Chapter NLP:V

V. Syntax

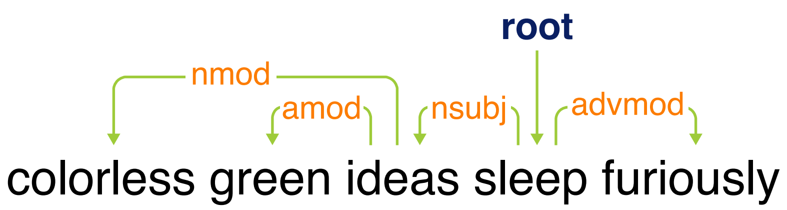
Dependency Grammars

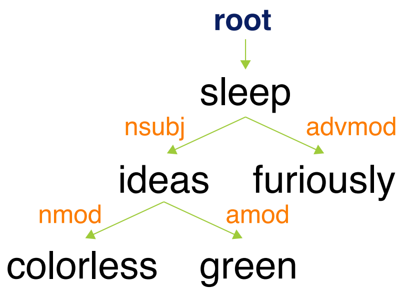
Definition

Dependency grammars describe syntax with a directed head-dependent relationship between words.

- There is exactly one **root** (usually the verb).
- Each word has 1 head and 0–n dependents.
- The head-dependent relation has a grammatical function.
- There is a single path from root to each vertex.

➜ Dependency structures are directed, acyclic, single-headed trees.



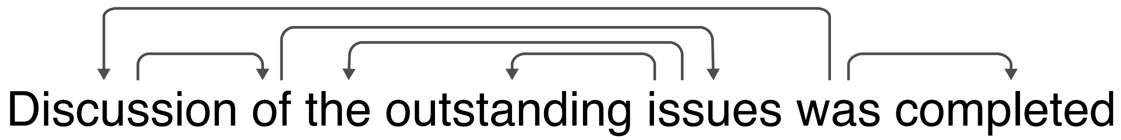


Dependency Grammars

Properties of Dependencies

Text features can be exploited in dependency parsing:

Plausibility	Some dependencies are more plausible than others. “issues $\rightarrow$ the” is more plausible than “the $\rightarrow$ issues”.
Distance	Dependencies more often hold between nearby words. Long-distance dependencies are often problematic. “Ich **muss** um 17 Uhr mit dem Bus nach Hause *fahren*.”.
Breaks	Dependencies rarely span intervening verbs or punctuation.
Valency	Usual numbers of dependents for a head on each side.



Remarks:

- ❑ Dependencies often approximate semantic relationships. Knowing the head-dependent relations of a sentence is very useful for coreference resolution, question answering, and information extraction.
- ❑ Lexicalized CFGs often add the head relation.

Dependency Grammars

Dependency Treebanks: Universal Dependencies [UD, 2021]

The largest treebank for dependencies is Universal Dependencies with "nearly 200 treebanks in over 100 languages".

UD uses the CoNLL-U format to store dependency annotations:

	Lexic		Morphology			Syntax		
ID	**Form**	**Lemma**	**UPOS**	**XPOS**	**Feats**	**Head**	**Deprel**	**Deps**
1	They	they	PRON	PRP	. . .	2	nsubj	2:nsubj|4:nsubj
2	buy	buy	VERB	VBP	. . .	0	root	0:root
3	and	and	CONJ	CC	. . .	4	cc	4:cc
4	sell	sell	VERB	VBP	. . .	2	conj	0:root|2:conj
5	books	book	NOUN	NNS	. . .	2	obj	2:obj|4:obj
6	.	.	PUNCT	.	. . .	2	punct	2:punct

- **Head**: The ID of the head of this item.
- **Deprel**: The dependency relation.
- **Deps**: A head:relation list of the Enhanced Dependencies, which includes advanced concepts but escalates the dependency tree to a graph.

Dependency Grammars

Universal Dependency Relations [de Marneffe et al., 2014]

The UD annotation guidelines use 37 “universal syntactic relations”.

Example selection of depency relations:

Relation	Description	Example with head and dependent
Clausal Arguments		
NSUBJ	Nominal subject	United canceled the flight.
DOBJ	Direct object	We booked her the flight to Miami.
IOBJ	Indirect object	We booked her the flight to Miami.
Nominal Modifier		
NMOD	Nominal modifier	We took the morning flight.
AMOD	Adjectival modifier	Book the cheapest flight.
CASE	Pre- and postpositions, . . .	Book the flight through Houston.
Others		
CONJ	Conjunct	We flew to Denver and drove to Steamboat.
CC	Coordinating conjunction	We flew to Denver and drove to Steamboat.

Dependency Grammars

Universal Dependency Relations [de Marneffe et al., 2014]

The UD annotation guidelines use 37 “universal syntactic relations”.

Example selection of depency relations:

Relation	Description	Example with head and dependent
Clausal Arguments		
NSUBJ	Nominal subject	United canceled the flight.
DOBJ	Direct object	We booked her the flight to Miami.
IOBJ	Indirect object	We booked her the flight to Miami.
Nominal Modifier		
NMOD	Nominal modifier	We took the morning flight.
AMOD	Adjectival modifier	Book the cheapest flight.
CASE	Pre- and postpositions, . . .	Book the flight through Houston.
Others		
CONJ	Conjunct	We flew to Denver and drove to Steamboat.
CC	Coordinating conjunction	We flew to Denver and drove to Steamboat.

Dependency Grammars

Universal Dependency Relations [de Marneffe et al., 2014]

The UD annotation guidelines use 37 “universal syntactic relations”.

Example selection of depency relations:

Relation	Description	Example with head and dependent
Clausal Arguments		
NSUBJ	Nominal subject	United canceled the flight.
DOBJ	Direct object	We booked her the flight to Miami.
IOBJ	Indirect object	We booked her the flight to Miami.
Nominal Modifier		
NMOD	Nominal modifier	We took the morning flight.
AMOD	Adjectival modifier	Book the cheapest flight.
CASE	Pre- and postpositions, . . .	Book the flight through Houston.
Others		
CONJ	Conjunct	We flew to Denver and drove to Steamboat.
CC	Coordinating conjunction	We flew to Denver and drove to Steamboat.

Dependency Grammars

Transition-based parsing [Nivre, 2008]

Dependency trees can be parsed in linear time using an incremental transition system S and an oracle o:

$S = (C, T, c_s, C_t)$

C	Set of configurations $\{(\beta_1, A_1), (\beta_2, A_2), ..\}$ β is a buffer of remaining nodes A is a set of dependency arcs
T	Set of transitions $t : C \rightarrow C$
c_s	Initialization function mapping $w_1, \ldots, w_n$ to (β, A) with $\beta = [1, \ldots, n], A = \emptyset$
C_t	Set of terminal configurations (parses) $C_t \subseteq C$

Dependency Grammars

Transition-based parsing [Nivre, 2008]

Dependency trees can be parsed in linear time using an incremental transition system S and an oracle o:

$S = (C, T, c_s, C_t)$

C	Set of configurations $\{(\beta_1, A_1), (\beta_2, A_2), ..\}$ β is a buffer of remaining nodes A is a set of dependency arcs
T	Set of transitions $t : C \to C$
c_s	Initialization function mapping $w_1, \ldots, w_n$ to (β, A) with $\beta = [1, \ldots, n], A = \emptyset$
C_t	Set of terminal configurations (parses) $C_t \subseteq C$

- The buffer β never increases.
- If β is empty, the parser terminates. a C_t should have been reached
- A never decreases. arcs are only added, never removed

Dependency Grammars

Transition-based parsing [Nivre, 2008]

Dependency trees can be parsed in linear time using an incremental transition system S and an oracle o:

$S = (C, T, c_s, C_t)$

C	Set of configurations $\{(\beta_1, A_1), (\beta_2, A_2), ..\}$ β is a buffer of remaining nodes A is a set of dependency arcs
T	Set of transitions $t : C \rightarrow C$
c_s	Initialization function mapping $w_1, \ldots, w_n$ to (β, A) with $\beta = [1, \ldots, n], A = \emptyset$
C_t	Set of terminal configurations (parses) $C_t \subseteq C$

There is an oracle $o : C \rightarrow T$:

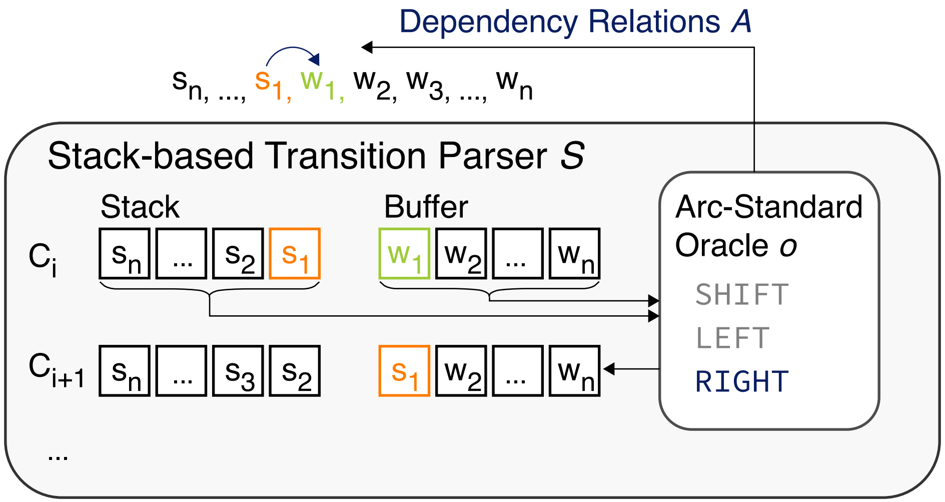
- The oracle determines the next transition given the current configuration. the history of buffers and arcs
- S applies the determined transition, leading to the next configuration.

Dependency Grammars

Arc-Standard Parsing

Arc-Standard is a transition-based parser with a stack σ and 3 transitions T:

SHIFT	Remove the first node from β and push it to σ.
LEFT	Add an arc from the first node in β to the top of σ. Pop σ. Don't LEFT if top of stack is `root` or top of stack has a head
RIGHT	Add an arc from the top of σ to the first node in β. Replace the first node in β with the top of σ. Pop σ. Don't RIGHT if the first node in β has a head

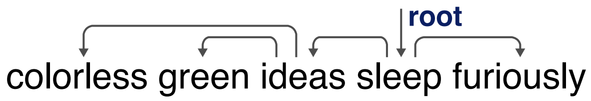


Dependency Grammars

Arc-Standard Parsing

Arc-Standard is a transistion based parser with a stack σ and 3 transitions T:

SHIFT	Remove the first node from β and push it to σ.
LEFT	Add an arc from the first node in β to the top of σ. Pop σ. Don't `LEFT` if top of stack is `root` or top of stack has a head
RIGHT	Add an arc from the top of σ to the first node in β. Replace the first node in β with the top of σ. Pop σ. Don't `RIGHT` if the first node in β has a head



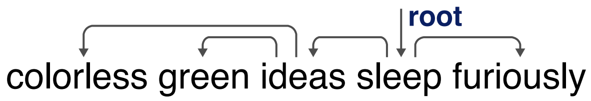
Transition	Stack σ	Buffer β	Relations A
init $\rightarrow$	[root]	[colorless, green, . . . , furiously]	–

Dependency Grammars

Arc-Standard Parsing

Arc-Standard is a transistion based parser with a stack σ and 3 transitions T:

SHIFT	Remove the first node from β and push it to σ.
LEFT	Add an arc from the first node in β to the top of σ. Pop σ. Don't LEFT if top of stack is `root` or top of stack has a head
RIGHT	Add an arc from the top of σ to the first node in β. Replace the first node in β with the top of σ. Pop σ. Don't RIGHT if the first node in β has a head



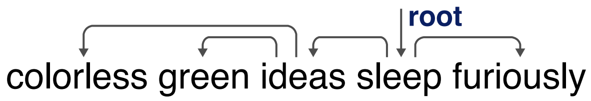
Transition	Stack σ	Buffer β	Relations A
init $\rightarrow$	[root]	[colorless, green, . . . , furiously]	–
SHIFT $\rightarrow$	[root, colorless]	[green, ideas, sleep, furiously]	–

Dependency Grammars

Arc-Standard Parsing

Arc-Standard is a transistion based parser with a stack σ and 3 transitions T:

SHIFT	Remove the first node from β and push it to σ.
LEFT	Add an arc from the first node in β to the top of σ. Pop σ. Don't LEFT if top of stack is root or top of stack has a head
RIGHT	Add an arc from the top of σ to the first node in β. Replace the first node in β with the top of σ. Pop σ. Don't RIGHT if the first node in β has a head



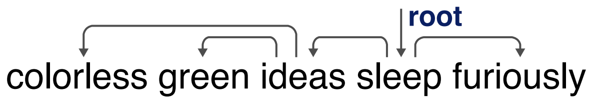
Transition	Stack σ	Buffer β	Relations A
init $\rightarrow$	[root]	[colorless, green, . . . , furiously]	–
SHIFT$\rightarrow$	[root, colorless]	[green, ideas, sleep, furiously]	–
SHIFT $\rightarrow$	[root, colorless, green]	[ideas, sleep, furiously]	–

Dependency Grammars

Arc-Standard Parsing

Arc-Standard is a transistion based parser with a stack σ and 3 transitions T:

SHIFT	Remove the first node from β and push it to σ.
LEFT	Add an arc from the first node in β to the top of σ. Pop σ. Don't LEFT if top of stack is `root` or top of stack has a head
RIGHT	Add an arc from the top of σ to the first node in β. Replace the first node in β with the top of σ. Pop σ. Don't RIGHT if the first node in β has a head



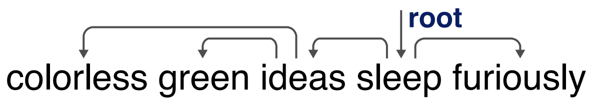
Transition	Stack σ	Buffer β	Relations A
init $\rightarrow$	[root]	[colorless, green, ..., furiously]	–
SHIFT$\rightarrow$	[root, colorless]	[green, ideas, sleep, furiously]	–
SHIFT$\rightarrow$	[root, colorless, green]	[ideas, sleep, furiously]	–
LEFT $\rightarrow$	[root, colorless]	[ideas, sleep, furiously]	$A \cup$ (ideas $\rightarrow$ green)

Dependency Grammars

Arc-Standard Parsing

Arc-Standard is a transistion based parser with a stack σ and 3 transitions T:

SHIFT	Remove the first node from β and push it to σ.
LEFT	Add an arc from the first node in β to the top of σ. Pop σ. Don't LEFT if top of stack is `root` or top of stack has a head
RIGHT	Add an arc from the top of σ to the first node in β. Replace the first node in β with the top of σ. Pop σ. Don't RIGHT if the first node in β already has a head

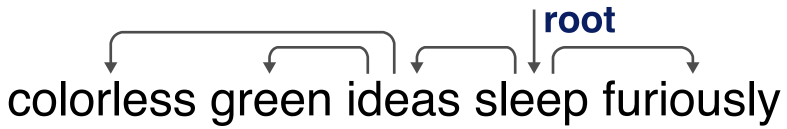


Transition	Stack σ	Buffer β	Relations A
...			
SHIFT $\rightarrow$	[root, sleep]	[furiously]	
RIGHT $\rightarrow$	[root]	[sleep]	$A \cup$ (sleep $\rightarrow$ furiously)

Dependency Grammars

Arc-Standard Parsing

Complete transition sequence until termination. A now contains all relations.



Transition	Stack σ	Buffer β	Relations A
init $\rightarrow$	[root]	[colorless, green, ..., furiously]	–
SHIFT$\rightarrow$	[root, colorless]	[green, ideas, sleep, furiously]	–
SHIFT$\rightarrow$	[root, colorless, green]	[ideas, sleep, furiously]	–
LEFT$\rightarrow$	[root, colorless]	[ideas, sleep, furiously]	$A \cup$ (ideas $\rightarrow$ green)
LEFT$\rightarrow$	[root]	[ideas, sleep, furiously]	$A \cup$ (ideas $\rightarrow$ colorless)
SHIFT$\rightarrow$	[root, ideas]	[sleep, furiously]	–
LEFT$\rightarrow$	[root]	[furiously]	$A \cup$ (sleep $\rightarrow$ ideas)
SHIFT$\rightarrow$	[root, sleep]	[furiously]	
RIGHT$\rightarrow$	[root]	[sleep]	$A \cup$ (sleep $\rightarrow$ furiously)
RIGHT$\rightarrow$	[]	[root]	$A \cup$ (root $\rightarrow$ sleep)
SHIFT$\rightarrow$	[root]	[]	–

Dependency Grammars

Arc-Standard Parsing: Oracles

The oracle $o : C \rightarrow T$ predicts which transition in T = {SHIFT, LEFT, RIGHT} is next.

- Usually classification models, neural or feature based.
- Typical features are based on the stack, buffer, and previous decisions.

  $\rightarrow$ similar to span-based sequence labeling.

Some training examples with class c_i:

$$\begin{aligned} o((\text{Top of } \sigma_{i-1}, \text{POS of } \sigma_{i-1}, \text{Top of } \beta_{i-1}, \text{POS of } \beta_{i-1}, c_{i-1}, c_{i-2})) &= c_i \\ o((\text{green}, \text{JJ}, \text{idea}, \text{NN}, \text{Shift}, \text{Shift})) &= \texttt{LEFT} \\ o((\text{colorless}, \text{JJ}, \text{idea}, \text{NN}, \text{Left}, \text{Shift})) &= \texttt{LEFT} \\ o((\text{root}, \text{root}, \text{idea}, \text{NN}, \text{Left}, \text{Left})) &= ? \end{aligned}$$

i	$o(C_{i-1})$	Configuration C_i		
		Stack σ	Buffer β	Relations A
3	SHIFT $\rightarrow$	[root, colorless, green]	[ideas, sleep, furiously]	–
4	LEFT $\rightarrow$	[root, colorless]	[ideas, sleep, furiously]	$A \cup$ (ideas $\rightarrow$ green)
5	LEFT $\rightarrow$	[root]	[ideas, sleep, furiously]	$A \cup$ (ideas $\rightarrow$ colorless)

Dependency Grammars

Arc-Standard Parsing: Oracles

Traning data can be generated from reference treebank parses:

- Transition through arc-standard as done when parsing.
- Instead of using the oracle, select the transition from the reference parse in this order:

  1. Use LEFT if (First of $\beta \rightarrow$ Top of σ) is in the reference parse.
  2. Else, use RIGHT if
     (a) (Top of $\sigma \rightarrow$ First of β) is in the reference parse and
     (b) all dependents of First of β are assigned.

     otherwise, First of β would vanish befor all dependents were assigned.
  3. Else, use SHIFT.

The arc-standard parse table can be reproduced from its reference parse in this way. The features to train the oracle can then be derived from the parse table.

Dependency Grammars

Remarks:

- There are several extensions to arc-standard, changing the transision rules. *Arc-eager*, for example, adds a REDUCE operator.
- Since the greedy transision system forces a decision and can't revise them, there are frequent errors with, for example, long-distance dependencies. A beam search can mitigate this.
- Predicting the dependency relations is done by extending the transitions to $T = \{\text{SHIFT}, \text{RIGHT}_{nsubj}, \text{LEFT}_{nsubj}, \text{RIGHT}_{dobj}, \ldots\}$

Dependency Grammars

Projectivity [McDonald et al., 2005]

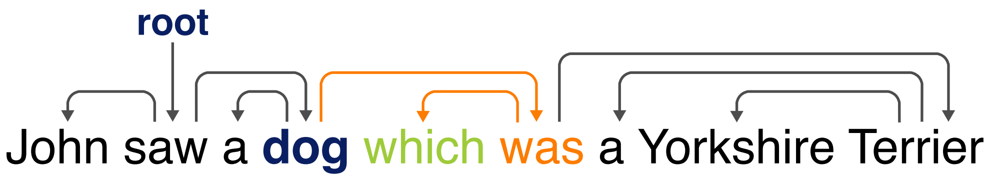
Definition 1 (Projectivity)

A dependency relation (arc) is projective if there is a path from the **head** of the relation to every word between head and dependent.

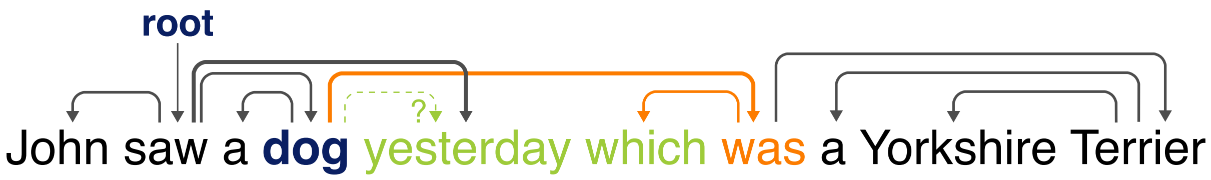
A dependency tree is projective if every arc in it is projective.

- ❑ Common in languages with free word (and attachment) order.
- ❑ Standard transistion-based parsers can not parse non-projective trees.
- ❑ Trees are projective when generated from CFG's. via head-finding rules
- ❑ In non-projective trees, the arcs overlap.

Projective



Non-projective



Dependency Grammars

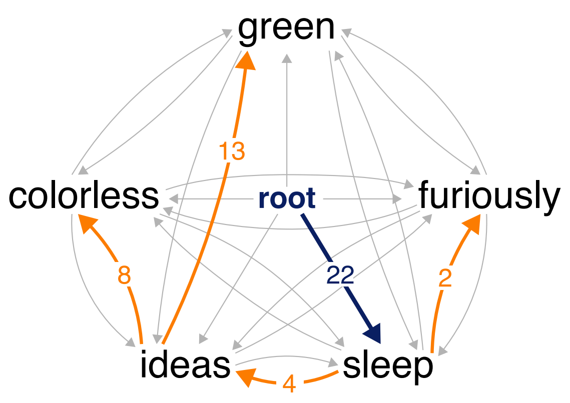
Graph-based Parsing

Idea: Use graph-alogirthms to find the best dependency tree in a fully-connected, directed, weighted graph.

- ❑ More accurate on long-distance dependencies.
- ❑ Can solve projective sentences.

Two problems to solve:

1. How to assign scores to each edge?
   $\rightarrow$ Machine Learning
2. How to find the best parse?
   $\rightarrow$ Maximum Spanning Tree



Graph construction:

- ❑ Create vertices for each word.
- ❑ Create a directed connection from each vertex to all other vertices.
- ❑ Create a root vertex.
- ❑ Create a directed connection from the root to all other vertices.

Dependency Grammars

Evaluation

Dependency parsing is evaluated with the Unlabeled Attachment Score (UAS) and the Labeled Attachment Score (LAS). Both are similar to accuracy.

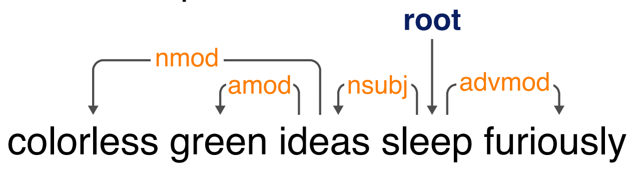
Unlabeled Attachment Score:

- Fraction of correctly attached heads.
- Independent of the assigned label.
- Example: $4/5 = 0.8$.
  green has the wrong head.

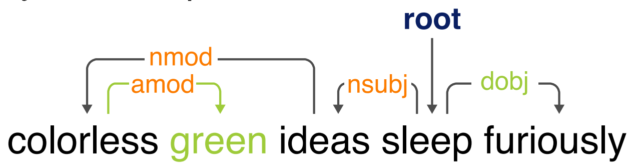
Labeled Attachment Score:

- Fraction of correctly attached heads and labels.
- Example: $3/5 = 0.6$.
  green has the wrong head.
  (sleep $\overrightarrow{dobj}$ furiously) has a wrong label.

Reference parse:



System output:



Dependency Grammars

Evaluation: Comparison of Methods

- All on the same setting: Stanford Dependency conversion of the Penn Treebank.

Approach	Source	UAS	LAS
Large Language Models	[Mrini et al., 2019]	97.4	96.3
Transition (beam search, dense features)	[Weiss, 2015]	94.0	92.0
Transition (arc-hybrid, LSTM features)	[Kiperwasser and Goldberg, 2016]	93.9	91.9
Transition (arc-hybrid, LSTM features)	[Dallesteros, 2016]	93.8	91.5
Graph (LSTM features)	[Kiperwasser and Goldberg, 2016]	93.0	90.9
Transition (arc-eager, beam search)	[Zhang and Nivre, 2011]	92.9	

- Note that the progress since 2011 is marginal.