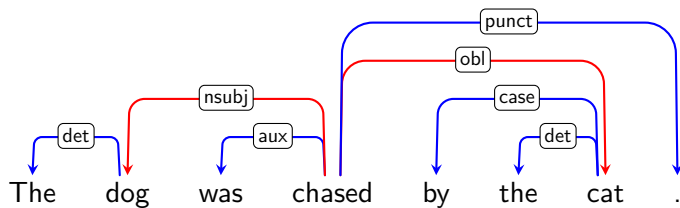
Recap: Cross-Linguistic Consistency



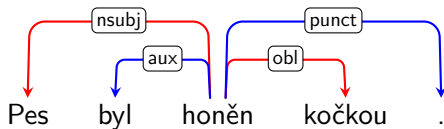
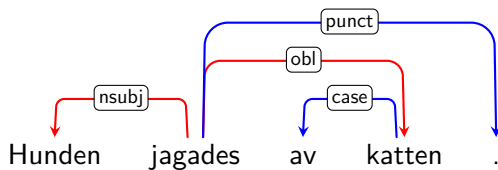
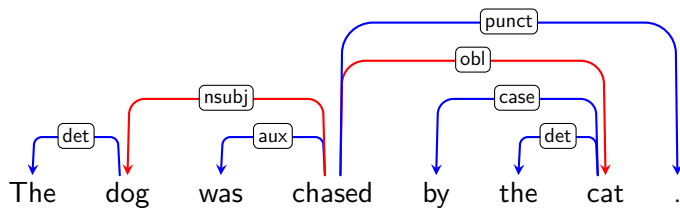
Recap: Cross-Linguistic Consistency



Recap: Cross-Linguistic Consistency



Recap: Cross-Linguistic Consistency

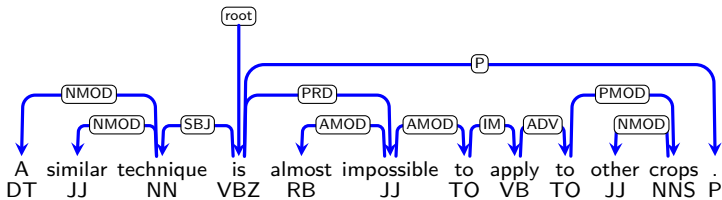


- Capitalize on **content words**, e.g. demote case-marking prepositions.

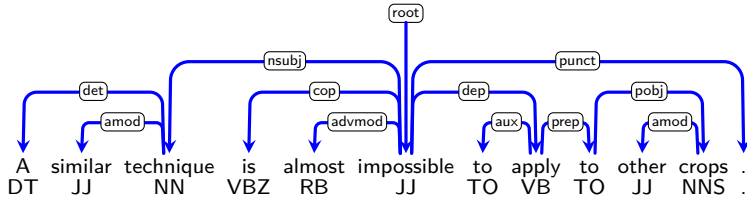
Functional vs. Content Heads



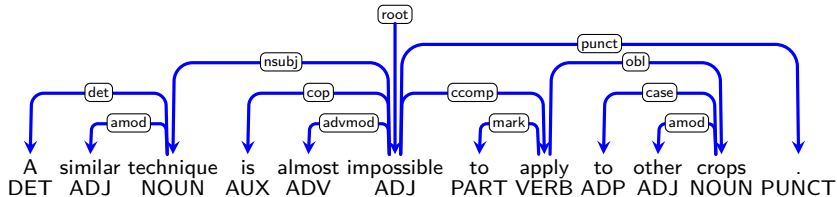
CoNLL



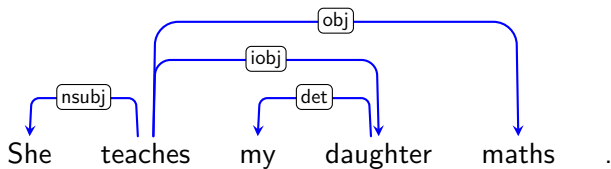
SB



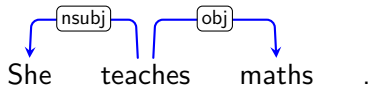
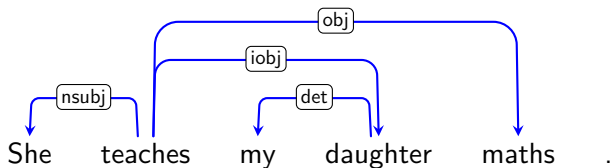
UD



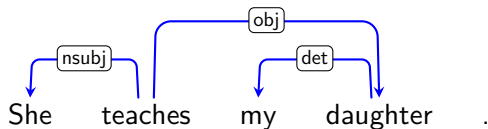
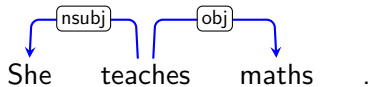
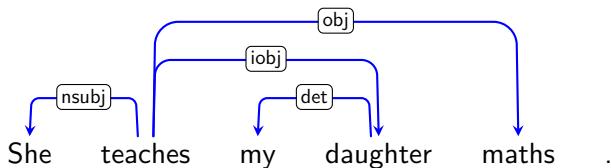
Consistency Can be Evasive



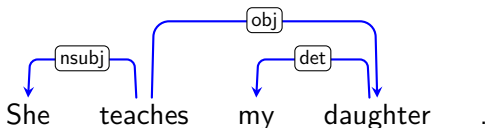
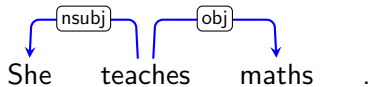
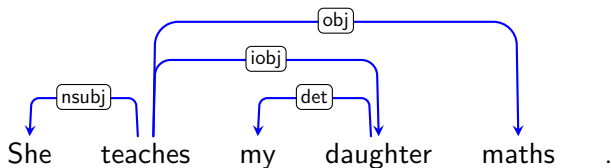
Consistency Can be Evasive



Consistency Can be Evasive



Consistency Can be Evasive



- UD guidelines: [...] if there is just one object, it should be labeled obj.

UD: The Big Picture



Browser tabs: Inbox (6), Google, Community, NLP L Y, Cross-Fram, 2019 Ph, IN Z110, GPU sup, Inspira, Søk ette, Seker | C, CuPy, Timepla, Universa x, obj

Address bar: <https://universaldependencies.org/u/dep/all.html> | 150% | Q, ulio inZ110

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	MWE	Loose	Special	Other
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep



General Ideas

- ▶ Similar to ParsEval, want to award partial credit: **granular** evaluation.
- ▶ Fixed number of tokens: per-token accuracy scores (like in tagging).



General Ideas

- ▶ Similar to ParsEval, want to award partial credit: **granular** evaluation.
- ▶ Fixed number of tokens: per-token accuracy scores (like in tagging).
- ▶ Can consider just **tree topology** or topology plus **dependency types**.



General Ideas

- ▶ Similar to ParsEval, want to award partial credit: **granular** evaluation.
- ▶ Fixed number of tokens: per-token accuracy scores (like in tagging).
- ▶ Can consider just **tree topology** or topology plus **dependency types**.
- ▶ Punctuation tokens (e.g. by Unicode property) are often excluded.



General Ideas

- ▶ Similar to ParsEval, want to award partial credit: **granular** evaluation.
- ▶ Fixed number of tokens: per-token accuracy scores (like in tagging).
- ▶ Can consider just **tree topology** or topology plus **dependency types**.
- ▶ Punctuation tokens (e.g. by Unicode property) are often excluded.

UAS: Unlabeled Attachment Score

- ▶ For each token, does it have correct head (source of incoming edge)?



General Ideas

- ▶ Similar to ParsEval, want to award partial credit: **granular** evaluation.
- ▶ Fixed number of tokens: per-token accuracy scores (like in tagging).
- ▶ Can consider just **tree topology** or topology plus **dependency types**.
- ▶ Punctuation tokens (e.g. by Unicode property) are often excluded.

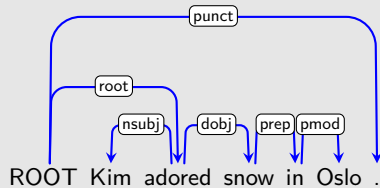
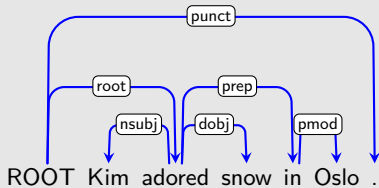
UAS: Unlabeled Attachment Score

- ▶ For each token, does it have correct head (source of incoming edge)?

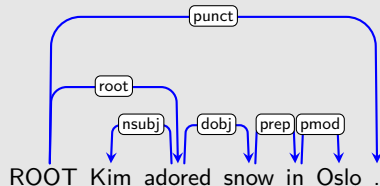
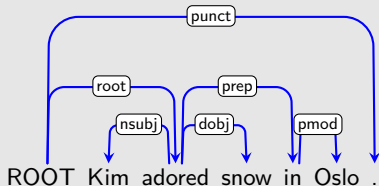
LAS: Labeled Attachment Score

- ▶ In addition to the head, is the dependency type (edge label) correct?

Exercise (5): Dependency Evaluation

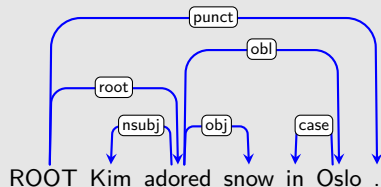
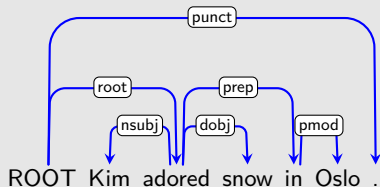


Exercise (5): Dependency Evaluation

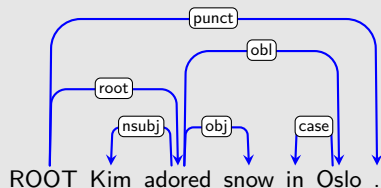
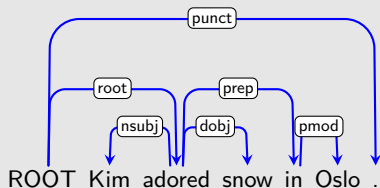


**(5) What are the LAS and UAS scores for the two trees?
Gold standard on the left, system prediction on the right.**

Exercise (6): More Dependency Evaluation



Exercise (6): More Dependency Evaluation



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



Transition-Based Dependency Parsing

- ▶ Transition system ensures **formal wellformedness** of dependency trees;
- ▶ A specific **sequence** of transitions determines the final parsing result.



Transition-Based Dependency Parsing

- ▶ Transition system ensures **formal wellformedness** of dependency trees;
- ▶ A specific **sequence** of transitions determines the final parsing result.
- ▶ Much room for experimentation: Feature models and types of classifiers;
- ▶ decent results with Maximum Entropy or Support Vector Machines.



Transition-Based Dependency Parsing

- ▶ Transition system ensures **formal wellformedness** of dependency trees;
- ▶ A specific **sequence** of transitions determines the final parsing result.
- ▶ Much room for experimentation: Feature models and types of classifiers;
- ▶ decent results with Maximum Entropy or Support Vector Machines.

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**.



Transition-Based Dependency Parsing

- ▶ Transition system ensures **formal wellformedness** of dependency trees;
- ▶ A specific **sequence** of transitions determines the final parsing result.
- ▶ Much room for experimentation: Feature models and types of classifiers;
- ▶ decent results with Maximum Entropy or Support Vector Machines.

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.



Transition-Based Dependency Parsing

- ▶ Transition system ensures **formal wellformedness** of dependency trees;
- ▶ A specific **sequence** of transitions determines the final parsing result.
- ▶ Much room for experimentation: Feature models and types of classifiers;
- ▶ decent results with Maximum Entropy or Support Vector Machines.

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.
- ▶ So-called **graph-based** dependency parsing: somewhat similar to CKY.



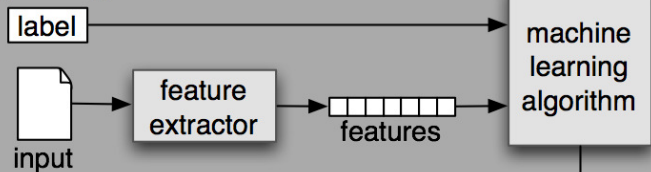
Transition-Based Dependency Parsing

- ▶ Transition system ensures **formal wellformedness** of dependency trees;
- ▶ A specific **sequence** of transitions determines the final parsing result.
- ▶ Much room for experimentation: Feature models and types of classifiers;
- ▶ decent results with Maximum Entropy or Support Vector Machines.

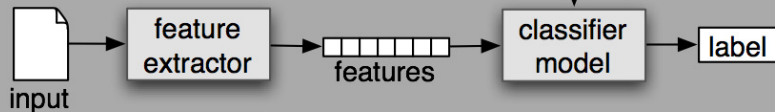
Variants on Data-Driven Dependency Parsing

- ▶ 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.
- ▶ So-called **graph-based** dependency parsing: somewhat similar to CKY.
- ▶ Multi-stratal (multi-layer) representations: MTT, FGD, enhanced UD.

(a) Training



(b) Prediction





- ▶ 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.



- ▶ 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.



- ▶ 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



- ▶ 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 should be fully processed (head and dependents).
- ▶ **LEFT-ARC** must respect single-head constraint and unique root node.

Recent Advances in Dependency Parsing

Tutorial, EACL, April 27th, 2014

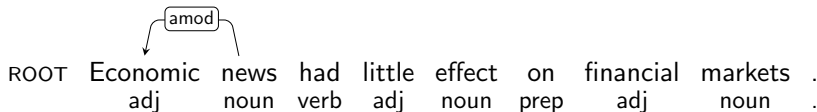
Ryan McDonald¹ Joakim Nivre²

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

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

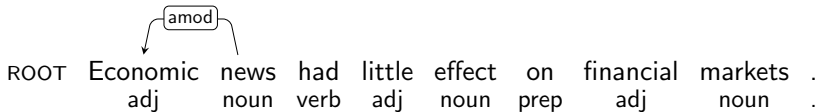
Example Transition Sequence

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



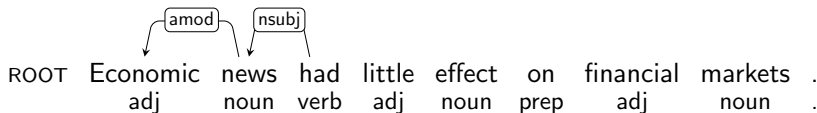
Example Transition Sequence

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



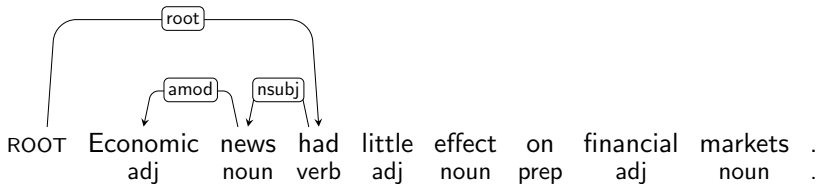
Example Transition Sequence

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



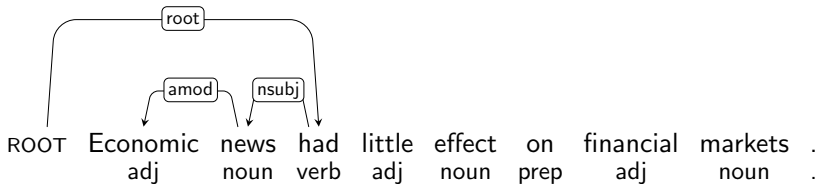
Example Transition Sequence

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



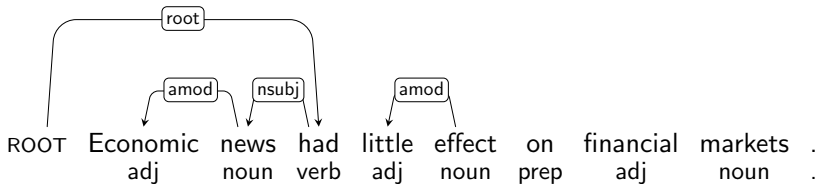
Example Transition Sequence

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



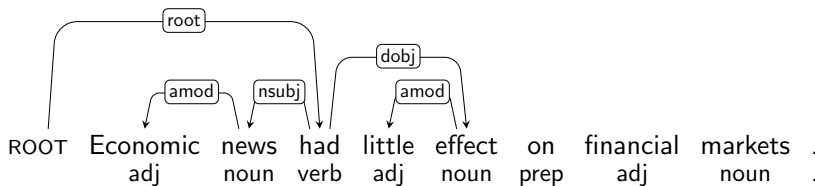
Example Transition Sequence

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



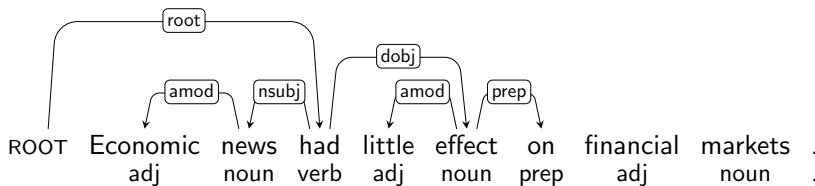
Example Transition Sequence

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



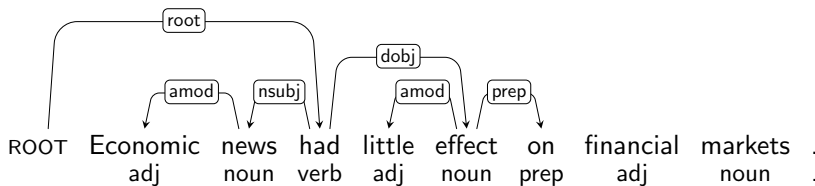
Example Transition Sequence

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



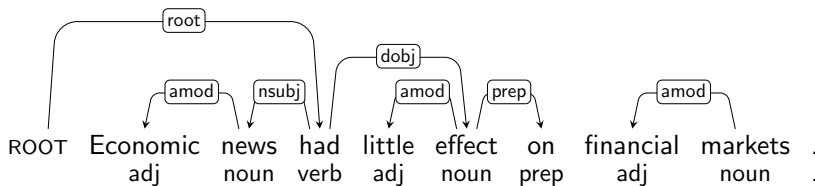
Example Transition Sequence

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



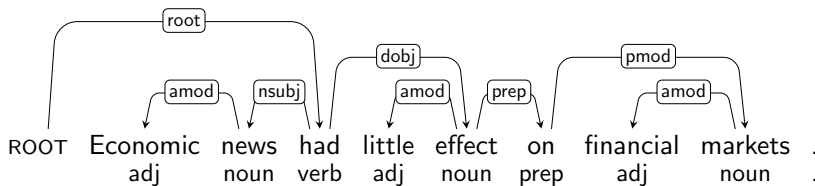
Example Transition Sequence

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



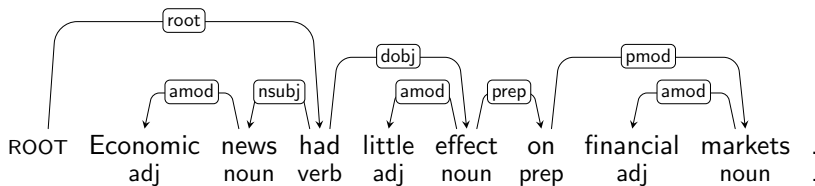
Example Transition Sequence

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



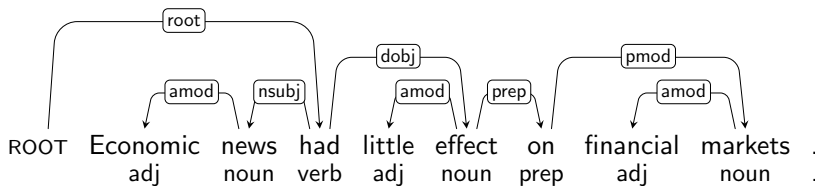
Example Transition Sequence

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



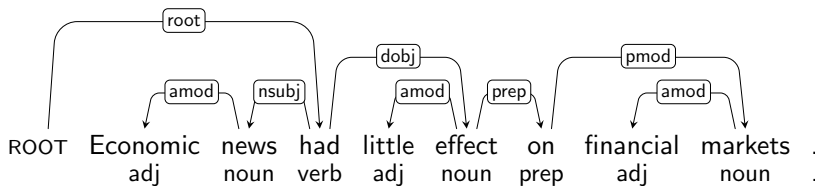
Example Transition Sequence

[ROOT, had, effect]_S [.]_B



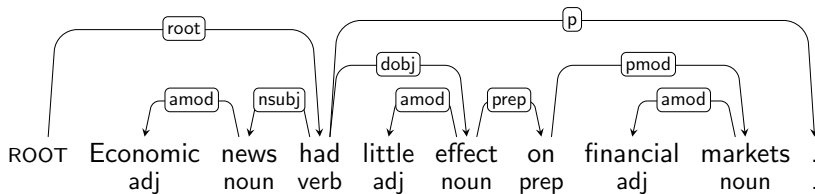
Example Transition Sequence

[ROOT, had]_S [.]_B



Example Transition Sequence

[ROOT, had, .]_S []_B



Arc-Standard Transition System [Nivre 2004]

Configuration: (S, B, A) [$S = \text{Stack}, B = \text{Buffer}, A = \text{Arcs}$]

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

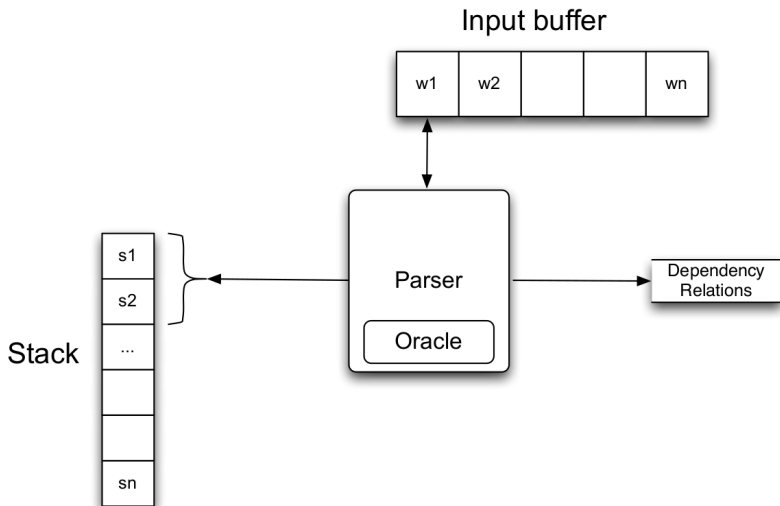
Terminal: $([0], [], A)$

Shift: $(S, i|B, A) \Rightarrow (S|i, B, A)$

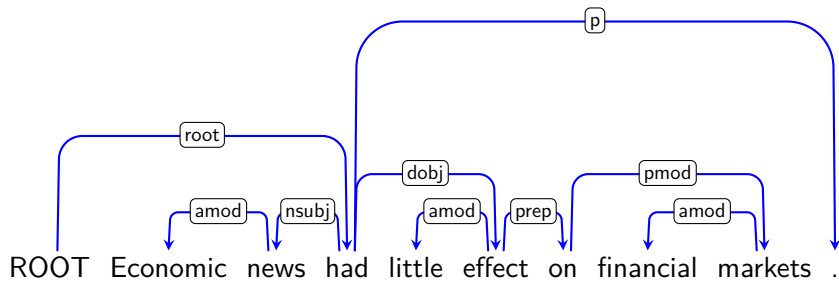
Right-Arc(k): $(S|i|j, B, A) \Rightarrow (S|i, B, A \cup \{(i, j, k)\})$

Left-Arc(k): $(S|i|j, B, A) \Rightarrow (S|j, B, A \cup \{(j, i, k)\}) \quad i \neq 0$

Arc-Standard More Like 'Classic' Shift-Reduce



Using the Arc-Standard Transition System





SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

SHIFT LEFT-ARC(NSUBJ)

SHIFT SHIFT LEFT-ARC(AMOD)

SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

RIGHT-ARC(PMOD)

RIGHT-ARC(PREP)

RIGHT-ARC(DOBJ)

SHIFT RIGHT-ARC(P)

RIGHT-ARC(ROOT)



SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

SHIFT LEFT-ARC(NSUBJ)

SHIFT SHIFT LEFT-ARC(AMOD)

SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

RIGHT-ARC(PMOD)

RIGHT-ARC(PREP)

RIGHT-ARC(DOBJ)

SHIFT RIGHT-ARC(P)

RIGHT-ARC(ROOT)

SHIFT 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



SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

SHIFT LEFT-ARC(NSUBJ)

SHIFT SHIFT LEFT-ARC(AMOD)

SHIFT SHIFT SHIFT

LEFT-ARC(AMOD)

RIGHT-ARC(PMOD)

RIGHT-ARC(PREP)

RIGHT-ARC(DOBJ)

SHIFT RIGHT-ARC(P)

RIGHT-ARC(ROOT)

SHIFT SHIFT LEFT-ARC(AMOD)

SHIFT LEFT-ARC(NSUBJ)

RIGHT-ARC(ROOT)

SHIFT LEFT-ARC(AMOD)

RIGHT-ARC(DOBJ)

RIGHT-ARC(PREP)

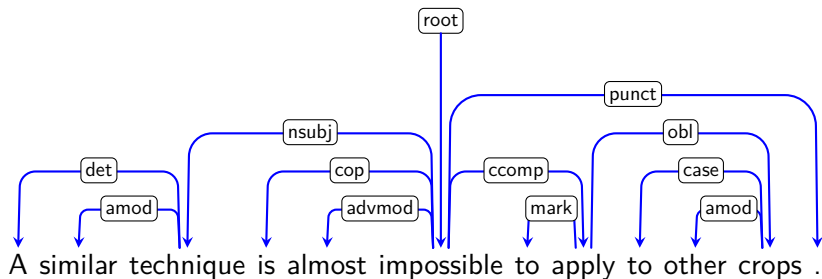
SHIFT LEFT-ARC(AMOD)

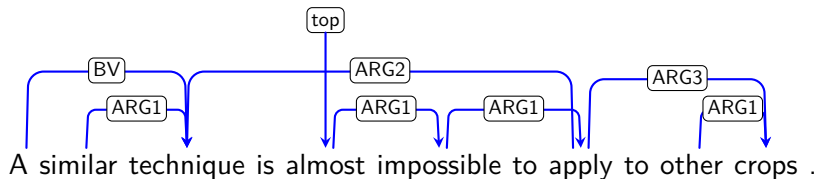
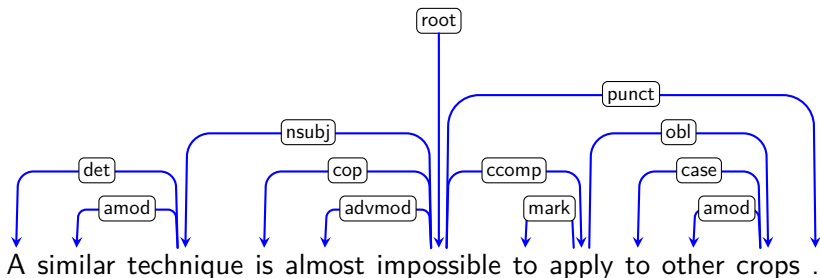
RIGHT-ARC(PMOD)

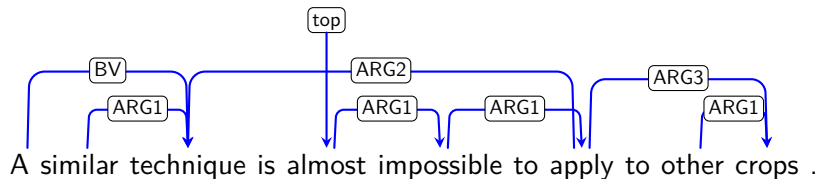
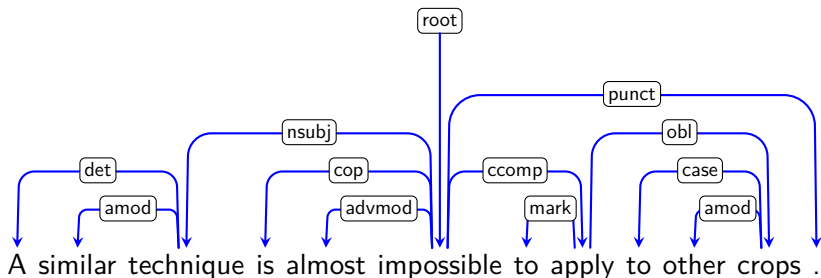
REDUCE REDUCE REDUCE

RIGHT-ARC(P)

REDUCE REDUCE







- ▶ DELPH-IN MRS Dependencies: General graph (beyond rooted trees).
- ▶ **Argument sharing** requires re-entrancy, e.g. control or relative clauses.

Looking Back: How We had Motivated Syntactic Structure

Formal grammars describe a language, providing key notions of:

Wellformedness

- ▶ *Kim was happy because _____ passed the exam.*
- ▶ *Kim was happy because _____ final grade was an A.*
- ▶ *Kim was happy when she saw _____ on television.*

Looking Back: How We had Motivated Syntactic Structure

Formal grammars describe a language, providing key notions of:

Wellformedness

- ▶ *Kim was happy because _____ passed the exam.*
- ▶ *Kim was happy because _____ final grade was an A.*
- ▶ *Kim was happy when she saw _____ on television.*

Meaning

- ▶ *Kim gave Sandy the book.*
- ▶ *Kim gave the book to Sandy.*
- ▶ *Sandy was given the book by Kim.*

Looking Back: How We had Motivated Syntactic Structure

Formal grammars describe a language, providing key notions of:

Wellformedness

- ▶ *Kim was happy because _____ passed the exam.*
- ▶ *Kim was happy because _____ final grade was an A.*
- ▶ *Kim was happy when she saw _____ on television.*

Meaning

- ▶ *Kim gave Sandy the book.*
- ▶ *Kim gave the book to Sandy.*
- ▶ *Sandy was given the book by Kim.*

Ambiguity

- ▶ *Kim ate sushi with chopsticks.*
- ▶ *Have her report on my desk by Friday!*