Lecture 7: Syntactic Parsing

LTAT.01.001 – Natural Language Processing
Kairit Sirts (kairit.sirts@ut.ee)
27.03.2019
• **Morphology** – internal structure of words

• **Syntax** – internal structure of sentences

Source: http://amontalenti.com/pub/nlp-training/
Syntactic ambiguity

Call me a cab!

OK, you’re a cab!

Source: https://www.thoughtco.com/syntactic-ambiguity-grammar-1692179
The chicken is ready to eat.
“One morning I shot an elephant in my pajamas.

How he got in my pajamas, I don’t know.”

Groucho Marx
More ambiguous sentences

• I saw the man with binoculars.
• Look at the dog with one eye.
• I watched her duck.
• The peasants are revolting.
• They are cooking apples.
• Stolen painting found by tree.
• Police help dog bite victim.
Syntactic analysis/parsing

• Shallow parsing
• Phrase structure / constituency parsing
• Dependency parsing
The role of syntax in NLP

- Text generation/summarization/machine translation
- Useful features for various information extraction tasks
- Syntactic structure also reflects the semantic relations between the words
Shallow Parsing

Also called chunking
Shallow parsing

• Also called chunking or light parsing
• Split the sentence into non-overlapping syntactic phrases

The morning flight from Denver has arrived.

NP PP NP VP

NP – Noun phrase
PP – Prepositional Phrase
VP – Verb phrase
BIO tagging

• A labelling scheme often used in information extraction problems, treated as a sequence tagging task

The ___ morning flight from ___ Denver has ___ arrived.
B_NP I_NP I_NP B_PP B_NP B_VP I_VP

B_NP – Beginning of a noun phrase
I_NP – Inside a noun phrase
B_VB – Beginning of a verb phrase etc
BIO tagging

• With only noun phrases

The__morning flight from __Denver has arrived.

B_NP  I_NP  I_NP  O  B_NP  O  O

B_NP – Beginning of a noun phrase
I_NP – Inside a noun phrase
O – Outside of a noun phrase
Sequence classifier

• Need annotated data for training: POS-tagged, phrase-annotated

• Use a sequence classifier of your choice
  • CRF with engineered features
  • Neural sequence tagger
Evaluation: precision and recall

\[
\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}
\]

\[
\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}
\]

Evaluation

• From SemEval 2013
  • Strict evaluation – all segments and labels must match
  • Exact boundary matching (regardless of the type)
  • Partial boundary matching (regardless of the type)
  • Type matching (some overlap between entities is required)

• Categories counted:
  • COR: the system’s output and gold standard annotations agree
  • INC: the system’s output and gold standard annotations disagree
  • PAR: the system’s output and gold standard annotations are not identical but have some overlap
  • MIS: there is a gold standard annotation that is not identified by the system
  • SPU: the system labels an entity that does not exist in the gold standard
Constituency Parsing
Constituency parsing

- Full constituency parsing helps to resolve structural ambiguities

Source: Jurafsky and Martin. Speech and Language processing, 3rd ed
Structural ambiguities

- **Attachment ambiguity** – a constituent/phrase can be attached to different places in the tree (the elephant example)

- **Coordination ambiguity**
  - [old [men and women]]
  - Both men and women are old
  - [old men] and [women]
  - Only men are old
Bracketed style

• The trees can be represented linearly with brackets

(S (Pr I)
  (Aux will)
  (VP (V do)
    (NP (Det my)
      (N homework)))_{NP}
  )_{VP}
)_{S}
Context-free grammars

\[ S \rightarrow NP \ VP \]
\[ VP \rightarrow V \ NP \]
\[ VP \rightarrow V \ NP \ PP \]
\[ NP \rightarrow NP \ NP \]
\[ NP \rightarrow NP \ PP \]
\[ NP \rightarrow N \]
\[ NP \rightarrow e \]
\[ PP \rightarrow P \ NP \]

\[ N \rightarrow \text{people} \]
\[ N \rightarrow \text{fish} \]
\[ N \rightarrow \text{tanks} \]
\[ N \rightarrow \text{rods} \]
\[ V \rightarrow \text{people} \]
\[ V \rightarrow \text{fish} \]
\[ V \rightarrow \text{tanks} \]
\[ P \rightarrow \text{with} \]

- \[ G = (T, N, S, R) \]
  - \[ T \] is a set of terminal symbols
  - \[ N \] is a set of nonterminal symbols
  - \[ S \] is the start symbol \( (S \in N) \)
  - \[ R \] is a set of rules/productions of the form \[ X \rightarrow \gamma \]
    - \[ X \in N \] and \[ \gamma \in (N \cup T)^* \]

- A grammar \( G \) generates a language \( L \).

\textit{people} \textit{fish} \textit{tanks}
\textit{people} \textit{fish} \textit{with} \textit{rods}

Source: Statistical Natural Language Parsing, Stanford lecture.
Probabilistic CFGs

- $G = (T, N, S, R, P)$
  - $T$ is a set of terminal symbols
  - $N$ is a set of nonterminal symbols
  - $S$ is the start symbol ($S \in N$)
  - $R$ is a set of rules/productions of the form $X \rightarrow \gamma$
  - $P$ is a probability function
    - $P: R \rightarrow [0,1]$
    - $\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$

- A grammar $G$ generates a language model $L$. 

Source: Statistical Natural Language Parsing, Stanford lecture.
A PCFG

\[
\begin{align*}
S & \rightarrow NP \ VP & 1.0 \\
VP & \rightarrow V \ NP & 0.6 \\
VP & \rightarrow V \ NP \ PP & 0.4 \\
NP & \rightarrow NP \ NP & 0.1 \\
NP & \rightarrow NP \ PP & 0.2 \\
NP & \rightarrow N & 0.7 \\
PP & \rightarrow P \ NP & 1.0 \\
N & \rightarrow people & 0.5 \\
N & \rightarrow fish & 0.2 \\
N & \rightarrow tanks & 0.2 \\
N & \rightarrow rods & 0.1 \\
V & \rightarrow people & 0.1 \\
V & \rightarrow fish & 0.6 \\
V & \rightarrow tanks & 0.3 \\
P & \rightarrow with & 1.0
\end{align*}
\]

Source: Statistical Natural Language Parsing, Stanford lecture.
The probability of strings and trees

- $P(t)$ – The probability of a tree $t$ is the product of the probabilities of the rules used to generate it.
- $P(s)$ – The probability of the string $s$ is the sum of the probabilities of the trees which have that string as their yield

\[ P(s) = \sum_j P(s, t) \quad \text{where } t \text{ is a parse of } s \]

\[ = \sum_j P(t) \]

Source: Statistical Natural Language Parsing, Stanford lecture.
Exercise

• Compute the probability for the tree:
PCFG for efficient parsing

- For efficient parsing the rules should be **unary** or **binary**
- Chomsky normal form – all rules have the form:
  - $X \rightarrow Y \ Z$
  - $X \rightarrow w$
  - $X, Y, Z$ - non-terminal symbols
  - $w$ – terminal symbol
  - No epsilon rules
Before binarization

S → NP VP
VP → V NP
VP → V NP PP
NP → NP NP
NP → NP PP
NP → N
NP → e
PP → P NP

N → people
N → fish
N → tanks
N → rods
V → people
V → fish
V → tanks
P → with

Source: Statistical Natural Language Parsing, Stanford lecture.
After binarization

S → NP VP
VP → V NP
S → V NP
VP → V @VP_V
@VP_V → NP PP
S → V @S_V
@S_V → NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP PP
NP → P NP
PP → P NP

Source: Statistical Natural Language Parsing, Stanford lecture.
Before and after binarization

Source: Statistical Natural Language Parsing, Stanford lecture.
Finding the most likely tree: CKY parsing

• Dynamic programming algorithm
• Proceeds bottom-up and performs Viterbi on trees
CKY parsing

• For a full example look at the slides at http://slideplayer.com/slide/4559350/
CKY parsing

S → NP VP 0.9
S → VP 0.1
VP → V NP 0.5
VP → V 0.1
VP → V @VP_V 0.3
VP → V PP 0.1
@VP_V → NP PP 1.0
NP → NP NP 0.1
NP → NP PP 0.2
NP → N 0.7
PP → P NP 1.0
N → people 0.5
N → fish 0.2
N → tanks 0.2
N → rods 0.1
V → people 0.1
V → fish 0.6
V → tanks 0.3
P → with 1.0

Source: Statistical Natural Language Parsing, Stanford lecture.
CKY parsing

S \rightarrow NP \ VP \ 0.9
S \rightarrow VP \ 0.1
VP \rightarrow V \ NP \ 0.5
VP \rightarrow V \ 0.1
VP \rightarrow V \@VP_\ V \ 0.3
VP \rightarrow V \ PP \ 0.1
@VP_\ V \rightarrow NP \ PP \ 1.0
NP \rightarrow NP \ NP \ 0.1
NP \rightarrow NP \ PP \ 0.2
NP \rightarrow N \ 0.7
PP \rightarrow P \ NP \ 1.0
N \rightarrow people \ 0.5
N \rightarrow fish \ 0.2
N \rightarrow tanks \ 0.2
N \rightarrow rods \ 0.1
V \rightarrow people \ 0.1
V \rightarrow fish \ 0.6
V \rightarrow tanks \ 0.3
P \rightarrow with \ 1.0

Source: Statistical Natural Language Parsing, Stanford lecture.
Evaluating constituency parsing

**Gold standard brackets:**
S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9), NP-(4:6), PP-(6-9), NP-(7,9), NP-(9:10)

**Candidate brackets:**
S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:6), PP-(6-10), NP-(7,10)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labeled Precision</td>
<td>3/7 = 42.9%</td>
</tr>
<tr>
<td>Labeled Recall</td>
<td>3/8 = 37.5%</td>
</tr>
<tr>
<td>LP/LR F1</td>
<td>40.0%</td>
</tr>
</tbody>
</table>

\[
F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Source: Statistical Natural Language Parsing, Stanford lecture.
Dependency Parsing
Dependency parsing

- Dependency parse is a directed graph $G = (V, A)$
  - $V$ – the set of vertices corresponding to words
  - $A$ – the set of nodes corresponding to dependency relations

[Diagram showing a dependency parse tree with labels for dependent, head, root of the sentence, labelled dependency relation, and corenlp.run for visualization.]
Dependency parsing

Source: Jurafsky and Martin. Speech and language processing, 3rd ed
Dependency relations

• The arrows connect **heads** and their **dependents**
• The main verb is the **head** or the **root** of the whole sentence
• The arrows are labelled with **grammatical functions/dependency relations**

![Diagram showing dependency relations with labels for heads, dependents, and arrows indicating grammatical functions/dependency relations.](image-url)
Properties of a dependency graph

A dependency tree is a directed graph that satisfies the following constraints:

1. There is a single designated root node that has no incoming arcs
   • Typically the main verb of the sentence
2. With the exception of the root node, each node has exactly one incoming arc
   • Each dependent has a single head
3. There is a unique path from the root node to each vertex in V
   • The graph is acyclic and connected
Projectivity

- Projective trees – there are no arc crossings in the dependency graphs
- Non-projective trees - crossings due to free word order

Source: Jurafsky and Martin. Speech and language processing, 3rd ed
# Dependency relations

## Clausal Argument Relations

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSUBJ</td>
<td>Nominal subject</td>
</tr>
<tr>
<td>DOBJ</td>
<td>Direct object</td>
</tr>
<tr>
<td>IOBJ</td>
<td>Indirect object</td>
</tr>
<tr>
<td>CCOMP</td>
<td>Clausal complement</td>
</tr>
<tr>
<td>XCOMP</td>
<td>Open clausal complement</td>
</tr>
</tbody>
</table>

## Nominal Modifier Relations

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMOD</td>
<td>Nominal modifier</td>
</tr>
<tr>
<td>AMOD</td>
<td>Adjectival modifier</td>
</tr>
<tr>
<td>NUMMOD</td>
<td>Numeric modifier</td>
</tr>
<tr>
<td>APPOS</td>
<td>Appositional modifier</td>
</tr>
<tr>
<td>DET</td>
<td>Determiner</td>
</tr>
<tr>
<td>CASE</td>
<td>Prepositions, postpositions and other case markers</td>
</tr>
</tbody>
</table>

## Other Notable Relations

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONJ</td>
<td>Conjunct</td>
</tr>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
</tr>
</tbody>
</table>

Source: Jurafsky and Martin. Speech and language processing, 3rd ed
## Dependency relations

<table>
<thead>
<tr>
<th>Relation</th>
<th>Examples with head and dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSUBJ</td>
<td>United canceled the flight.</td>
</tr>
<tr>
<td>DOBJ</td>
<td>United diverted the flight to Reno.</td>
</tr>
<tr>
<td></td>
<td>We booked her the first flight to Miami.</td>
</tr>
<tr>
<td>IOBJ</td>
<td>We booked her the flight to Miami.</td>
</tr>
<tr>
<td>NMOD</td>
<td>We took the morning flight.</td>
</tr>
<tr>
<td>AMOD</td>
<td>Book the cheapest flight.</td>
</tr>
<tr>
<td>NUMMOD</td>
<td>Before the storm JetBlue canceled 1000 flights.</td>
</tr>
<tr>
<td>APPOS</td>
<td>United, a unit of UAL, matched the fares.</td>
</tr>
<tr>
<td>DET</td>
<td>The flight was canceled.</td>
</tr>
<tr>
<td></td>
<td>Which flight was delayed?</td>
</tr>
<tr>
<td>CONJ</td>
<td>We flew to Denver and drove to Steamboat.</td>
</tr>
<tr>
<td>CC</td>
<td>We flew to Denver and drove to Steamboat.</td>
</tr>
<tr>
<td>CASE</td>
<td>Book the flight through Houston.</td>
</tr>
</tbody>
</table>

Source: Jurafsky and Martin. Speech and language processing, 3rd ed
| Source: universaldependencies.org |

<table>
<thead>
<tr>
<th>Core arguments</th>
<th>Nominals</th>
<th>Clauses</th>
<th>Modifier words</th>
<th>Function Words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nsubj</td>
<td>csubj</td>
<td>advmod</td>
<td>aux</td>
</tr>
<tr>
<td></td>
<td>obj</td>
<td>ccomp</td>
<td>discourse</td>
<td>cop</td>
</tr>
<tr>
<td></td>
<td>obj</td>
<td>xcomp</td>
<td></td>
<td>mark</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-core dependents</th>
<th>Obl</th>
<th>vocative</th>
<th>expl</th>
<th>dislocated</th>
<th>advcl</th>
<th>advmod*</th>
<th>discourse</th>
<th>aux</th>
<th>cop</th>
<th>mark</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Nominal dependents</th>
<th>nmod</th>
<th>appos</th>
<th>nummod</th>
<th>acl</th>
<th>amod</th>
<th>det</th>
<th>clf</th>
<th>case</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Coordination</th>
<th>MWE</th>
<th>Loose</th>
<th>Special</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>conj</td>
<td></td>
<td>fixed</td>
<td>list</td>
<td>orphan</td>
</tr>
<tr>
<td>cc</td>
<td></td>
<td>flat</td>
<td>parataxis</td>
<td>goeswith</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compound</td>
<td>reparandum</td>
<td>root</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>punct</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>root</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dep</td>
</tr>
</tbody>
</table>
Universal dependencies

- [http://universaldependencies.org/](http://universaldependencies.org/)
- Annotated treebanks in many languages
- Uniform annotation scheme across all languages:
  - Universal POS tags
  - Universal morphological features
  - Universal dependency relations
- All data in CONLL-U tabulated format
CONLL-U format

• A standard tabular format for certain type of annotated data
• Each word is in a separate line
• 10 tab-separated columns on each line:
  1. Word index
  2. The word itself
  3. Lemma
  4. Universal POS
  5. Language specific POS
  6. Morphological features
  7. **Head of the current word**
  8. **Dependency relation to the head**
  9. Enhanced dependencies
  10. Any other annotation
CONLL-U format: example

# text = They buy and sell books.

<table>
<thead>
<tr>
<th></th>
<th>They</th>
<th>they</th>
<th>PRON</th>
<th>PRP</th>
<th>Case=Nom</th>
<th>Number=Plur</th>
<th>2</th>
<th>nsubj</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>buy</td>
<td>buy</td>
<td>VERB</td>
<td>VBP</td>
<td>Number=Plur</td>
<td>Person=3</td>
<td>Tense=Pres</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>and</td>
<td>and</td>
<td>CONJ</td>
<td>CC</td>
<td>_</td>
<td></td>
<td>4</td>
<td>cc</td>
</tr>
<tr>
<td>4</td>
<td>sell</td>
<td>sell</td>
<td>VERB</td>
<td>VBP</td>
<td>Number=Plur</td>
<td>Person=3</td>
<td>Tense=Pres</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>books</td>
<td>book</td>
<td>NOUN</td>
<td>NNS</td>
<td>Number=Plur</td>
<td></td>
<td>2</td>
<td>obj</td>
</tr>
<tr>
<td>6</td>
<td>.</td>
<td>.</td>
<td>PUNCT</td>
<td>.</td>
<td>_</td>
<td></td>
<td>2</td>
<td>punct</td>
</tr>
</tbody>
</table>

# text = I had no clue.

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>I</th>
<th>PRON</th>
<th>PRP</th>
<th>Case=Nom</th>
<th>Number=Sing</th>
<th>Person=1</th>
<th>2</th>
<th>nsubj</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>had</td>
<td>have</td>
<td>VERB</td>
<td>VBD</td>
<td>Number=Sing</td>
<td>Person=1</td>
<td>Tense=Past</td>
<td>0</td>
<td>root</td>
</tr>
<tr>
<td>3</td>
<td>no</td>
<td>no</td>
<td>DET</td>
<td>DT</td>
<td>PronType=Neg</td>
<td></td>
<td>4</td>
<td>det</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>clue</td>
<td>clue</td>
<td>NOUN</td>
<td>NN</td>
<td>Number=Sing</td>
<td></td>
<td>2</td>
<td>obj</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.</td>
<td>.</td>
<td>PUNCT</td>
<td>.</td>
<td>_</td>
<td></td>
<td>2</td>
<td>punct</td>
<td></td>
</tr>
</tbody>
</table>
Dependency parsing methods

• Transition-based parsing
  • stack-based algorithms/shift-reduce parsing
  • only generate projective trees

• Graph-based algorithms
  • can also generate non-projective trees
Transition-based parsing

• Three main components:
  • Stack
  • Buffer
  • Set of dependency relations

• A configuration is the current state of the stack, buffer and the relation set

Source: Jurafsky and Martin. Speech and Language processing, 3rd ed
Arc-standard parsing system

• Initial configuration: \( \sigma = [\text{ROOT}], \beta = [w_1, \ldots, w_n], A = \emptyset \)
  • Stack contains the ROOT symbol
  • Buffer contains all words in the sentence
  • Dependency relation set is empty

• At each step perform either:
  • **Shift** – move a word from the buffer to the stack:
    \[
    \sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A
    \]
  • **LeftArc** – left arc between top two words in the stack, pop the second word:
    \[
    \sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{w_j, w_i\}
    \]
  • **RightArc** – right arc between top two words in the stack, pop the first word:
    \[
    \sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_i, \beta, A \cup \{w_i, w_j\}
    \]
Oracle

• The annotated data is in the form of a treebank
  • Each sentence is annotated with its dependency tree

• The task of the transition-based parser is to predict the correct parsing operation at each step:
  • Input is configuration
  • Output is parsing action: Shift, RightArc or LeftArc

• The role of the oracle is to return the correct parsing operation for each configuration in the training set
  • It creates a training set for the parsing model
Oracle

• Choose **LeftArc** if it produces a correct head-dependent relation given the reference parse and the current configuration

• Choose **RightArc** if:
  • It produces a correct head-dependent relation given the reference parse and the current configuration
  • All of the dependents of the word at the top of the stack have already been assigned

• Otherwise choose **Shift**
Example

Shift: \[ \sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A \]

LeftArc: \[ \sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{w_j, w_i\} \]

RightArc: \[ \sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_i, \beta, A \cup \{w_i, w_j\} \]
# Example

<table>
<thead>
<tr>
<th>Stack</th>
<th>Buffer</th>
<th>Action</th>
<th>Arc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Stack</th>
<th>Buffer</th>
<th>Action</th>
<th>Arc</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ROOT]</td>
<td>[The, cat, sat, on, the, mat]</td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>[ROOT, The]</td>
<td>[cat, sat, on, the, mat]</td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>[ROOT, The, cat]</td>
<td>[sat, on, the, mat]</td>
<td>Left-Arc</td>
<td>det(The &lt;-- cat)</td>
</tr>
<tr>
<td>[ROOT, cat]</td>
<td>[sat, on, the, mat]</td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>[ROOT, cat, sat]</td>
<td>[on, the, mat]</td>
<td>Left-Arc</td>
<td>nsubj(cat &lt;-- sat)</td>
</tr>
<tr>
<td>[ROOT, sat]</td>
<td>[on, the, mat]</td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>[ROOT, sat, on]</td>
<td>[the, mat]</td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>[ROOT, sat, on, the]</td>
<td>[mat]</td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>[ROOT, sat, on, the, mat]</td>
<td>[]</td>
<td>Left-Arc</td>
<td>det(the &lt;-- mat)</td>
</tr>
<tr>
<td>[ROOT, sat, on, mat]</td>
<td>[]</td>
<td>Left-Arc</td>
<td>case(on &lt;-- mat)</td>
</tr>
<tr>
<td>[ROOT, sat, mat]</td>
<td>[]</td>
<td>Right-Arc</td>
<td>nmod(sat --&gt; mat)</td>
</tr>
<tr>
<td>[ROOT, sat]</td>
<td>[]</td>
<td>Right-Arc</td>
<td>root(ROOT, sat)</td>
</tr>
<tr>
<td>[ROOT]</td>
<td>[]</td>
<td>Done</td>
<td></td>
</tr>
</tbody>
</table>
Typical features

• First word from the stack
• second word from the stack
• The POS of the first word in the stack
• The POS of the second word in the stack
• The first word in the buffer
• The POS of the first word in the buffer
• The word and the POS of the top word in the stack
• ...

Evaluation

**Unlabelled attachment score:**
- The proportion of correct head attachments

**Labelled attachment score:**
- The proportion of correct head attachments labelled with the correct relation

**Label accuracy**
- The proportion of correct incoming relation labels ignoring the head
Evaluation

UAS =
LAS =
LA =

Source: Jurafsky and Martin. Speech and Language processing, 3rd ed
UAS = 5/6
LAS = 4/6
LA = 4/6

Source: Jurafsky and Martin. Speech and Language processing, 3rd ed
Neural Dependency parsers

Configuration:

```
s_2
  the

s_1
  jumped

s_0
  over

b_0
  the

b_1
  the

b_2
  lazy
dog

b_3
  ROOT
```

Scoring:

```
(Score_{Left Arc}, Score_{Right Arc}, Score_{Shift})
```

Kipperwasser and Goldberg, 2016. Simple and Accurate Parsing Using Bidirectional LSTM Feature Representations
Neural Dependency parsers

Dyer et al., 2015. Transition-based Dependency Parsing with Stack Long Short-Term Memory
Stack implemented with an LSTM

Dyer et al., 2015. Transition-based Dependency Parsing with Stack Long Short-Term Memory
Graph-based neural parser

Dozat et al., 2017. Stanford’s Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task
Trained parsers

• UDPipe – Parsing pipeline for UD languages by Charles University (Czech): http://ufal.mff.cuni.cz/udpipe

• StanfordNLP – Parsing pipeline for UD languages by Stanford University: https://stanfordnlp.github.io/stanfordnlp/

• Spacy parser for English and German (and few other languages): https://spacy.io/
Parsing Estonian

• Estnltk has two parsers:
  • A trained MaltParser model
  • A rule-based parser based on Constraint Grammar


<table>
<thead>
<tr>
<th></th>
<th>LAS</th>
<th>UAS</th>
<th>LA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford</td>
<td>76.3</td>
<td>80.4</td>
<td>87.6</td>
</tr>
<tr>
<td>spaCy</td>
<td>76.6</td>
<td>82.2</td>
<td>85.5</td>
</tr>
<tr>
<td>SyntaxNet</td>
<td>78.3</td>
<td>83.4</td>
<td>87.1</td>
</tr>
<tr>
<td>MaltParser</td>
<td>80.0</td>
<td>83.6</td>
<td>89.2</td>
</tr>
<tr>
<td>UDPipe</td>
<td>79.1</td>
<td>82.5</td>
<td>90.1</td>
</tr>
</tbody>
</table>
Parsing Estonian

• CoNLL 2018 Shared task
• Labelled attachment scores (LAS)

<table>
<thead>
<tr>
<th>Tool</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford</td>
<td>76.3</td>
</tr>
<tr>
<td>spaCy</td>
<td>76.6</td>
</tr>
<tr>
<td>SyntaxNet</td>
<td>78.3</td>
</tr>
<tr>
<td>MaltParser</td>
<td>80.0</td>
</tr>
<tr>
<td>UDPipe</td>
<td>79.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team/Lab</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HIT-SCIR (Harbin)</td>
<td>85.35</td>
</tr>
<tr>
<td>2</td>
<td>TurkuNLP (Turku)</td>
<td>84.15</td>
</tr>
<tr>
<td>3</td>
<td>Stanford (Stanford)</td>
<td>83.84</td>
</tr>
<tr>
<td>4</td>
<td>CEA LIST (Paris)</td>
<td>83.52</td>
</tr>
<tr>
<td>5</td>
<td>ICS PAS (Warszawa)</td>
<td>83.46</td>
</tr>
<tr>
<td>6</td>
<td>UDPipe Future (Praha)</td>
<td>83.26</td>
</tr>
<tr>
<td>7</td>
<td>LATTICE (Paris)</td>
<td>82.33</td>
</tr>
<tr>
<td>8</td>
<td>NLP-Cube (București)</td>
<td>82.30</td>
</tr>
<tr>
<td>9</td>
<td>SLT-Interactions (Bengaluru)</td>
<td>81.85</td>
</tr>
<tr>
<td>10</td>
<td>ParisNLP (Paris)</td>
<td>81.78</td>
</tr>
<tr>
<td>11</td>
<td>AntNLP (Shanghai)</td>
<td>81.37</td>
</tr>
<tr>
<td>12</td>
<td>Uppsala (Uppsala)</td>
<td>81.09</td>
</tr>
<tr>
<td>13</td>
<td>UniMelb (Melbourne)</td>
<td>81.02</td>
</tr>
<tr>
<td>14</td>
<td>LeisureX (Shanghai)</td>
<td>80.59</td>
</tr>
<tr>
<td>15</td>
<td>IBM NY (Yorktown Heights)</td>
<td>79.89</td>
</tr>
<tr>
<td>16</td>
<td>Fudan (Shanghai)</td>
<td>76.66</td>
</tr>
<tr>
<td>17</td>
<td>BOUN (İstanbul)</td>
<td>75.47</td>
</tr>
<tr>
<td>18</td>
<td>BASELINE UDPipe 1.2 (Praha)</td>
<td>75.02</td>
</tr>
<tr>
<td>19</td>
<td>Phoenix (Shanghai)</td>
<td>74.65</td>
</tr>
<tr>
<td>20</td>
<td>KParse (İstanbul)</td>
<td>74.52</td>
</tr>
<tr>
<td>21</td>
<td>CUNI x-ling (Praha)</td>
<td>72.66</td>
</tr>
<tr>
<td>22</td>
<td>HUJI (Yerushalayim)</td>
<td>67.45</td>
</tr>
<tr>
<td>23</td>
<td>ONLP lab (Ra’anana)</td>
<td>64.52</td>
</tr>
<tr>
<td>24</td>
<td>ArmParser (Yerevan)</td>
<td>57.44</td>
</tr>
<tr>
<td>25</td>
<td>SParse (İstanbul)</td>
<td>0.00</td>
</tr>
<tr>
<td>26</td>
<td>iParse (Pittsburgh)</td>
<td>0.00</td>
</tr>
</tbody>
</table>