Parsing Algorithms

Human Language from a Computational Perspective
May 30, 2018

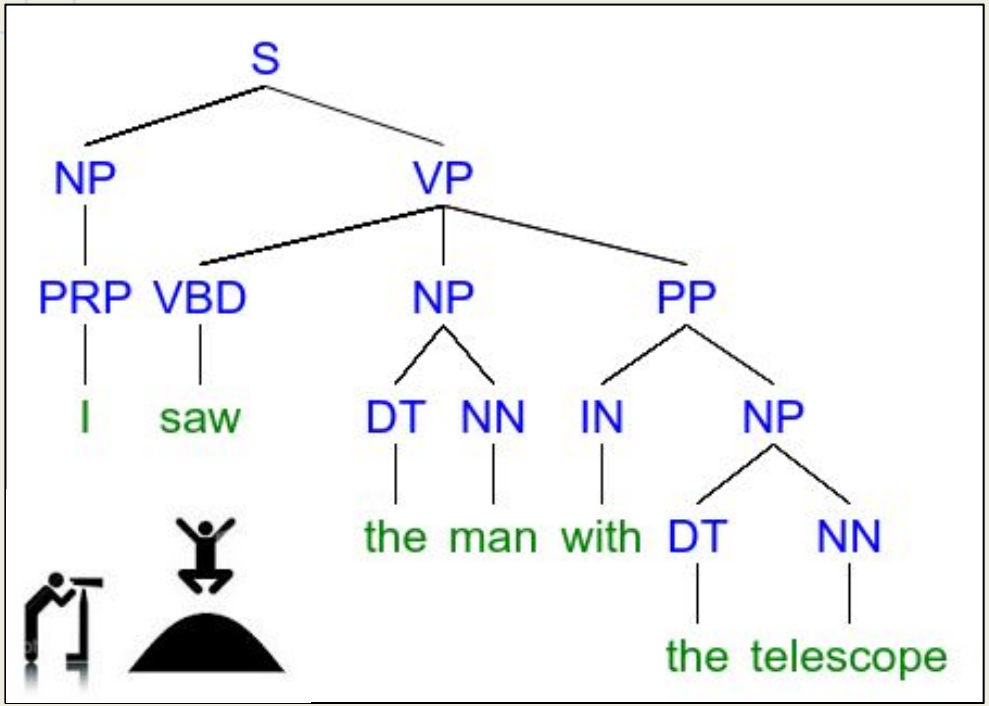
Data structures so far

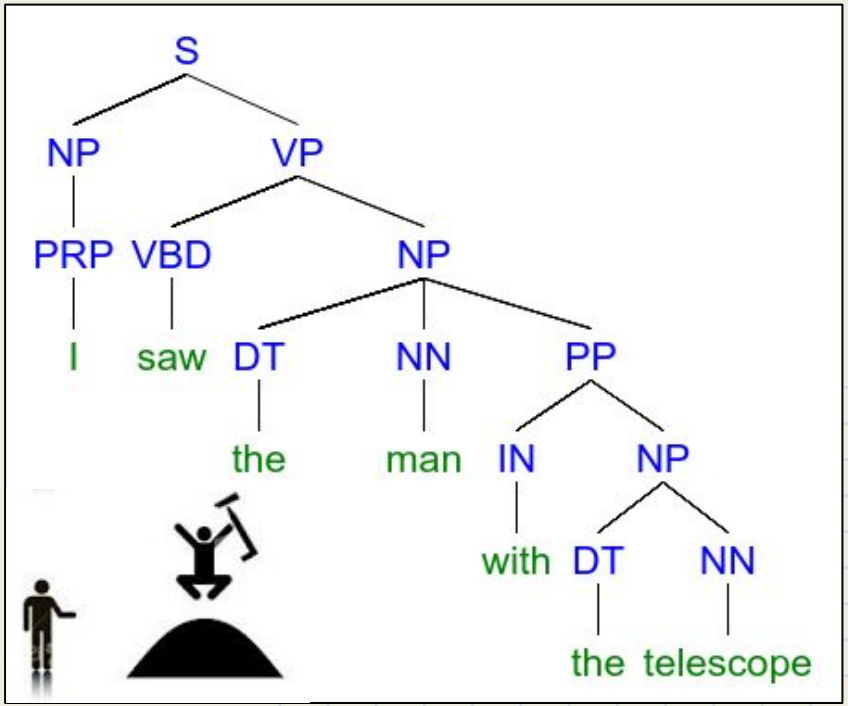
- Lists [3, 16, 8, 0]

  [HE, GAVE, HER, A, BOOK]

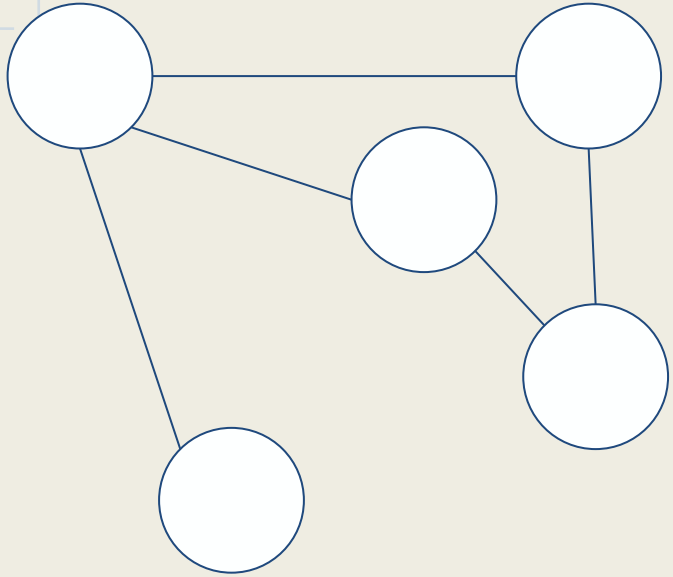
- Tables [THE: 462, FISH: 31, SEE: 9]

Reminder: phrase structure

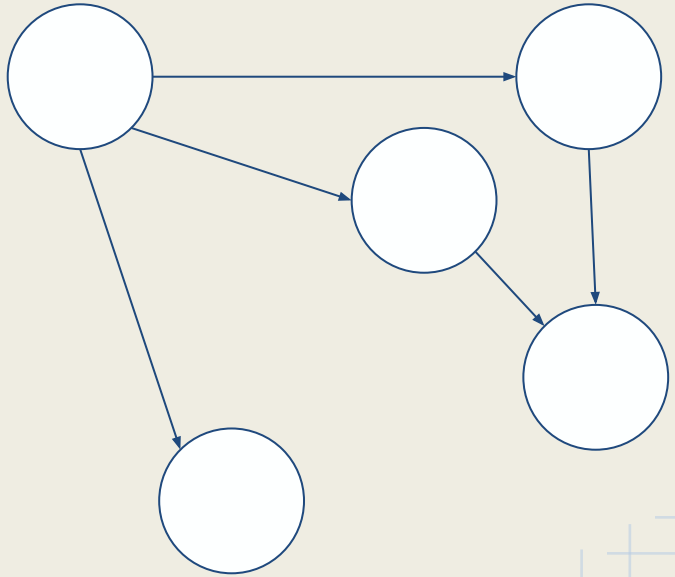




Graphs

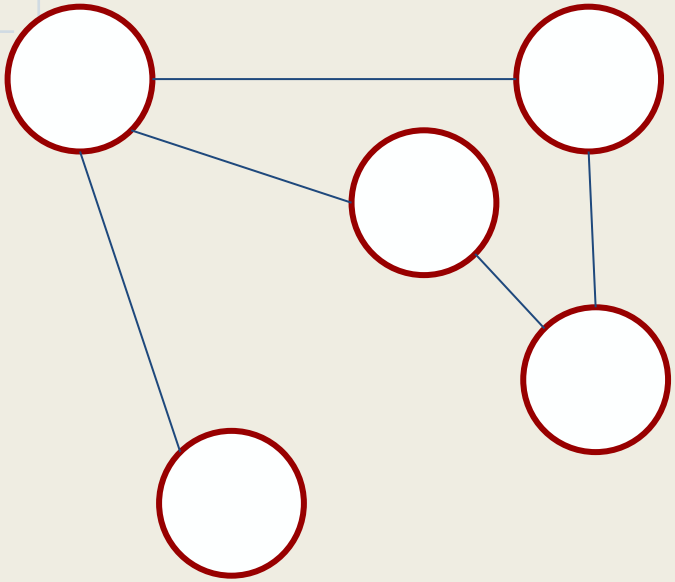


Undirected



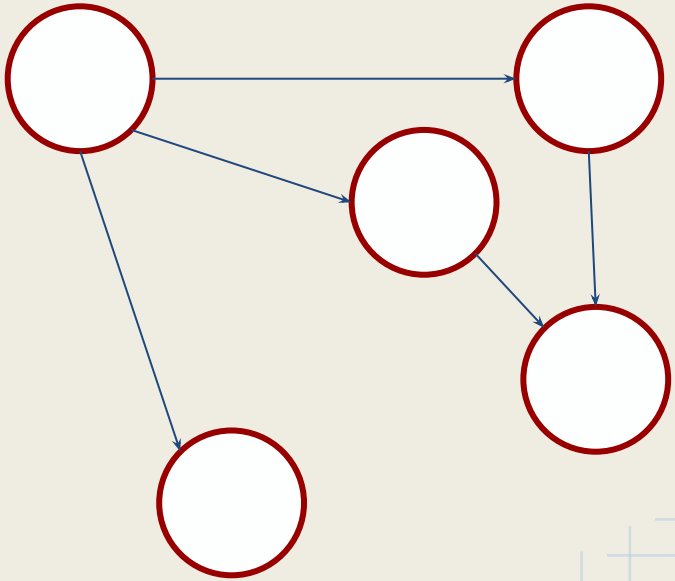
Directed

Graphs



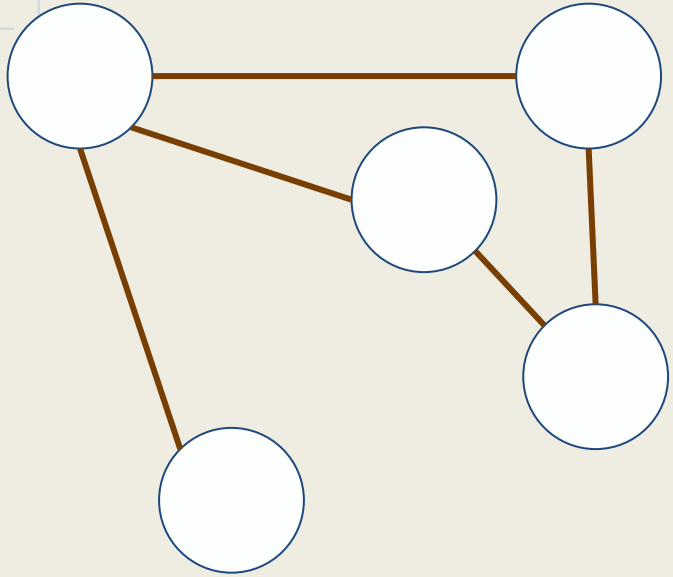
Undirected

Node

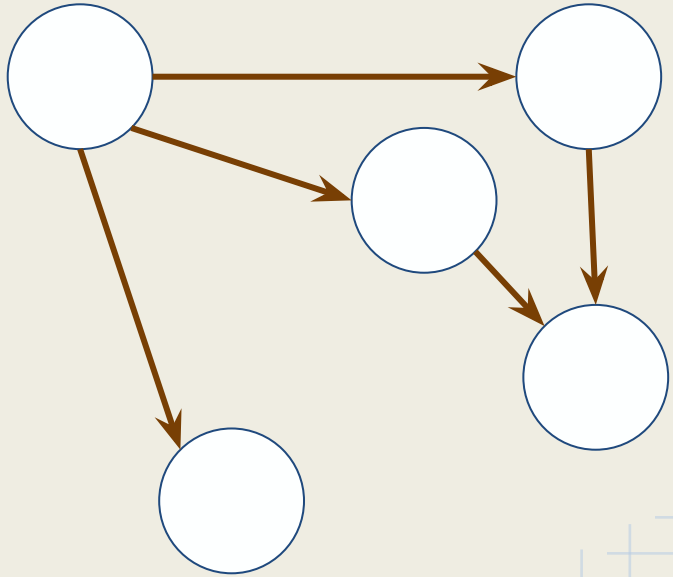


Directed

Graphs



Edge
(arc)

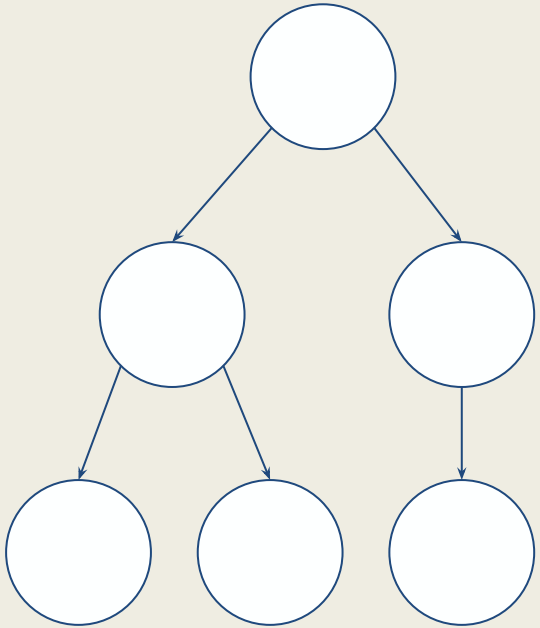


Undirected

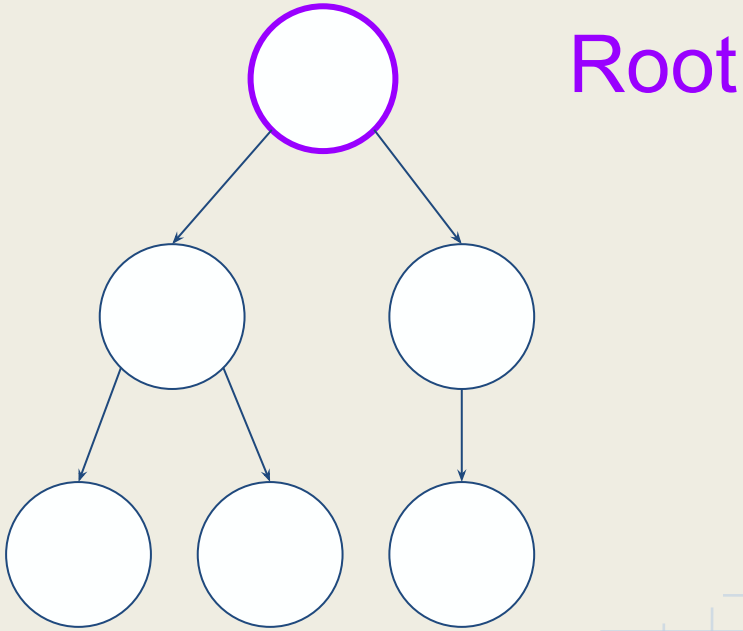
Directed

Trees

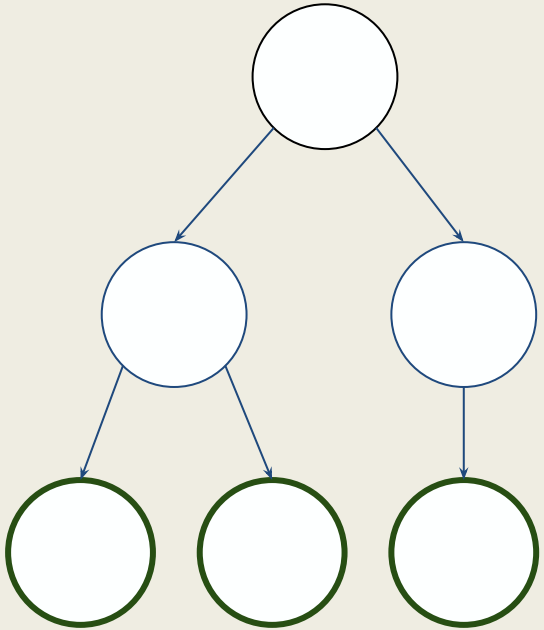
At most one incoming edge per node



Trees

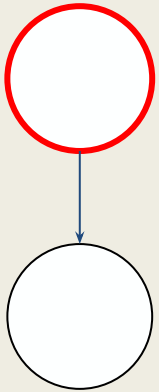


Trees



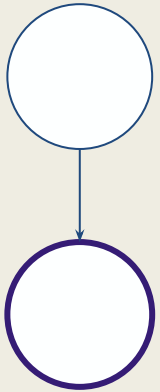
Leaf

Trees



Parent (head)

Trees



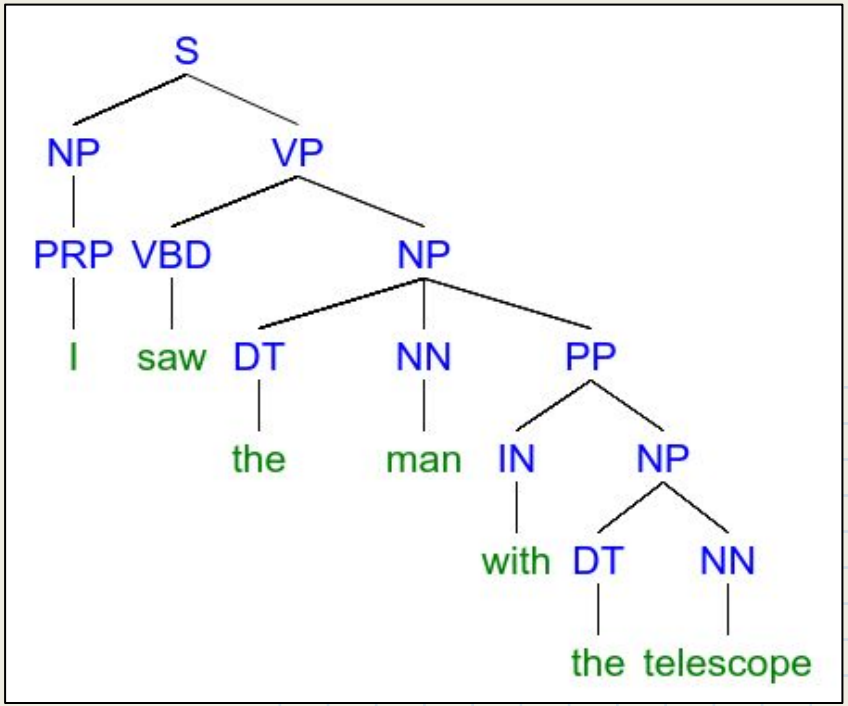
Child (dependent)

Phrase structure trees

Syntactic theory based on phrases.

Nodes have labels.

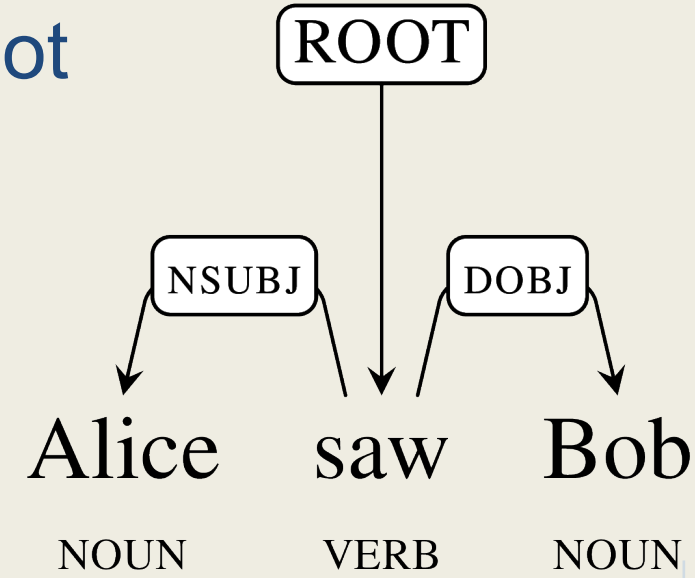
Tokens are leaves.



http://nlp.stanford.edu/software/dependencies_manual.pdf

Dependency parsing

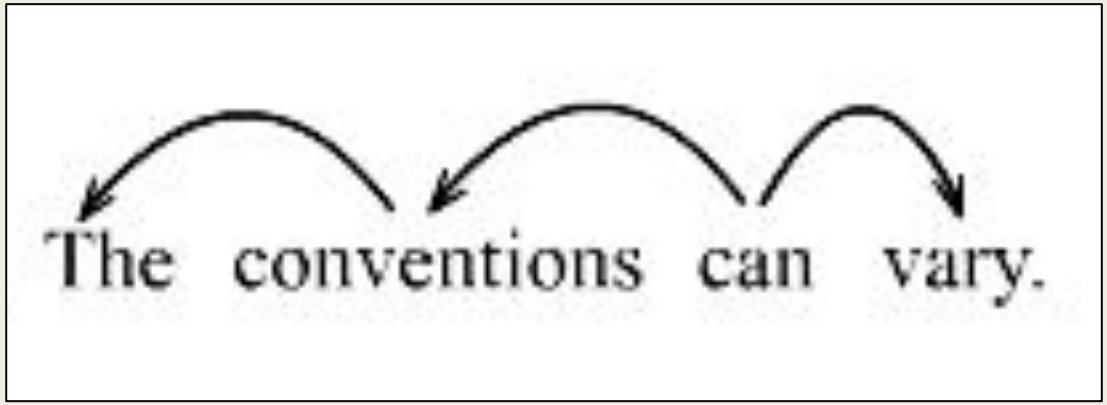
Also syntax, but based on dependencies, not phrases.

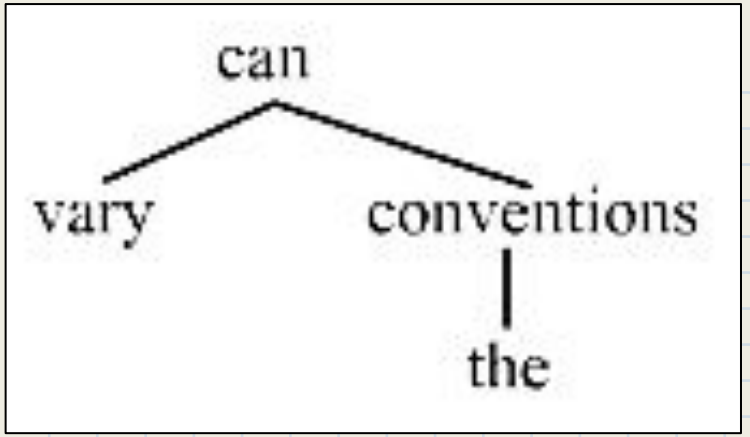


Dependency parsing

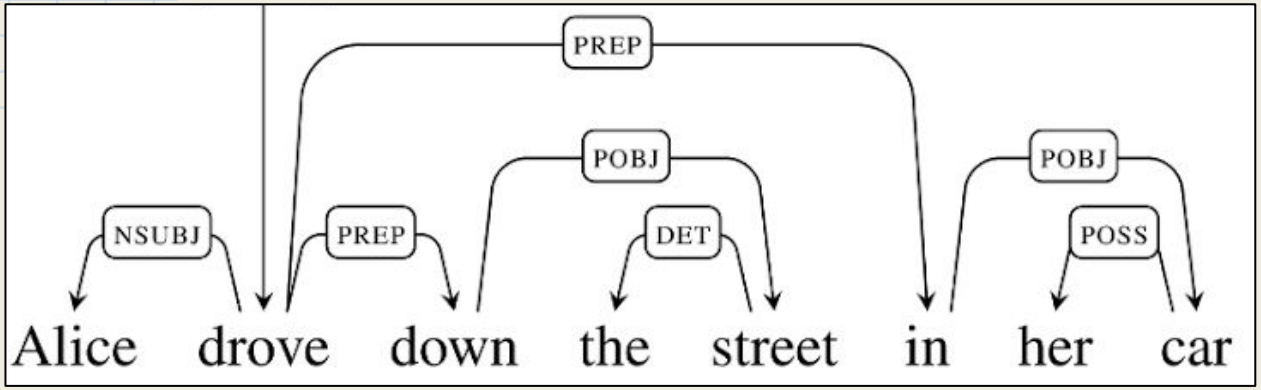
Also a tree, but **edges** are labeled

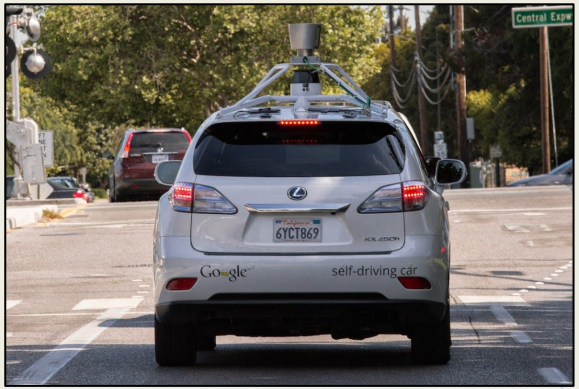
Tokens are all the nodes (not just leaves)

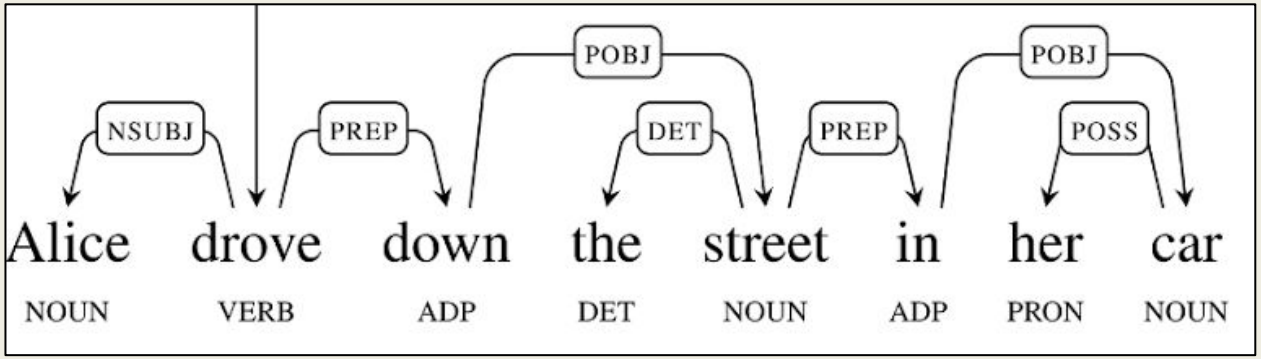




Syntactic ambiguity

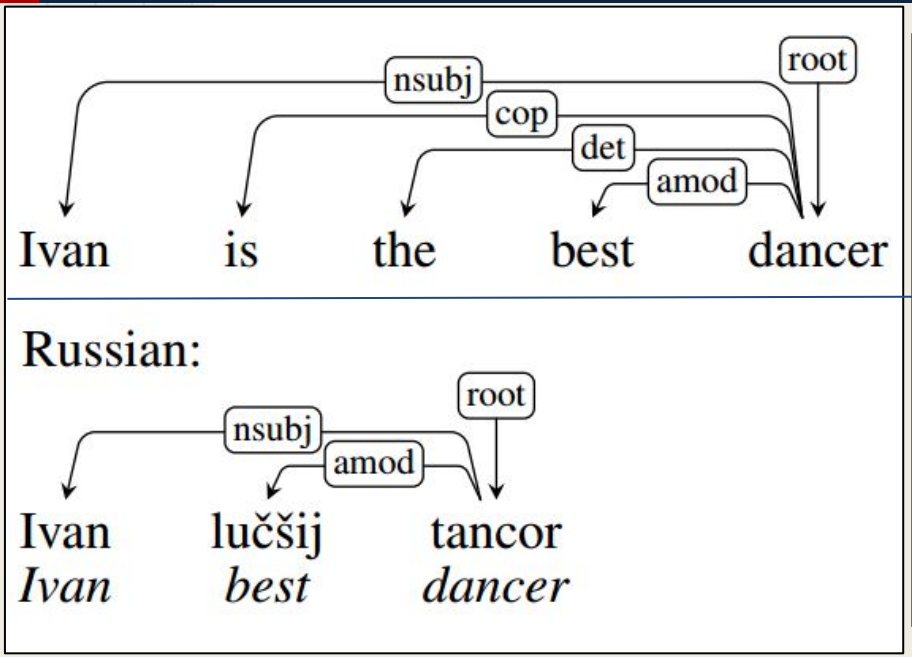


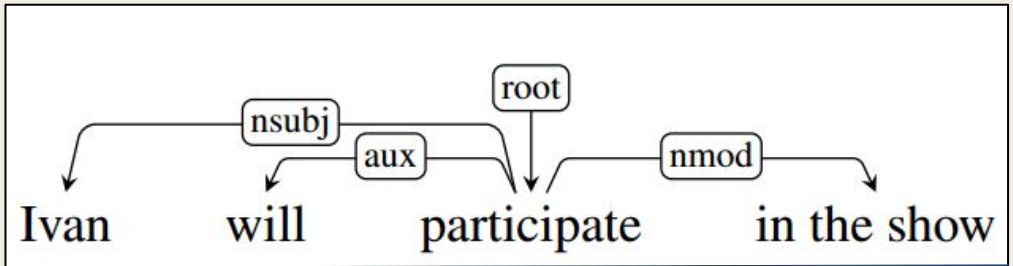




universaldependencies.org

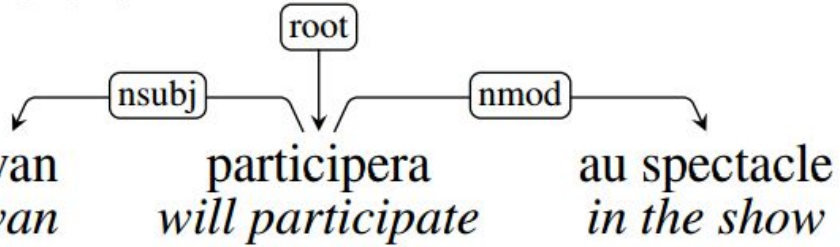
Universal dependencies





Russian:

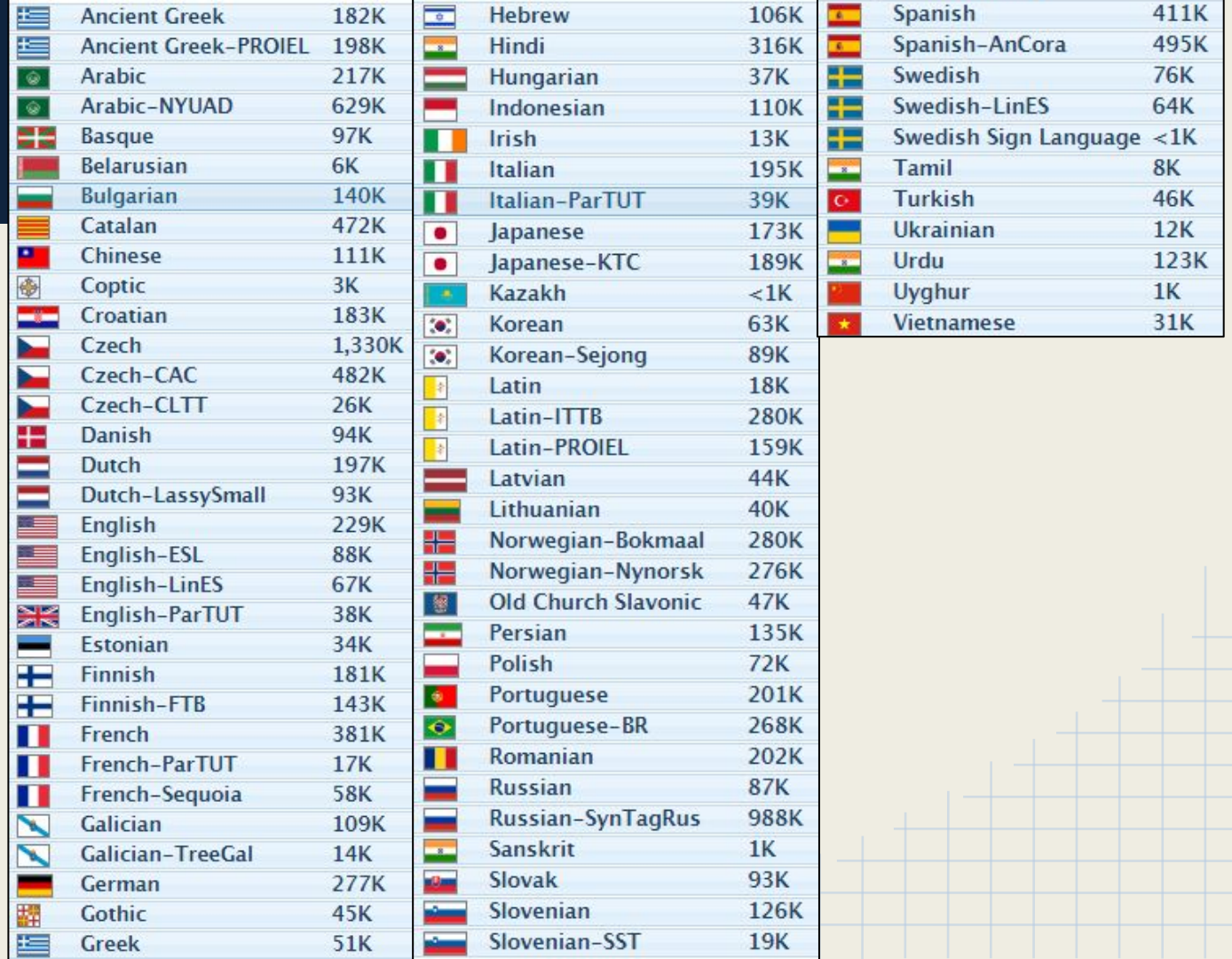
French:



http://universaldependencies.org/u/dep/index.html

universaldependencies.org

Many languages parsed manually.



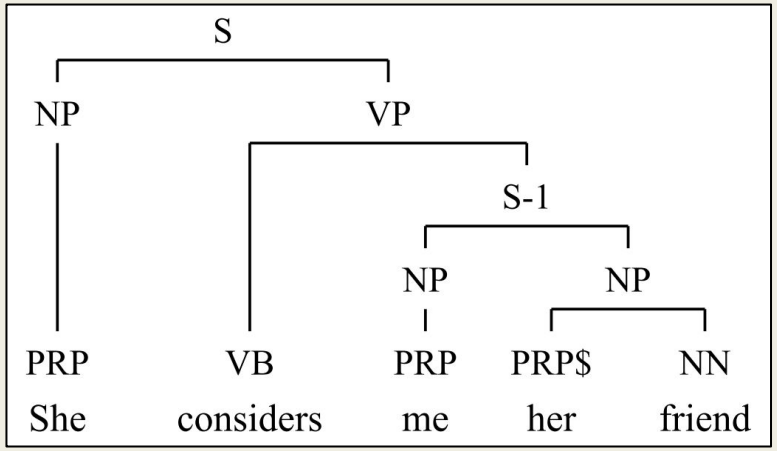
Ancient Greek	182K
Ancient Greek-PROIEL	198K
Arabic	217K
Arabic-NYUAD	629K
Basque	97K
Belarusian	6K
Bulgarian	140K
Catalan	472K
Chinese	111K
Coptic	3K
Croatian	183K
Czech	1,330K
Czech-CAC	482K
Czech-CLTT	26K
Danish	94K
Dutch	197K
Dutch-LassySmall	93K
English	229K
English-ESL	88K
English-LinES	67K
English-ParTUT	38K
Estonian	34K
Finnish	181K
Finnish-FTB	143K
French	381K
French-ParTUT	17K
French-Sequoia	58K
Galician	109K
Galician-TreeGal	14K
German	277K
Gothic	45K
Greek	51K
Hebrew	106K
Hindi	316K
Hungarian	37K
Indonesian	110K
Irish	13K
Italian	195K
Italian-ParTUT	39K
Japanese	173K
Japanese-KTC	189K
Kazakh	<1K
Korean	63K
Korean-Sejong	89K
Latin	18K
Latin-ITTB	280K
Latin-PROIEL	159K
Latvian	44K
Lithuanian	40K
Norwegian-Bokmaal	280K
Norwegian-Nynorsk	276K
Old Church Slavonic	47K
Persian	135K
Polish	72K
Portuguese	201K
Portuguese-BR	268K
Romanian	202K
Russian	87K
Russian-SynTagRus	988K
Sanskrit	1K
Slovak	93K
Slovenian	126K
Slovenian-SST	19K
Spanish	411K
Spanish-AnCora	495K
Swedish	76K
Swedish-LinES	64K
Swedish Sign Language	<1K
Tamil	8K
Turkish	46K
Ukrainian	12K
Urdu	123K
Uyghur	1K
Vietnamese	31K

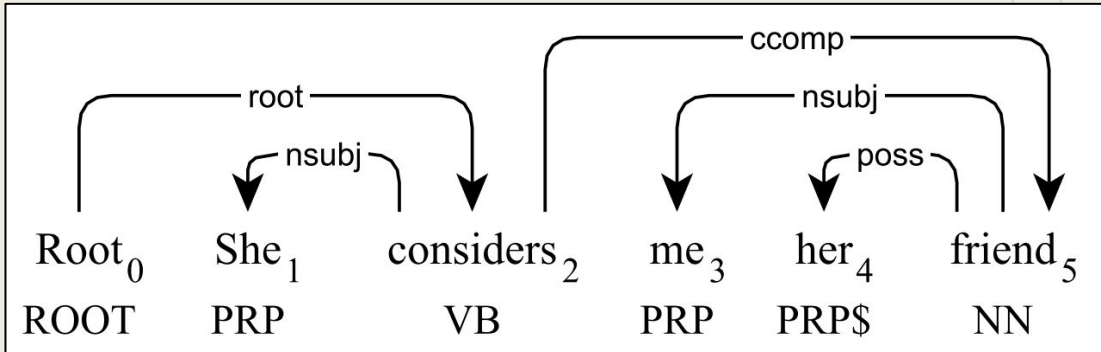
Resources: Treebanks

- Many text corpora parsed by humans
- Used for training automatic parsers

Treebank conversion

Trees can be automatically converted to save manual work





Dependency Parsing algorithm

Input: sentence (list of tokens)

Output: dependency tree

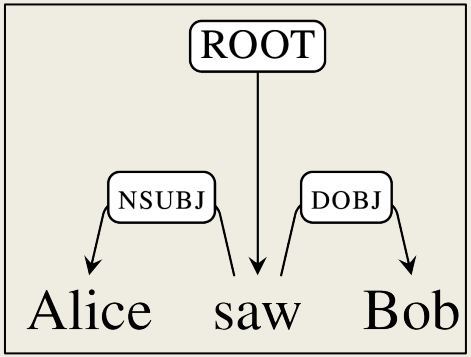
or simply, for each word, what is its head and arc label

Dependency Parsing algorithm

[alice, saw, bob] →

[(2, NSUBJ), (0, ROOT), (2, DOBJ)]

alice saw bob



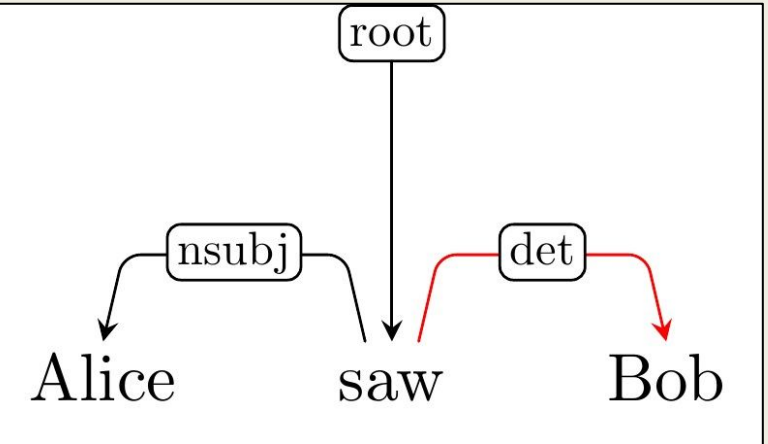
Use the index for each token.

The root node is denoted by 0.

Evaluation

We have a corpus to **train** the algorithm,

And another labeled corpus to **test** it.



What if it returned an incorrect tree?

Evaluation

Labeled Attachment Score (LAS):

% of words with correct head and label

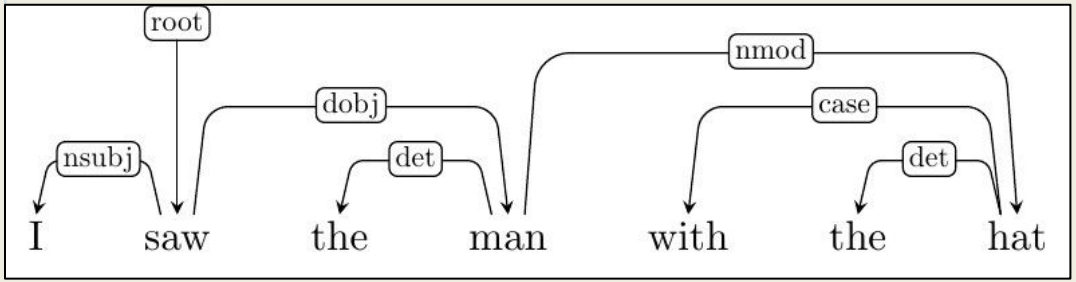
Unlabeled Attachment Score (UAS):

% of words with correct head

Always $0 \leq \text{LAS} \leq \text{UAS} \leq 100\%$

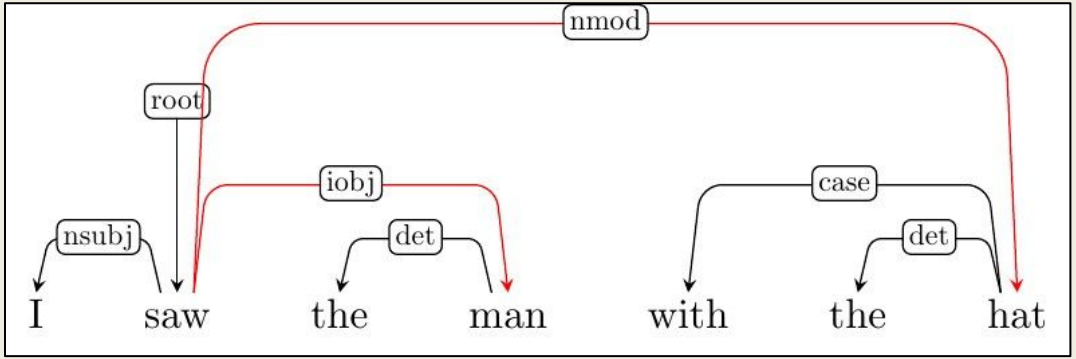
Evaluation example

Correct tree:



LAS = $^{5}/_{7}$ ≅ 71%

Evaluated tree:



UAS = $^{6}/_{7}$ ≅ 86%

"hat" has an incorrect head
"man" has a correct head but incorrect label

googleresearch.blogspot.co.il/2016/05/announcing-syntaxnet-worlds-most.html

Parser scores (English)

Parser	UAS (%)	LAS (%)
MaltParser	90.93	88.95
MSTParser	92.17	89.86
ZPar	92.93	91.28
TurboParser	93.80	92.00
Parsey McParseface	94.41	92.55

Transition-based parsing

Incremental parsing algorithms:

Build the tree one arc at a time.

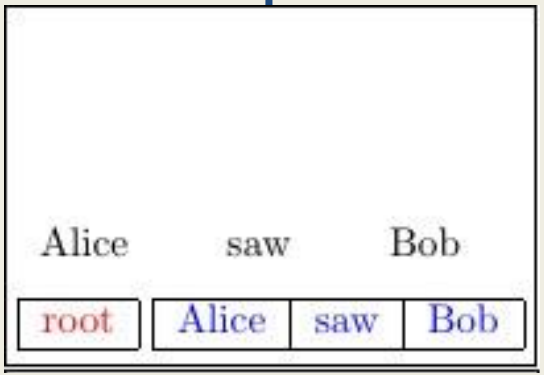
Apply **transitions** until the full tree is built.

Transition-based parsing

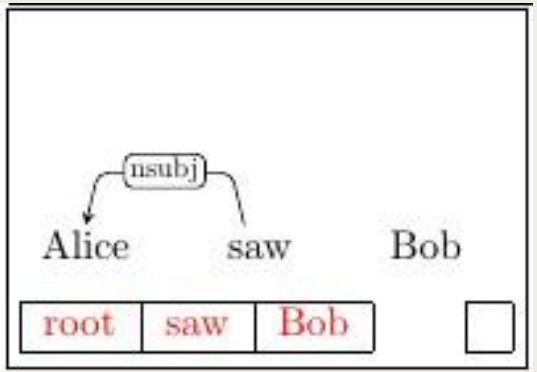
Using two lists: **stack** and **buffer**.

The stack keeps nodes being processed.

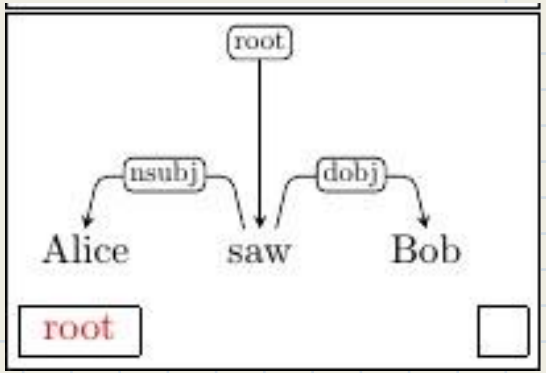
The input tokens are taken from the buffer.



...



...



Transition-based parsing

Possible transitions at each time step:

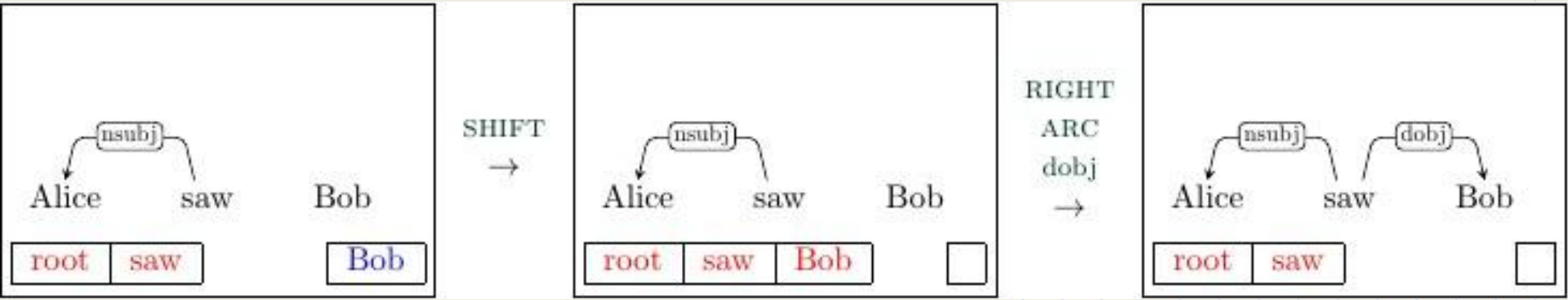
(Move node from buffer to stack)

SHIFT

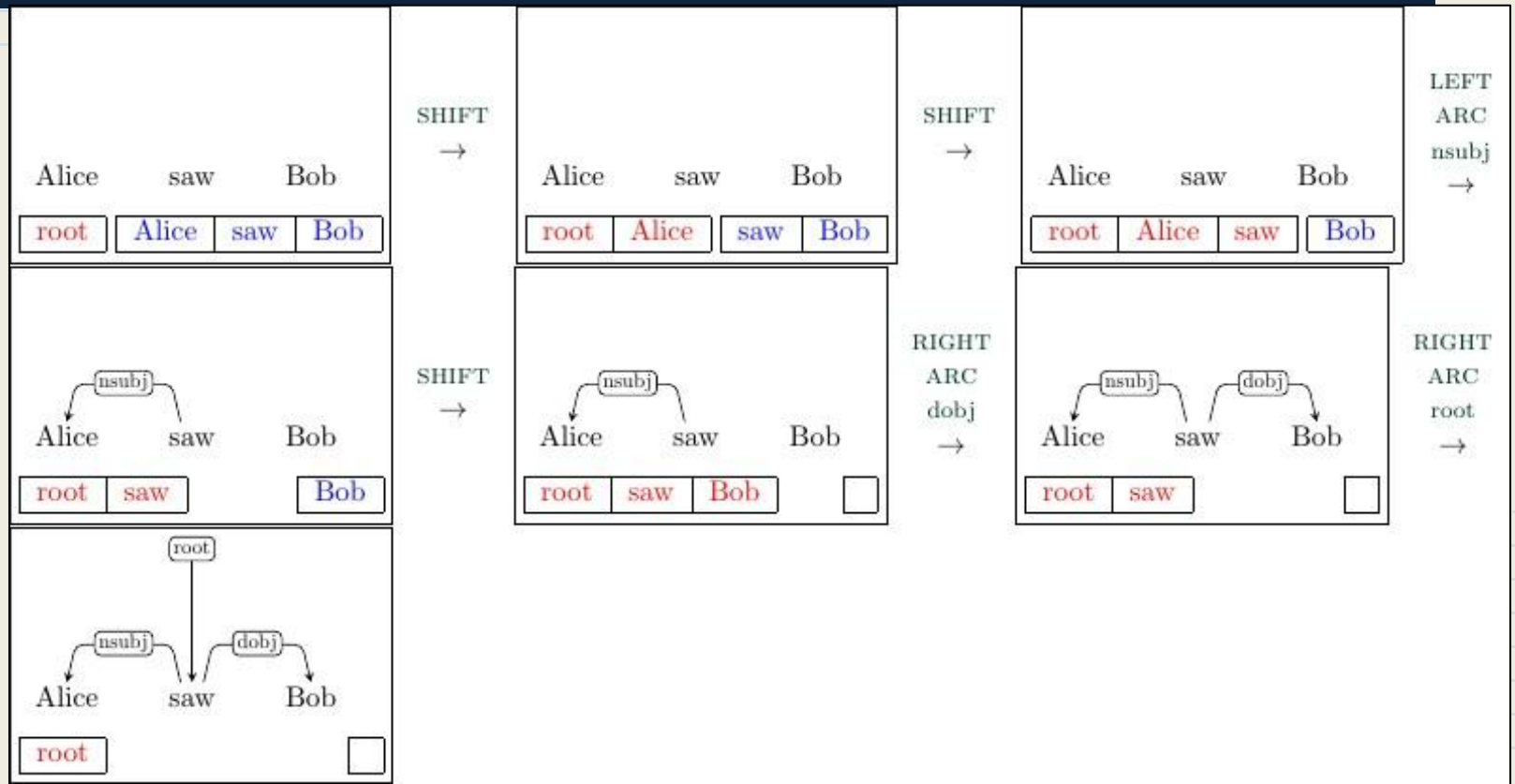
(Create left/right arc between two rightmost stack nodes, and delete child)

LEFT-ARC **RIGHT-ARC**

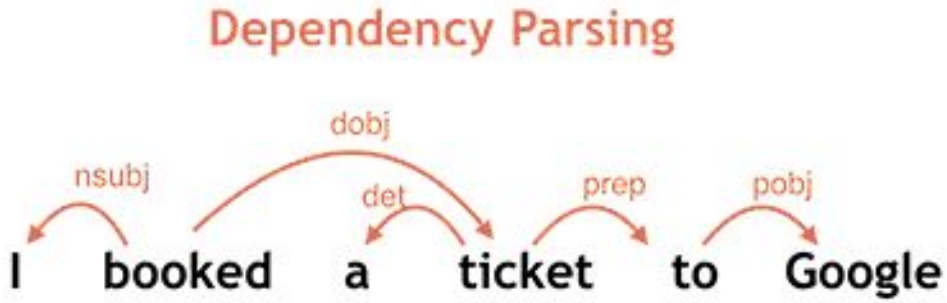
Need to say the label to create (e.g. dobj)



Transition-based parsing



Transition-based parsing



Transition-based parsing

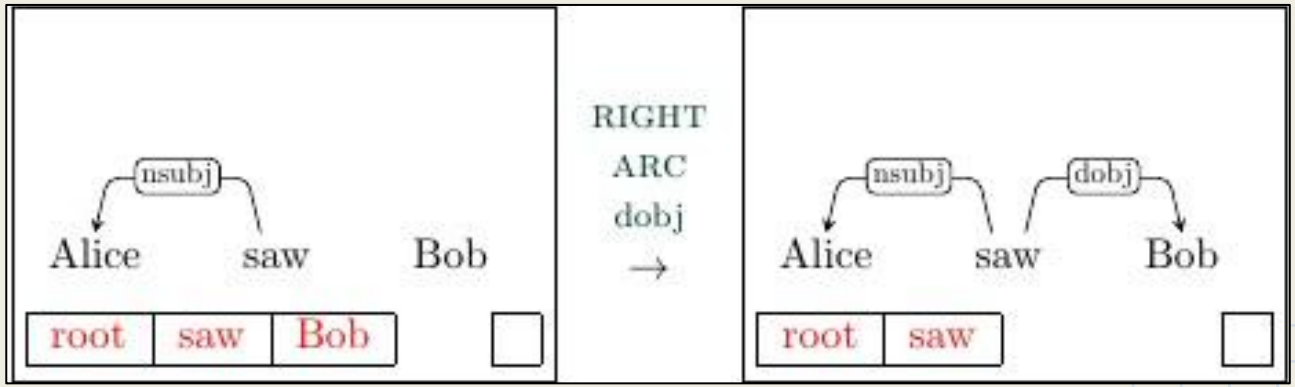
Where is the learning in the algorithm?

When we have a labeled tree, we know which transitions we need to get to it.

The parser learns how to make these decisions so it can parse new sentences correctly.

Transition-based parsing

If we see the state on the left here, we need to know to apply **RIGHT-ARC**$_{\textbf{dobj}}$



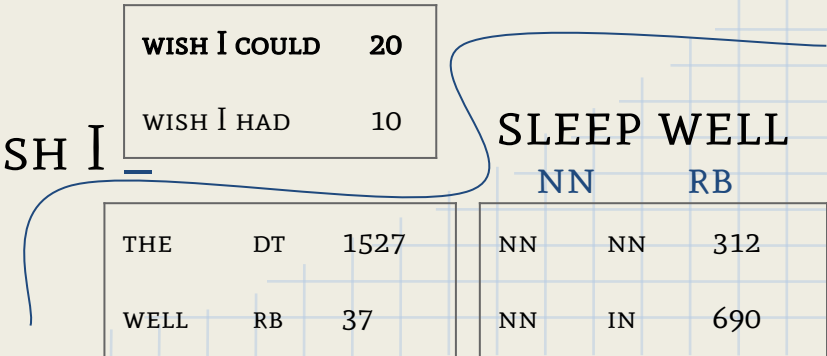
Machine learning

Learning: getting better in a task based on experience.

Examples we have seen in this course:

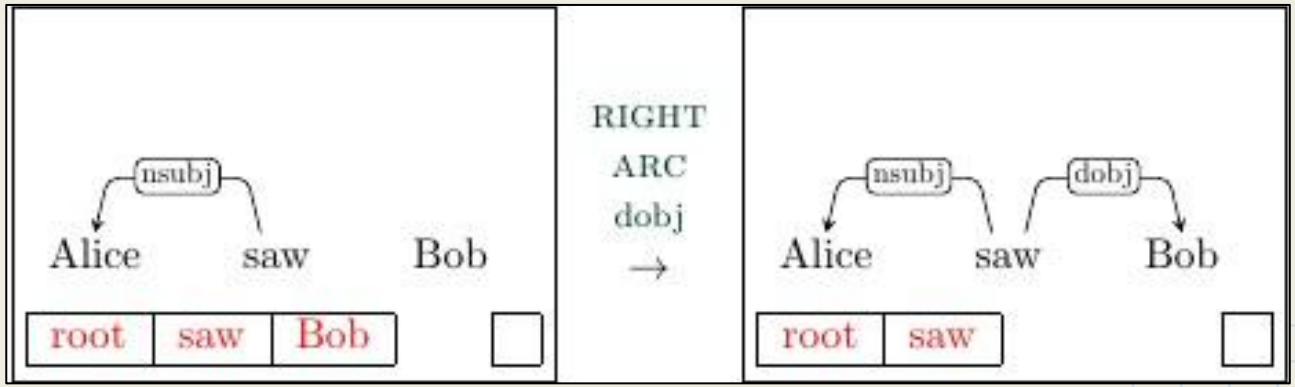
Language modeling,

Part-of-speech tagging



Machine learning

Count-based learning would not work well for transition-based parsing



Machine learning

Learning algorithms used for parsing:

- Perceptron
- Neural networks
- ...