Chapter NLP:V

V. Syntax

- Introduction
- Context-Free Grammar
- Dependency Grammars
- Features and Unification
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 → the” is more plausible than “the → 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.

Discussion of the outstanding issues was completed
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:

<table>
<thead>
<tr>
<th>ID</th>
<th>Form</th>
<th>Lemma</th>
<th>UPOS</th>
<th>XPOS</th>
<th>Feats</th>
<th>Head</th>
<th>Deprel</th>
<th>Deps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>They</td>
<td>they</td>
<td>PRON</td>
<td>PRP</td>
<td>...</td>
<td>2</td>
<td>nsubj</td>
<td>2:nsubj</td>
</tr>
<tr>
<td>2</td>
<td>buy</td>
<td>buy</td>
<td>VERB</td>
<td>VBP</td>
<td>...</td>
<td>0</td>
<td>root</td>
<td>0:root</td>
</tr>
<tr>
<td>3</td>
<td>and</td>
<td>and</td>
<td>CONJ</td>
<td>CC</td>
<td>...</td>
<td>4</td>
<td>cc</td>
<td>4:cc</td>
</tr>
<tr>
<td>4</td>
<td>sell</td>
<td>sell</td>
<td>VERB</td>
<td>VBP</td>
<td>...</td>
<td>2</td>
<td>conj</td>
<td>0:root</td>
</tr>
<tr>
<td>5</td>
<td>books</td>
<td>book</td>
<td>NOUN</td>
<td>NNS</td>
<td>...</td>
<td>2</td>
<td>obj</td>
<td>2:obj</td>
</tr>
<tr>
<td>6</td>
<td>.</td>
<td>.</td>
<td>PUNCT</td>
<td>.</td>
<td>...</td>
<td>2</td>
<td>punct</td>
<td>2:punct</td>
</tr>
</tbody>
</table>

- **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 \cite{deMarneffe2014}

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

Example selection of dependency relations:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
<th>Example with head and dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clausal Arguments</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSUBJ</td>
<td>Nominal subject</td>
<td>United canceled the flight.</td>
</tr>
<tr>
<td>DOBJ</td>
<td>Direct object</td>
<td>We booked her the flight to Miami.</td>
</tr>
<tr>
<td>IOBJ</td>
<td>Indirect object</td>
<td>We booked her the flight to Miami.</td>
</tr>
<tr>
<td><strong>Nominal Modifier</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMOD</td>
<td>Nominal modifier</td>
<td>We took the morning flight.</td>
</tr>
<tr>
<td>AMOD</td>
<td>Adjectival modifier</td>
<td>Book the cheapest flight.</td>
</tr>
<tr>
<td>CASE</td>
<td>Pre- and postpositions, ...</td>
<td>Book the flight through Houston.</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONJ</td>
<td>Conjunct</td>
<td>We flew to Denver and drove to Steamboat.</td>
</tr>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
<td>We flew to Denver and drove to Steamboat.</td>
</tr>
</tbody>
</table>
Dependency Grammars
Universal Dependency Relations [de Marneffe et al., 2014]

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

Example selection of dependency relations:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
<th>Example with head and dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clausal Arguments</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSUBJ</td>
<td>Nominal subject</td>
<td>United canceled the flight.</td>
</tr>
<tr>
<td>DOBJ</td>
<td>Direct object</td>
<td>We booked her the flight to Miami.</td>
</tr>
<tr>
<td>IOBJ</td>
<td>Indirect object</td>
<td>We booked her the flight to Miami.</td>
</tr>
<tr>
<td><strong>Nominal Modifier</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMOD</td>
<td>Nominal modifier</td>
<td>We took the <strong>morning</strong> flight.</td>
</tr>
<tr>
<td>AMOD</td>
<td>Adjectival modifier</td>
<td>Book the <strong>cheapest</strong> flight.</td>
</tr>
<tr>
<td>CASE</td>
<td>Pre- and postpositions, . . .</td>
<td>Book the flight <strong>through Houston</strong>.</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONJ</td>
<td>Conjunct</td>
<td>We flew to Denver and drove to Steamboat.</td>
</tr>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
<td>We flew to Denver and drove to Steamboat.</td>
</tr>
</tbody>
</table>
Dependency Grammars
Universal Dependency Relations [de Marneffe et al., 2014]

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

Example selection of dependency relations:

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

NLP:V-74 Syntax © WIEGMANN/WOLSKA/WACHSMUTH/HAGEN/POTTHAST/STEIN 2023
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 $\{ (\beta_1, A_1), (\beta_2, A_2), .. \}$
  - $\beta$ 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 $(\beta, A)$ 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 $\{(\beta_1, A_1), (\beta_2, A_2), ..\}$
  - $\beta$ 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 $(\beta, A)$ with $\beta = [1, \ldots, n], A = \emptyset$

- $C_t$: Set of terminal configurations (parses) $C_t \subseteq C$

- The buffer $\beta$ never increases.
- If $\beta$ 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 \cite{Nivre:2008a}

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 $\{ (\beta_1, A_1), (\beta_2, A_2), \ldots \}$
  - $\beta$ 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 $(\beta, A)$ 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 $\sigma$ and 3 transitions $T$:

- **SHIFT**: Remove the first node from $\beta$ and push it to $\sigma$.
- **LEFT**: Add an arc from the first node in $\beta$ to the top of $\sigma$.
  - Pop $\sigma$. Don’t **LEFT** if top of stack is $\text{root}$ or top of stack has a head.
- **RIGHT**: Add an arc from the top of $\sigma$ to the first node in $\beta$.
  - Replace the first node in $\beta$ with the top of $\sigma$.
  - Pop $\sigma$. Don’t **RIGHT** if the first node in $\beta$ has a head.

Stack-based Transition Parser $S$

Buffer

$s_1$, $s_2$, $s_3$, ..., $w_1$, $w_2$, $w_3$, ..., $w_n$

Dependency Relations $A$

$s_n$, ..., $s_1$, $w_1$, $w_2$, $w_3$, ..., $w_n$
Dependency Grammars

Arc-Standard Parsing

Arc-Standard is a transition based parser with a stack $\sigma$ and 3 transitions $T$:  

**SHIFT**  Remove the first node from $\beta$ and push it to $\sigma$.

**LEFT**  Add an arc from the first node in $\beta$ to the top of $\sigma$.

Pop $\sigma$. Don’t **LEFT** if top of stack is $\text{root}$ or top of stack has a head.

**RIGHT**  Add an arc from the top of $\sigma$ to the first node in $\beta$.

Replace the first node in $\beta$ with the top of $\sigma$.

Pop $\sigma$. Don’t **RIGHT** if the first node in $\beta$ has a head.

---

<table>
<thead>
<tr>
<th>Transition</th>
<th>Stack $\sigma$</th>
<th>Buffer $\beta$</th>
<th>Relations $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong></td>
<td>$[\text{root}]$</td>
<td>$[\text{colorless, green, . . ., furiously}]$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
Dependency Grammars
Arc-Standard Parsing

Arc-Standard is a transition based parser with a stack $\sigma$ and 3 transitions $T$:

**SHIFT**  Remove the first node from $\beta$ and push it to $\sigma$.

**LEFT**   Add an arc from the first node in $\beta$ to the top of $\sigma$.
            Pop $\sigma$. Don’t **LEFT** if top of stack is root or top of stack has a head.

**RIGHT**  Add an arc from the top of $\sigma$ to the first node in $\beta$.
            Replace the first node in $\beta$ with the top of $\sigma$.
            Pop $\sigma$. Don’t **RIGHT** if the first node in $\beta$ has a head.

```
colorless green ideas sleep furiously
```

<table>
<thead>
<tr>
<th>Transition</th>
<th>Stack $\sigma$</th>
<th>Buffer $\beta$</th>
<th>Relations $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>init $\rightarrow$</td>
<td>[root]</td>
<td>[colorless, green, ... , furiously]</td>
<td>–</td>
</tr>
<tr>
<td><strong>SHIFT</strong> $\rightarrow$</td>
<td>[root, colorless]</td>
<td>[green, ideas, sleep, furiously]</td>
<td>–</td>
</tr>
</tbody>
</table>
Dependency Grammars
Arc-Standard Parsing

Arc-Standard is a transition based parser with a stack $\sigma$ and 3 transitions $T$:

- **SHIFT** Remove the first node from $\beta$ and push it to $\sigma$.
- **LEFT** Add an arc from the first node in $\beta$ to the top of $\sigma$.
  Pop $\sigma$. Don’t LEFT if top of stack is $\text{root}$ or top of stack has a head.
- **RIGHT** Add an arc from the top of $\sigma$ to the first node in $\beta$.
  Replace the first node in $\beta$ with the top of $\sigma$.
  Pop $\sigma$. Don’t RIGHT if the first node in $\beta$ has a head.

---

Transition | Stack $\sigma$ | Buffer $\beta$ | 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 transition based parser with a stack $\sigma$ and 3 transitions $T$:

- **SHIFT**: Remove the first node from $\beta$ and push it to $\sigma$.
- **LEFT**: Add an arc from the first node in $\beta$ to the top of $\sigma$.
  - Pop $\sigma$. Don’t **LEFT** if top of stack is $\text{root}$ or top of stack has a head.
- **RIGHT**: Add an arc from the top of $\sigma$ to the first node in $\beta$.
  - Replace the first node in $\beta$ with the top of $\sigma$.
  - Pop $\sigma$. Don’t **RIGHT** if the first node in $\beta$ has a head.

Transition Stack $\sigma$ Buffer $\beta$ Relations $A$

<table>
<thead>
<tr>
<th>Transition</th>
<th>Stack $\sigma$</th>
<th>Buffer $\beta$</th>
<th>Relations $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>init $\rightarrow$</td>
<td>[root]</td>
<td>[colorless, green, . . . , furiously]</td>
<td>–</td>
</tr>
<tr>
<td>SHIFT $\rightarrow$</td>
<td>[root, colorless]</td>
<td>[green, ideas, sleep, furiously]</td>
<td>–</td>
</tr>
<tr>
<td>SHIFT $\rightarrow$</td>
<td>[root, colorless, green]</td>
<td>[ideas, sleep, furiously]</td>
<td>–</td>
</tr>
<tr>
<td>LEFT $\rightarrow$</td>
<td>[root, colorless]</td>
<td>[ideas, sleep, furiously]</td>
<td>$A \cup (\text{ideas} \rightarrow \text{green})$</td>
</tr>
</tbody>
</table>
Dependency Grammars
Arc-Standard Parsing

Arc-Standard is a transition based parser with a stack $\sigma$ and 3 transitions $T$:

- **SHIFT**  Remove the first node from $\beta$ and push it to $\sigma$.
- **LEFT**   Add an arc from the first node in $\beta$ to the top of $\sigma$.
  Pop $\sigma$. Don’t LEFT if top of stack is $\text{root}$ or top of stack has a head.

- **RIGHT**  Add an arc from the top of $\sigma$ to the first node in $\beta$.
  Replace the first node in $\beta$ with the top of $\sigma$.
  Pop $\sigma$. Don’t RIGHT if the first node in $\beta$ already has a head.

```
colorless green ideas sleep furiously

Transition       Stack $\sigma$  Buffer $\beta$  Relations $A$
```

```
...  
| SHIFT $\rightarrow$ | [root, sleep] | [furiously] |
| RIGHT $\rightarrow$  | [root]        | [sleep]     |
|                   | $A \cup (\text{sleep} \rightarrow \text{furiously})$ |
```

© WIEGMANN/WOLSKA/WACHSMUTH/HAGEN/POTTHAST/STEIN 2023
Dependency Grammars
Arc-Standard Parsing

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

colorless green ideas sleep furiously

<table>
<thead>
<tr>
<th>Transition</th>
<th>Stack $\sigma$</th>
<th>Buffer $\beta$</th>
<th>Relations $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>init $\rightarrow$</td>
<td>[root]</td>
<td>[colorless, green, . . . , furiously]</td>
<td>$-$</td>
</tr>
<tr>
<td>SHIFT $\rightarrow$</td>
<td>[root, colorless]</td>
<td>[green, ideas, sleep, furiously]</td>
<td>$-$</td>
</tr>
<tr>
<td>SHIFT $\rightarrow$</td>
<td>[root, colorless, green]</td>
<td>[ideas, sleep, furiously]</td>
<td>$-$</td>
</tr>
<tr>
<td>LEFT $\rightarrow$</td>
<td>[root, colorless]</td>
<td>[ideas, sleep, furiously]</td>
<td>$A \cup$ (ideas $\rightarrow$ green)</td>
</tr>
<tr>
<td>LEFT $\rightarrow$</td>
<td>[root]</td>
<td>[ideas, sleep, furiously]</td>
<td>$A \cup$ (ideas $\rightarrow$ colorless)</td>
</tr>
<tr>
<td>SHIFT $\rightarrow$</td>
<td>[root, ideas]</td>
<td>[sleep, furiously]</td>
<td>$-$</td>
</tr>
<tr>
<td>LEFT $\rightarrow$</td>
<td>[root]</td>
<td>[furiously]</td>
<td>$A \cup$ (sleep $\rightarrow$ ideas)</td>
</tr>
<tr>
<td>SHIFT $\rightarrow$</td>
<td>[root, sleep]</td>
<td>[furiously]</td>
<td>$-$</td>
</tr>
<tr>
<td>RIGHT $\rightarrow$</td>
<td>[root]</td>
<td>[sleep]</td>
<td>$A \cup$ (sleep $\rightarrow$ furiously)</td>
</tr>
<tr>
<td>RIGHT $\rightarrow$</td>
<td>[]</td>
<td>[root]</td>
<td>$A \cup$ (root $\rightarrow$ sleep)</td>
</tr>
<tr>
<td>SHIFT $\rightarrow$</td>
<td>[root]</td>
<td>[]</td>
<td>$-$</td>
</tr>
</tbody>
</table>
Dependency Grammars
Arc-Standard Parsing: Oracles

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

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

→ similar to span-based sequence labeling.

Some training examples with class $c_i$:

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>$o(C_{i-1})$</th>
<th>Configuration $C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stack $\sigma$</td>
<td>Buffer $\beta$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{SHIFT} \rightarrow$ [root, colorless, green]</td>
<td>[ideas, sleep, furiously]</td>
</tr>
<tr>
<td>4</td>
<td>$\text{LEFT} \rightarrow$ [root, colorless]</td>
<td>[ideas, sleep, furiously]</td>
</tr>
<tr>
<td>5</td>
<td>$\text{LEFT} \rightarrow$ [root]</td>
<td>[ideas, sleep, furiously]</td>
</tr>
</tbody>
</table>
Dependency Grammars

Arc-Standard Parsing: Oracles

Training 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 \texttt{LEFT} if \((\text{First of } \beta \rightarrow \text{Top of } \sigma)\) is in the reference parse.
  2. Else, use \texttt{RIGHT} if
     (a) \((\text{Top of } \sigma \rightarrow \text{First of } \beta)\) is in the reference parse and
     (b) all dependents of First of \(\beta\) are assigned.
        otherwise, First of \(\beta\) would vanish before all dependents were assigned.
  3. Else, use \texttt{SHIFT}.

The \texttt{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 transition rules. *Arc-eager*, for example, adds a `REDUCE` operator.
- Since the greedy transition 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
  \[ T = \{ \text{\textsc{shift}}, \text{\textsc{right}_{nsubj}}, \text{\textsc{left}_{nsubj}}, \text{\textsc{right}_{dobj}}, \ldots \} \]
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 transition-based parsers cannot parse non-projective trees.
- Trees are projective when generated from CFG’s via head-finding rules.
- In non-projective trees, the arcs overlap.
Dependency Grammars
Graph-based Parsing

Idea: Use graph-algorithms 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?  
   → Machine Learning
2. How to find the best parse?  
   → 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: \( \frac{4}{5} = 0.8 \).
  
  green has the wrong head.

Labeled Attachment Score:
- Fraction of correctly attached heads and labels.
- Example: \( \frac{3}{5} = 0.6 \).
  
  green has the wrong head.
  (sleep \( \rightarrow \) furiously) has a wrong label.

Reference parse:
```
colorless green ideas sleep furiously
```
```
  root
  ↓ nsubj
  ↓ nmod
  ↓ amod
```

System output:
```
colorless green ideas sleep furiously
```
```
  root
  ↓ nsubj
  ↓ dobj
```

© WIEGMANN/WOLSKA/WACHSMUTH/HAGEN/POTTHAST/STEIN 2023
## Dependency Grammars

### Evaluation: Comparison of Methods

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

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

- Note that the progress since 2011 is marginal.