Computational Syntax

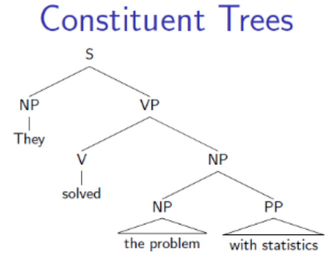
Koldo Gojenola. HAP/LAP.

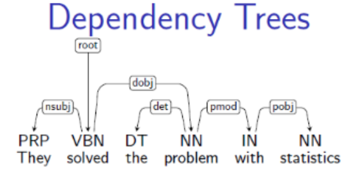
Bibliography

- Computational Approaches to Morphology and Syntax (Oxford Surveys in Syntax & Morphology), 2007 by Brian Roark, Richard Sproat
- Introduction to Natural Language Processing (Adaptive Computation and Machine Learning series, MIT Press). Jacob Eisenstein. 2019
  Notes (2018 version): https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf
- Michael Collins, slides:
  - Probabilistic Context-Free Grammars (PCFGs): http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf
  - Lexicalized Probabilistic Context-Free Grammars (PCFGs): http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/lexpcfgs.pdf
- Natural Language Processing with Python. NLTK Book. Chapter 8. Analyzing Sentence Structure. http://www.nltk.org/book/ch08.html
- Dependency Parsing (Synthesis Lectures on Human Language Technologies), 2009 by Sandra Kubler, Ryan McDonald, Joakim Nivre. Morgan & Claypool Publishers

Index

Two approaches to syntax





Why Context-Free Grammar?

- Regular languages *can't count*
- Example: $a^n b^n$ $(with\ n > 0)$ can not be generated by a regular grammar
- But a simple CFG grammar can
- There are syntactic constructions that have a similar pattern, like center embedding:

  the dog
  the cat the dog chased
  the goat the cat the dog chased kissed

  It corresponds to the pattern $noun^n verb^{n-1}$

Context-Free Grammar (CFG)

- S $\rightarrow$ NP VP
- VP $\rightarrow$ Verb NP
- NP $\rightarrow$ PropN
- NP $\rightarrow$ Pron

Context-Free Grammar (CFG)

- S $\rightarrow$ NP VP
- VP $\rightarrow$ Verb NP
- NP $\rightarrow$ PropN
- NP $\rightarrow$ Pron

General form:

- Rules have the form: a $\rightarrow$ b c d
  where a is a non-terminal symbol and b c d are terminal symbols

Context-Free Grammar (CFG)

- S $\rightarrow$ NP VP
- VP $\rightarrow$ Verb NP
- NP $\rightarrow$ PropN
- NP $\rightarrow$ Pron

General form:

- Rules have the form: a $\rightarrow$ b c d
  where a is a non-terminal symbol and b c d are terminal symbols
- Starting from the axiom (S) and applying the rules we can generate the strings of the language

Context-Free Grammar (CFG)

- S $\rightarrow$ NP VP
- VP $\rightarrow$ Verb NP
- NP $\rightarrow$ PropN
- NP $\rightarrow$ Pron

General form:

- Rules have the form: a $\rightarrow$ b c d
  where a is a non-terminal symbol and b c d are terminal symbols
- Starting from the axiom (S) and applying the rules we can generate the strings of the language
- The sequences generated by the grammar are represented by a tree

Context-Free Grammar (CFG)

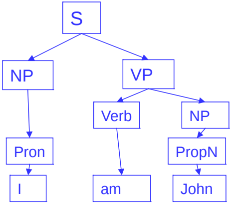
- S $\rightarrow$ NP VP
- VP $\rightarrow$ Verb NP
- NP $\rightarrow$ PropN
- NP $\rightarrow$ Pron

Context-Free Grammar (CFG)

- S $\rightarrow$ NP VP
- VP $\rightarrow$ Verb NP
- NP $\rightarrow$ PropN
- NP $\rightarrow$ Pron

Lexical rules

- Pron $\rightarrow$ I
- Verb $\rightarrow$ am
- PropN $\rightarrow$ John



S
NP
VP
Verb
NP
Pron
PropN
I
am
John

General architecture

General architecture

Sentence

↓

Morphological analysis and disambiguation

↓

Syntactic analysis

↓

Syntactic constituents (tree)

Main Syntactic Categories

- Sentence
- Noun Phrase
- Verb Phrase
- Other
  - Prepositional Phrases
  - Adverbial Phrases
  - Adjectival Phrases

Sentence

- Basic Rule: S → NP VP
- Other:
  - S → ADVP NP VP
    Unfortunately Abigail ate the kimchi.
  - S → S CC S
    Abigail ate the kimchi and Max had a burger.
  - S → VP
    Eat the kimchi.

Noun Phrase

- NP → NN | NNS | NNP | PRP
  singular, plural, and proper nouns; PRP: personal pronouns
- NP → DET NN | DET NNS | DET NNP | PRP
- NP → NN NN | NN NNS | DET NN NN | ...
- Recursive NP Phrases:
  - NP → NP CC NP
    the red and the black
  - NP → NP PP
    the President of the Georgia Institute of Technology
  - NP → NP SBAR
    a whale which he had wounded
  - NP → NP VP
    a whale taken near Shetland

Verb Phrase

- VP → VB | VBZ | VBD | VBN | VBG | VBP
  base form (VB: she likes to snack), present-tense third-person singular (VBZ: she snacks), present tense but not third-person singular (VBP: they snack), past tense (VBD: they snacked), present participle (VBG: they are snacking), and past participle (VBN: they had snacked)
- Recursive VP Phrases:
  - VP → MD VP
    She **will snack**
  - VP → VBD VP
    She **had snacked**
  - VP → VBZ VP
    She **has been snacking**
  - VP → VBN VP
    She has **been snacking**
  - VP → TO VP
    She wants **to snack**
  - VP → VP CC VP
    She **buys and eats many snacks**

Verb Phrase (continued):

- Verb complements:
  - VP → VBZ NP
    She **teaches algebra**
  - VP → VBG NP
    She has been **teaching algebra**
  - VP → VBD NP NP
    She **taught her brother algebra**
  - VP → VBZ S
    Hunter **wants to eat the kimchi**
  - VP → VBZ SBAR
    Hunter **knows that Tristan ate the kimchi**
- Prepositional and Adverbial Phrases:
  - VP → VBZ PP
    She **studies at night**
  - VP → VBZ ADVP
    She **studies intensively**
  - VP → ADVP VBG
    She is **not studying**

Verb Phrase (continued):

- Copula:
  - VP $\rightarrow$ VBZ ADJP
    She **is hungry**
  - VP $\rightarrow$ VBP ADJP
    Success **seems increasingly unlikely**

Other constituents:

- Prepositional Phrases:
  - PP → IN NP
    the whiteness **of the whale**
  - PP → TO NP
    What the white whale was **to Ahab, has been hinted**
- Complement Clauses:
  - SBAR → IN S
    She said **that it was spicy**
  - SBAR → S
    She said **it was spicy**
- Adverbial Clauses:
  - ADVP → RB RBR
    They went **considerably further**
  - ADVP → ADVP PP
    They went **considerably further than before**

Other constituents:

- Adjectival Clauses:
  - ADJP → RB JJ
    very hungry
  - ADJP → RBR JJ
    more hungry
  - ADJP → JJS JJ
    best possible
  - ADJP → RB JJR
    even bigger
  - ADJP → JJ CC JJ
    high and mighty
  - ADJP → JJ JJ
    West German
  - ADJP → RB VBN
    previously reported
- Coordination:
  - PP → PP CC PP
    on time and under budget
  - ADVP → ADVP CC ADVP
    now and two years ago
  - ADJP → ADJP CC ADJP
    quaint and rather deceptive
  - SBAR → SBAR CC SBAR
    whether they want control or whether they want exports

Ambiguity:

- PP attachment: I saw the man on the hill with a statue
- Coordination: The man took the hammer and saw
- Modifier scope: plastic bag container

Exercise I: produce a parse tree for these sentences using the rules that have been presented:

- This aggression will not stand.
- I can get you a toe.
- Sometimes you eat the bar and sometimes the bar eats you.

Exercise II: write a grammar to capture the following agreement in Spanish:

- La casa bonita
- El perro bonito

Specific domains: semantic grammars

- Intervention → question | order | ...
- order → v $\{\mathit{imperative}(1), \mathit{order}(1)\}$
- np → baseNp |
- np → baseNp npMod $\{\mathit{agreement}(1, 2)\}$
- baseNp → n |
- baseNp → det adj n $\{\mathit{agreement}(1, 2, 3)\}$
- npMod → pp | ...
- pp → prep np
- np → "barcelona" | "valencia" | ...
- n → “ticket” | "euromed" ...
- v → “give” | ...
- det → “a” | “the” | ...

Context-free Grammars

- NLTK exercise (open the CFG notebook in egela)
- Try different sentences with a basic grammar
- Extend the grammar

Index

Parsing algorithms

- Top-down parsing
- Shift-reduce parsing (bottom-up)
- Chart-based algorithms

Parsing algorithms: Recursive descent parsing

- Top-down parsing
- Idea: try to apply rules starting from the top symbol
- If a rule can not be applied, then try the next rule
- Until all the sentence is covered or there are no more rules
- Problem: a lot of work can be repeated

Recursive descent demo (NLTK)

- python (from the command line)
- >>> import nltk
- >>> nltk.app.rdparser()

Parsing algorithms: shift-reduce parsing

- Bottom-up parsing
- Idea: two main structures: stack (of analyzed elements) and input sequence
- The elements will be shifted onto the stack until a right-hand side of a rule is formed, and then it is replaced by the left-hand side of the rule
- Until the sentence has been analyzed or no rule can be applied
- Problem: the process can go to a dead end, even when there is one analysis (improvement: backtracking)

Shift-reduce parsing demo (NLTK)

- python (from the command line)
- >>> import nltk
- >>> nltk.app.srparser()

Parsing algorithms: chart parsing

- Idea: store the obtained analysis in a table, so that no analysis will be repeated
- Many alternatives and algorithms: top-down, bottom-up and hybrid

Chart parsing demo (NLTK)

- python (from the command line)
- `>>> import nltk`
- `>>> nltk.app.chartparser()`

The CKY algorithm

- Grammar in Chomsky Normal Form
- Chart parsing, bottom-up dynamic programming algorithm
- Main idea:
  - Start finding the smallest elements (length 1)
  - Then continue finding elements of length 2, 3, ...
  - Repeat until finding elements of length M (length of the sentence)
- Time: $O(M^3\ N)$, where M = length of the sentence; N = number of grammar rules

CKY algorithm: Grammar in Chomsky Normal Form (CNF)

- Grammar equivalence
- A single CF Language can be expressed by more than one CF Grammar
- Two grammars are **weakly equivalent** if they generate the same strings
- Two grammars are **strongly equivalent** if they generate the same strings via the same derivations
- For example:
  - $S \rightarrow aSb \mid ab$
  - $S \rightarrow aSb \mid aabb \mid ab$

Grammar in Chomsky Normal Form (CNF)

- The right-hand side of every production includes either
  - two nonterminals, e.g. $A \rightarrow B\ C$, or
  - a single terminal symbol, e.g. $A \rightarrow a$

Grammar Transformation (CNF)

- Any CFG can be converted to a CNF grammar
- For example: $W \rightarrow X\ Y\ Z$
- Can be replaced by two productions:
  - $W \rightarrow X\ W\backslash X$
  - $W\backslash X \rightarrow Y\ Z$ |

Grammar Transformation (CNF)

- Any CFG can be converted to a CNF grammar
- For example: W → X Y Z
- Can be replaced by two productions:
  - W → X W\X
  - W\X → Y Z |

Exercise1: convert the following grammar to CNF:

- S → a S b | a b

Exercise2: convert the following grammar to CNF:

- NP → Det ADJ N | NP CORD NP
- Det → a | the
- N → dog | cat
- ADJ → big | ADJ big
- CORD → and

Example. A grammar in CNF:

- Sentence $\rightarrow$ NP VP
- NP $\rightarrow$ A B
- VP $\rightarrow$ C NP
- A $\rightarrow$ det
- B $\rightarrow$ n
- NP $\rightarrow$ n
- VP $\rightarrow$ vi
- C $\rightarrow$ vt

Example of the CKY algorithm: initial state

0	the	1	cat	2	eats	3	fish	4
	det		n		vi, vt		n	

Sentence → NP VP
NP → A B
VP → C NP
A → det
B → n
VP → vi
NP → n
C → vt

Example of the CKY algorithm. Initialization: elements of length 1

0 **the** 1 **cat** 2 **eats** 3 **fish** 4
det n vi, vt n

the (det) (0,1) **A**			
	cat (n) (1,2) **B**, **NP**		
		eats (vt, vi) (2,3) **VP**, **C**	
			fish (n) (3,4) **B**, **NP**

Sentence → NP VP
NP → A B
VP → C NP
A → det
B → n
VP → vi
NP → n
C → vt

Example of the CKY algorithm. Find elements of length 2

0	the (det)	1	cat (n)	2	eats (vi, vt)	3	fish (n)	4

the (det) (0, 1) **A**	**the cat** (0,2) **NP**		
	cat (n) (1,2) **B**, **NP**	**cat eats** (1,3) **Sentence**	
		eats (vt, vi) (2,3) **VP**, **C**	**eats fish** (2,4) **VP**
			fish (n) (3,4) **B**, **NP**

Sentence → NP VP
NP → A B
VP → C NP
A → det
B → n
VP → vi
NP → n
C → vt

Example of the CKY algorithm. Find elements of length 3

0	the (det)	1	cat (n)	2	eats (vi, vt)	3	fish (n)	4

the (det) (0, 1) **A**	**the cat** (0,2) **NP**	**the cat eats** (0,3) **Sentence**	
	cat (n) (1,2) **B**, **NP**	**cat eats** (1,3) **Sentence**	**cat eats fish** (1,4) **Sentence**
		eats (vt, vi) (2,3) **VP**, **C**	**eats fish** (2,4) **VP**
			fish (n) (3,4) **B**, **NP**

- Sentence → NP VP
- NP → A B
- VP → C NP
- A → det
- B → n
- VP → vi
- NP → n
- C → vt

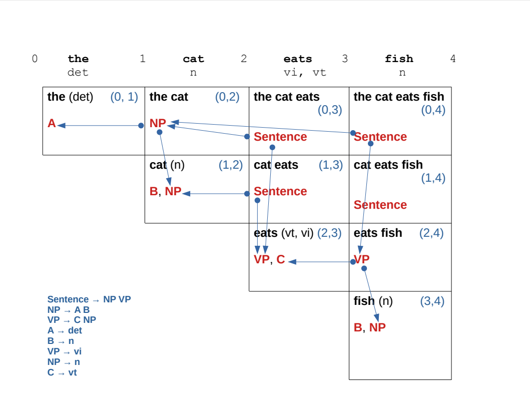
Example of the CKY algorithm. Find elements of length 4

0 **the** 1 **cat** 2 **eats** 3 **fish** 4
det n vi, vt n

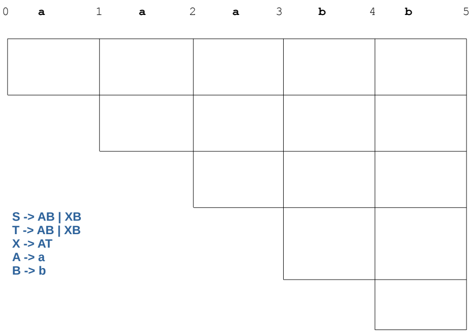
the (det) (0, 1) **A**	**the cat** (0,2) **NP**	**the cat eats** (0,3) **Sentence**	**the cat eats fish** (0,4) **Sentence**
	cat (n) (1,2) **B**, **NP**	**cat eats** (1,3) **Sentence**	**cat eats fish** (1,4) **Sentence**
		eats (vt, vi) (2,3) **VP**, **C**	**eats fish** (2,4) **VP**
			fish (n) (3,4) **B**, **NP**

Sentence → NP VP
NP → A B
VP → C NP
A → det
B → n
VP → vi
NP → n
C → vt

Example of the CKY algorithm. How to recover the tree: backpointers

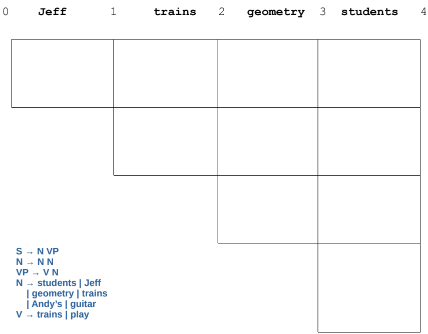


Example of the CKY algorithm.



S -> AB | XB
T -> AB | XB
X -> AT
A -> a
B -> b

Example of the CKY algorithm



S → N VP
N → N N
VP → V N
N → students | Jeff
| geometry | trains
| Andy's | guitar
V → trains | play

Example of the CKY algorithm.

0 **Andy's** 1 **students** 2 **play** 3 **guitar** 4

S → N VP
N → N N
VP → V N
N → students | Jeff
 | geometry | trains
 | Andy's | guitar
V → trains | play

The CKY algorithm (taken from Eisenstein 2019)

Algorithm 13 The CKY algorithm for parsing a sequence $\boldsymbol{w} \in \Sigma^*$ in a context-free grammar $G = (N, \Sigma, R, S)$, with non-terminals N, production rules R, and start symbol S. The grammar is assumed to be in Chomsky normal form (section 9.2.1). The function PICKFROM($b[i, j, X]$) selects an element of the set $b[i, j, X]$ arbitrarily. All values of t and b are initialized to $\varnothing$.

procedure CKY($\boldsymbol{w}, G = (N, \Sigma, R, S)$)
    for $m \in \{1 \dots M\}$ **do**
        $t[m-1, m] \leftarrow \{X : (X \rightarrow w_m) \in R\}$
    for $\ell \in \{2, 3, \dots, M\}$ **do** ▷ Iterate over constituent lengths
        for $m \in \{0, 1, \dots M - \ell\}$ **do** ▷ Iterate over left endpoints
            for $k \in \{m+1, m+2, \dots, m+\ell-1\}$ **do** ▷ Iterate over split points
                for $(X \rightarrow Y\ Z) \in R$ **do** ▷ Iterate over rules
                    if $Y \in t[m, k] \wedge Z \in t[k, m+\ell]$ **then**
                        $t[m, m+\ell] \leftarrow t[m, m+\ell] \cup X$ ▷ Add non-terminal to table
                        $b[m, m+\ell, X] \leftarrow b[m, m+\ell, X] \cup (Y, Z, k)$ ▷ Add back-pointers
    if S $\in t[0, M]$ **then**
        return TRACEBACK($S, 0, M, b$)
    else
        return $\varnothing$
procedure TRACEBACK(X, i, j, b)
    if $j = i + 1$ **then**
        return X
    else
        $(Y, Z, k) \leftarrow$ PICKFROM($b[i, j, X]$)
        return $X \rightarrow$ (TRACEBACK(Y, i, k, b), TRACEBACK(Z, k, j, b))

Implementation details for CKY. Every item in the chart must indicate:

- X $\rightarrow \alpha$ (i,j,k)
- X spans $w_{i+1,j}$
- For binary rules, k marks the split point $i < k < j$
- For example, if $\alpha = Y\ Z$, then Y spans $w_{i+1,k}$ and Z spans $w_{k+1,j}$
- Another table (or the same one) can store the backpointers to Y and Z

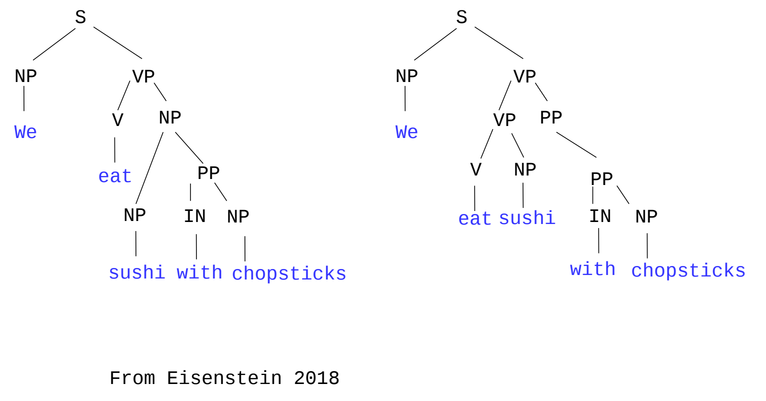
Example: CKY demo

http://sujeet.me/CYK/parser.html

Evaluating parsers:

- Precision: the fraction of constituents in the system parse that match a constituent in the reference parse.
- Recall: the fraction of constituents in the reference parse that match a constituent in the system parse.

Evaluating parsers



Evaluate precision and recall (left tree: system tree, right tree: gold/reference tree)

Weighted Context-free Grammars (WCFG)

- With a real language grammar, typically the parser can obtain hundreds or thousands of syntactic trees for a sentence

Weighted Context-free Grammars (WCFG)

- With a real language grammar, typically the parser can obtain hundreds or thousands of syntactic trees for a sentence
- How to select the best one?

Weighted Context-free Grammars (WCFG)

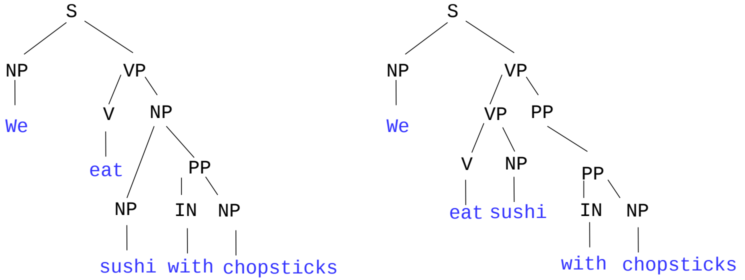
- With a real language grammar, typically the parser can obtain hundreds or thousands of syntactic trees for a sentence
- How to select the best one?
- Use weights calculated somehow (treebank, corpora, ...)

Example of Weighted Context-Free Grammar (WCFG) Eisenstein 2019

		$\psi(\cdot)$	$\exp\psi(\cdot)$
S	$\rightarrow$ NP VP	0	1
NP	$\rightarrow$ NP PP	-1	$\frac{1}{2}$
	$\rightarrow$ *we*	-2	$\frac{1}{4}$
	$\rightarrow$ *sushi*	-3	$\frac{1}{8}$
	$\rightarrow$ *chopsticks*	-3	$\frac{1}{8}$
PP	$\rightarrow$ IN NP	0	1
IN	$\rightarrow$ *with*	0	1
VP	$\rightarrow$ V NP	-1	$\frac{1}{2}$
	$\rightarrow$ VP PP	-2	$\frac{1}{4}$
	$\rightarrow$ MD V	-2	$\frac{1}{4}$
V	$\rightarrow$ *eat*	0	1

Table 10.2: An example weighted context-free grammar (WCFG). The weights are chosen so that $\exp\psi(\cdot)$ sums to one over right-hand sides for each non-terminal; this is required by probabilistic context-free grammars, but not by WCFGs in general.

Weighted Context-Free Grammar (WCFG)



From Eisenstein 2018

- Scoring function of a tree: sum of item scores
- Given the two trees, calculate the score of each one

Parsing with Weighted Context-Free Grammar (WCFG)

- For each item in the chart: X $\rightarrow$ Y Z (i,j,k)
  we must keep the *score of the best derivation of X spanning* $w_{i+1,j}$
- We will compute each the score of each element X as the maximum of:
  - Given X $\rightarrow$ Y Z (i,j,k)
  - The score of the production X $\rightarrow$ Y Z plus
  - The score of the best derivation for Y:$w_{i+1,k}$ plus
  - The score of the best derivation for Z:$w_{k+1,k}$
- The scores will be combined by addition.
- The score (ψ) of a tree formed by rules ($\alpha_1 \rightarrow \beta_1, \ldots \alpha_N \rightarrow \beta_N$):
  $\psi(t) = \sum_{i=1}^{N} \psi(\alpha_i \rightarrow \beta_i)$

The CKY algorithm with a WCFG (Eisenstein 2019)

Algorithm 14 CKY algorithm for parsing a string $\boldsymbol{w} \in \Sigma^*$ in a weighted context-free grammar (N, Σ, R, S), where N is the set of non-terminals and R is the set of weighted productions. The grammar is assumed to be in Chomsky normal form (section 9.2.1). The function TRACEBACK is defined in Algorithm 13.

procedure WCKY($\boldsymbol{w}, G = (N, \Sigma, R, S)$)
  for all i, j, X **do** ▷ Initialization
    $t[i, j, X] \leftarrow 0$
    $b[i, j, X] \leftarrow \varnothing$
  for $m \in \{1, 2, \ldots, M\}$ **do**
    for all $X \in N$ **do**
      $t[m, m+1, X] \leftarrow \psi(X \rightarrow w_m, (m, m+1, m))$
  for $\ell \in \{2, 3, \ldots M\}$ **do**
    for $m \in \{0, 1, \ldots, M - \ell\}$ **do**
      for $k \in \{m+1, m+2, \ldots, m+\ell-1\}$ **do**
        $t[m, m+\ell, X] \leftarrow \max_{k,Y,Z} \psi(X \rightarrow Y\ Z, (m, m+\ell, k)) + t[m, k, Y] + t[k, m+\ell, Z]$
        $b[m, m+\ell, X] \leftarrow \operatorname{argmax}_{k,Y,Z} \psi(X \rightarrow Y\ Z, (m+\ell, k)) + t[m, k, Y] + t[k, m+\ell, Z]$
  return TRACEBACK($S, 0, M, b$)

Applying CKY to WCFG

0 **We** 1 **eat** 2 **sushi** 3 **with** 4 **chopsticks** 5

Probabilistic Context-free Grammars (PCFG): Special case of WCFG

- The weights are probabilities

Probabilistic Context-free Grammars (PCFG): Special case of WCFG

- The weights are probabilities
- Advantage: easier interpretation

Probabilistic Context-free Grammars (PCFG): Special case of WCFG

- The weights are probabilities
- Advantage: easier interpretation
- They must obey the constraints on probabilities (sum to 1, ...)

Probabilistic Context-free Grammars (PCFG): Special case of WCFG

- The weights are probabilities
- Advantage: easier interpretation
- They must obey the constraints on probabilities (sum to 1, ...)
- Parsing: apply multiplication of probabilities

Probabilistic Context-free Grammars (PCFG)

- $p(t) \geq 0, \forall t \in$ the set of trees given by a grammar G

Probabilistic Context-free Grammars (PCFG)

- $p(t) \geq 0, \forall t \in$ the set of trees given by a grammar G
- $\sum_{t \in T_G} p(t) = 1$

  where T_G is the set of trees generated by the grammar

How do we calculate the best parse tree of a sentence s?

- Calculate: $Optimal\ parse = argmax_{t\ \in\ T_G(s)}\ p(t)$

How do we calculate the best parse tree of a sentence s?

- Calculate: $Optimal\ parse = argmax_{t\ \in\ T_G(s)}\ p(t)$
- Probability of a tree formed by rules $(\alpha_1 \rightarrow \beta_1, \ldots \alpha_N \rightarrow \beta_N)$:
  $$p(t) = \prod_{i=1}^{N} p(\alpha_i \rightarrow \beta_i)$$
- Given a sentence, calculate all the possible trees
- The result is the tree with the maximum probability

Questions

- How do we calculate p(t)?

Questions

- How do we calculate p(t)?
- Learning: how do we calculate the parameters from training examples?

Questions

- How do we calculate p(t)?
- Learning: how do we calculate the parameters from training examples?
- Parsing: for a given sentence s, how do we find the most likely tree?

Example PCFG (M. Collins, http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf)

S	$\rightarrow$	NP	VP	1.0

Example PCFG (M. Collins, http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf)

S	→	NP	VP	1.0
VP	→	Vi		0.3
VP	→	Vt	NP	0.5
VP	→	VP	PP	0.2

Example PCFG (M. Collins, http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf)

S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vi		0.3
VP	$\rightarrow$	Vt	NP	0.5
VP	$\rightarrow$	VP	PP	0.2
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2

Example PCFG (M. Collins, http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf)

S	→	NP	VP	1.0
VP	→	Vi		0.3
VP	→	Vt	NP	0.5
VP	→	VP	PP	0.2
NP	→	DT	NN	0.8
NP	→	NP	PP	0.2
PP	→	IN	NP	1.0

Example PCFG (M. Collins, http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf)

S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vi		0.3
VP	$\rightarrow$	Vt	NP	0.5
VP	$\rightarrow$	VP	PP	0.2
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

Example PCFG

Vi	$\rightarrow$	sleeps	1.0

Example PCFG

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0

Example PCFG

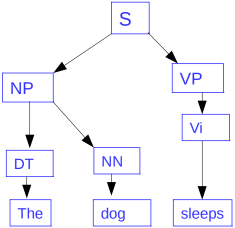
Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5

Example PCFG

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0

Example PCFG

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4



How do we calculate the probability of a tree?

$$
\begin{aligned}
p(t) = \; & q(S \rightarrow NP\ VP) \times \\
& q(NP \rightarrow DT\ NN) \times \\
& q(DT \rightarrow the) \times \\
& q(NN \rightarrow dog) \times \\
& q(VP \rightarrow Vi) \times \\
& q(Vi \rightarrow sleeps)
\end{aligned}
$$

Learning: how do we calculate the parameters from training examples?

- $$q_{ML}(\alpha \rightarrow \beta) = \frac{count(\alpha \rightarrow \beta)}{count(\alpha)}$$

Learning: how do we calculate the parameters from training examples?

- $$q_{ML}(\alpha \rightarrow \beta) = \frac{count(\alpha \rightarrow \beta)}{count(\alpha)}$$

Learning: how do we calculate the parameters from training examples?

- $$q_{ML}(\alpha \to \beta) = \frac{count(\alpha \to \beta)}{count(\alpha)}$$

- Example:
  - The rule $VP \to Vt\ NP$ is seen 105 times in a corpus
  - The non-terminal VP is seen 1000 times
  - Then
  $$q(VP \to Vt\ NP) = \frac{105}{1000}$$

PCFG: generative model: assumption that parse trees are generated stochastically

- Define $s_1 = S, i = 1$

PCFG: generative model: assumption that parse trees are generated stochastically

- Define $s_1 = S, i = 1$
- While s_i contains at least one non-terminal:
  - Find the left-most non-terminal in s_i, call this X
  - Choose one of the rules of the form $X \rightarrow \beta$ from the distribution q($X \rightarrow \beta$)
  - Create s_{i+1} by replacing the left-most X in s_i by β
  - Set i = i + 1

Exercise I, giving a treebank:

- (N (A long) (N (A red) (N hair)))
- (N (A nice) (N tie))
- (N (A (A dark) (A red)) (N hair))

Exercise I, giving a treebank:

- (N (A long) (N (A red) (N hair)))
- (N (A nice) (N tie))
- (N (A (A dark) (A red)) (N hair))

Calculate its corresponding PCFG:

$$\left| \begin{array}{lll} N & \rightarrow & A \quad N \\ & & \text{hair} \\ & & \text{tie} \end{array} \right.$$

Exercise I, giving a treebank:

- (N (A long) (N (A red) (N hair)))
- (N (A nice) (N tie))
- (N (A (A dark) (A red)) (N hair))

Calculate its corresponding PCFG:

N	→	A	N
		hair	
		tie	
A	→	A	A
		long	
		red	
		dark	
		nice	

Exercise I: Calculate the best tree for "nice red hair"

Exercise II. Given a PCFG:

- S $\rightarrow$ NP NP [1.0]
- NP $\rightarrow$ NP PP [0.2]
- NP $\rightarrow$ NP NP [0.2]
- PP $\rightarrow$ P NP [1.0]
- VP $\rightarrow$ V NP [0.7]
- VP $\rightarrow$ VP PP [0.3]
- P $\rightarrow$ with [1.0]
- V $\rightarrow$ saw [1.0]
- NP $\rightarrow$ astronomers [0.1]
- NP $\rightarrow$ ears [0.18]
- NP $\rightarrow$ saw [0.02]
- NP $\rightarrow$ stars [0.18]
- NP $\rightarrow$ telescopes [0.1]
- NP $\rightarrow$ astronomer's [0.02]

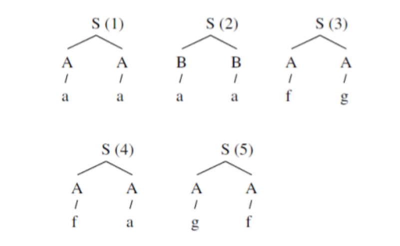
Calculate the trees corresponding to the sentence *astronomers saw stars with telescopes* and give the one with the highest probability.

Exercise III. Given a PCFG:

- S $\rightarrow$ V N [0.6]
- S $\rightarrow$ NP V [0.4]
- NP $\rightarrow$ D N [1.0]
- D $\rightarrow$ a [0.2]
- D $\rightarrow$ the [0.8]
- V $\rightarrow$ support [0.6]
- V $\rightarrow$ hate [0.4]
- N $\rightarrow$ president [1.0]

Calculate all the sentences with $p(x) > 0$, each with its corresponding probability.

Exercise IV. Given the following trees:



The first tree appeared 500 times in a corpus, the second one 250 times, the third one 333 times, the fourth one 789 times and the fifth one 12 times. Calculate the probabilities corresponding to the following rules:

- A → f
- A → g
- B → a
- S → B B

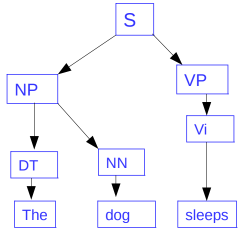
NLTK: Probabilistic Context-free Grammars

- Notebook in egela (PCFG)
- Apply the grammars

Some weaknesses of Probabilistic Context-free Grammars

- Lack of sensitivity to lexical information
- Lack of sensitivity to structural preferences

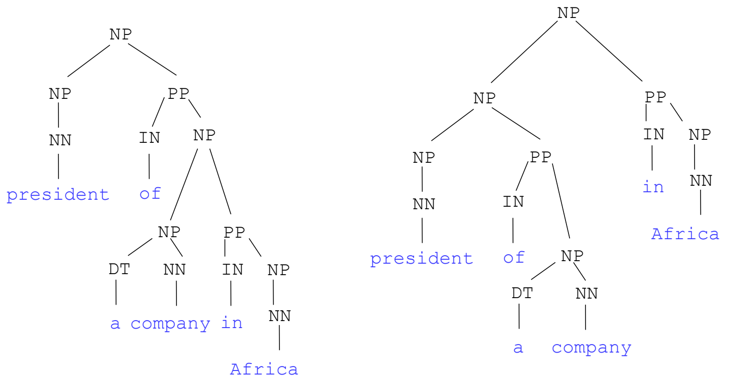
Lack of sensitivity to lexical information



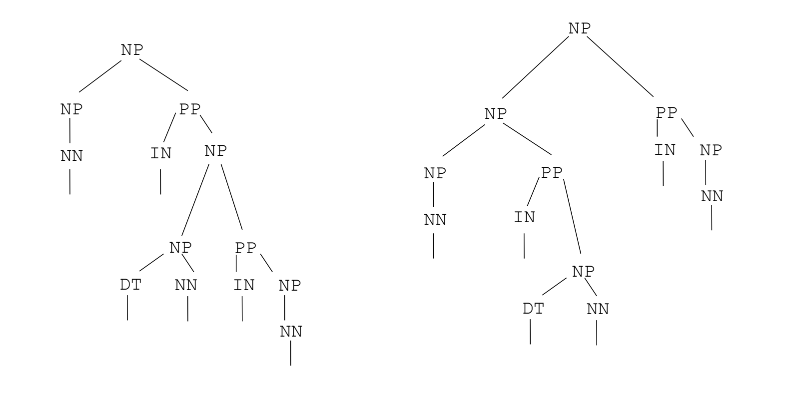
The word "dog" is only dependent on its tag *NN* and conditionally independent of the entire tree

Another example: PPs with into as the preposition are almost nine times more likely to attach to a VP rather than an NP

Lack of sensitivity to structural preferences



Lack of sensitivity to structural preferences



Both trees use the same rules and have equal probability
But the first structure is two times more frequent

Adding more context: parent annotation:

Example: PP attachment to NP:
More likely in object position: They amused the students from Georgia than in The students from Georgia were amused

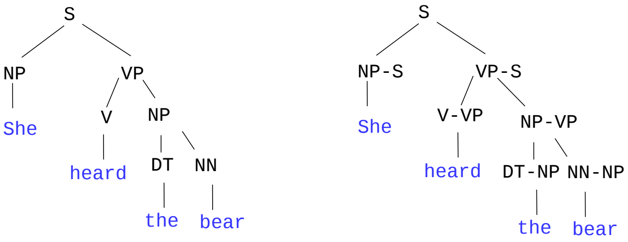
- $\Pr(\text{NP} \rightarrow \text{NP PP}) = 11\%$

Adding more context: parent annotation:

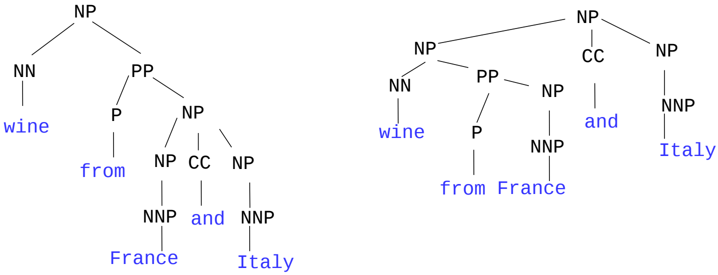
Example: PP attachment to NP:
More likely in object position: They amused the students from Georgia than in The students from Georgia were amused

- Pr(NP $\rightarrow$ NP PP) $=$ 11%
- Pr(NP under S $\rightarrow$ NP PP) $=$ 9%
- Pr(NP under VP $\rightarrow$ NP PP) $=$ 23%.y

Adding more context: parent annotation:

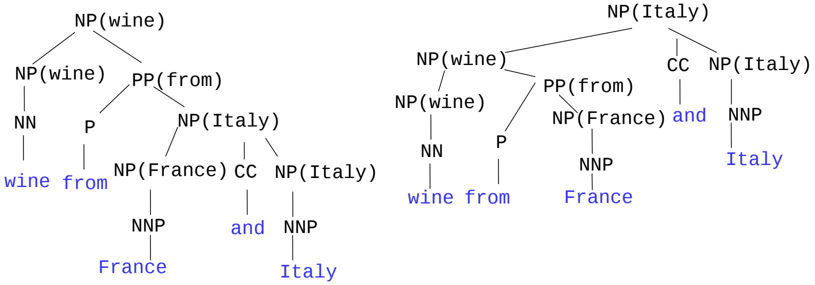


Adding more context: parent annotation?



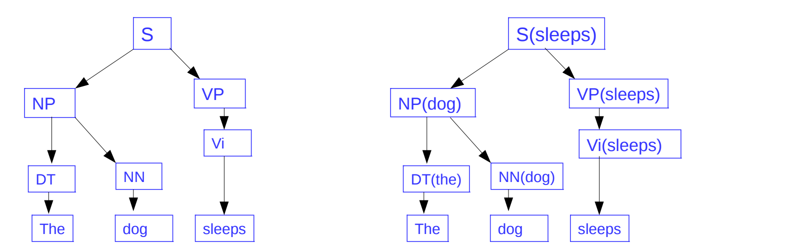
Example of ambiguity.
(from Eisenstein 2018)

Adding more context: lexicalization



Example of lexicalization.
(from Eisenstein 2018)

Lexicalization



Lexicalization

- Rules will be of the form S(sleeps) $\rightarrow$ NP(dog) VP(sleeps)

Lexicalization

- Rules will be of the form S(sleeps) $\rightarrow$ NP(dog) VP(sleeps)
- Many rules!

Lexicalization

- Rules will be of the form S(sleeps) $\rightarrow$ NP(dog) VP(sleeps)
- Many rules!
- Smoothing is necessary

Performance od PCFGs

A Comparison of PCFGs

Parser	F_1 Error
Plain PCFG (Charniak, 1996)	28.0%
Parent annotations (Johnson, 1999)	20.4%
Lexicalized PCFGs (Collins, 1999)	11.8%
Latent variables, EM (Petrov & Klein 2007)	9.9%

Beyond Context-free Parsing

- Reranking

Beyond Context-free Parsing

- Reranking
  - A context-free parser generates a k-best list of candidates

Beyond Context-free Parsing

- Reranking
  - A context-free parser generates a k-best list of candidates
  - The reranker selects the best parse
  - Arbitrary non-local features can be incorporated (e.g. NP(France) CC NP(Italy))
  - Can obtain substantial improvements in accuracy
- Transition-based parsing (shift-reduce parsing)

Beyond Context-free Parsing

- Reranking
  - A context-free parser generates a k-best list of candidates
  - The reranker selects the best parse
  - Arbitrary non-local features can be incorporated (e.g. NP(France) CC NP(Italy))
  - Can obtain substantial improvements in accuracy
- Transition-based parsing (shift-reduce parsing)
  - Two structures: stack and input
  - Two actions: shift and reduce
  - Very efficient
  - Error propagation when taking a bad decision
  - Example: analyze They eat sushi

Index

Problems with context-free grammars

- Agreement:
  - The man sleep
  - These house

Problems with context-free grammars

- Agreement:
  - The man sleep
  - These house
- How to examine agreement?
  - VP $\rightarrow$ NP_3S VP_3S he sleeps

Problems with context-free grammars

- Agreement:
  - The man sleep
  - These house
- How to examine agreement?
  - VP $\rightarrow$ NP_3S VP_3S he sleeps
  - VP $\rightarrow$ NP_3P VP_3P they sleep

Problems with context-free grammars

- Agreement:
  - The man sleep
  - These house
- How to examine agreement?
  - VP → NP_3S VP_3S he sleeps
  - VP → NP_3P VP_3P they sleep
  - NP → Det_3P N_3P these houses
  - ...

Problems with context-free grammars

- Agreement:
  - The man sleep
  - These house
- How to examine agreement?
  - VP → NP_3S VP_3S he sleeps
  - VP → NP_3P VP_3P they sleep
  - NP → Det_3P N_3P these houses
  - ...
- Many rules!

Problems with context-free grammars

- Agreement:
  - The man sleep
  - These house
- How to examine agreement?
  - VP → NP_3S VP_3S he sleeps
  - VP → NP_3P VP_3P they sleep
  - NP → Det_3P N_3P these houses
  - ...
- Many rules!
- Feature-structures: each syntactic constituent will have features

Unification-based Grammars are useful for treating several phenomena

- Agreement:
- Case control
- Subcategorization
- Long-distance dependencies
- Control
- Coordination

Unification-based Grammars. Different formalisms

- Lexical-Functional Grammar (LFG)
  Treebanks for several languages and parser demo:
  `http://clarino.uib.no/iness/xle-web`
- Head-Driven Phrase-Structure Grammar (HPSG)
  English demo: `http://erg.delph-in.net/logon`
- . . .

Unification-based grammars. Example (I)

$$\left| \begin{array}{llll} \text{S} & \rightarrow & \text{NP} & \text{VP} \\ \text{NP NUM} = \text{VP NUM} & & & \end{array} \right.$$

Unification-based grammars. Example (I)

S	$\rightarrow$	NP	VP
NP NUM = VP NUM			
NP	$\rightarrow$	N	
NP NUM = N NUM			

Unification-based grammars. Example (I)

S	$\rightarrow$	NP	VP
NP NUM = VP NUM			
NP	$\rightarrow$	N	
NP NUM = N NUM			
NP	$\rightarrow$	PropN	
NP NUM = PropN NUM			

Unification-based grammars. Example (I)

S	$\rightarrow$	NP	VP
NP NUM = VP NUM			
NP	$\rightarrow$	N	
NP NUM = N NUM			
NP	$\rightarrow$	PropN	
NP NUM = PropN NUM			
NP	$\rightarrow$	Det	N
Det NUM = N NUM			
NP NUM = N NUM			

Unification-based grammars. Example (I)

S	$\rightarrow$	NP	VP
NP NUM = VP NUM			
NP	$\rightarrow$	N	
NP NUM = N NUM			
NP	$\rightarrow$	PropN	
NP NUM = PropN NUM			
NP	$\rightarrow$	Det	N
Det NUM = N NUM			
NP NUM = N NUM			
VP	$\rightarrow$	IV	
VP TENSE = IV TENSE			
VP NUM = IV NUM			

Unification-based grammars. Example (I)

S	$\rightarrow$	NP	VP
NP NUM = VP NUM			
NP	$\rightarrow$	N	
NP NUM = N NUM			
NP	$\rightarrow$	PropN	
NP NUM = PropN NUM			
NP	$\rightarrow$	Det	N
Det NUM = N NUM			
NP NUM = N NUM			
VP	$\rightarrow$	IV	
VP TENSE = IV TENSE			
VP NUM = IV NUM			
VP	$\rightarrow$	TV	NP
VP TENSE = TV TENSE			
VP NUM = TV NUM			

Lexical Productions

this	every
form = this	form = every
type = Det	type = Det
NUM = sg	NUM = sg

Lexical Productions

this form = this type = Det NUM = sg	every form = every type = Det NUM = sg
these form = these type = Det NUM = pl	all form = all type = Det NUM = pl

Lexical Productions

this form = this type = Det NUM = sg	every form = every type = Det NUM = sg
these form = these type = Det NUM = pl	all form = all type = Det NUM = pl
Kim form = Kim type = PropN	Jody form = some type = PropN

Lexical Productions

this form = this type = Det NUM = sg	every form = every type = Det NUM = sg
these form = these type = Det NUM = pl	all form = all type = Det NUM = pl
Kim form = Kim type = PropN	Jody form = some type = PropN
dog form = dog type = N NUM = sg car form = car type = N NUM = sg	girl form = girl type = N NUM = sg child form = child type = N NUM = sg

Lexical Productions

dogs	girls
form = dogs	form = girls
type = N	type = N
NUM = pl	NUM = pl
cars	children
form = cars	form = children
type = N	type = N
NUM = pl	NUM = pl

Lexical Productions

dogs form = dogs type = N NUM = pl cars form = cars type = N NUM = pl	girls form = girls type = N NUM = pl children form = children type = N NUM = pl

disappears form = disappears type = IV NUM = sg TENSE = pres sees form = sees type = TV NUM = sg TENSE = pres	walks form = walks type = IV NUM = sg TENSE = pres likes form = likes type = TV NUM = sg TENSE = pres

Lexical Productions

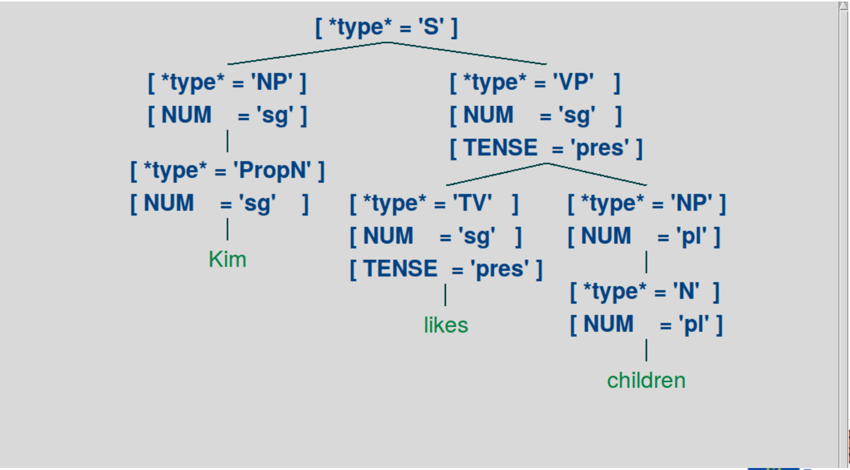
disappear form = disappear type = IV NUM = pl TENSE = pres	walk form = walk type = IV NUM = pl TENSE = pres
see form = see type = TV NUM = pl TENSE = pres	like form = like type = TV NUM = pl TENSE = pres

Lexical Productions

disappear form = disappear type = IV NUM = pl TENSE = pres see form = see type = TV NUM = pl TENSE = pres	walk form = walk type = IV NUM = pl TENSE = pres like form = like type = TV NUM = pl TENSE = pres
disappeared form = disappeared type = IV NUM = pl TENSE = past saw form = saw type = TV NUM = pl TENSE = past	walked form = walked type = IV NUM = pl TENSE = past liked form = liked type = TV NUM = pl TENSE = past

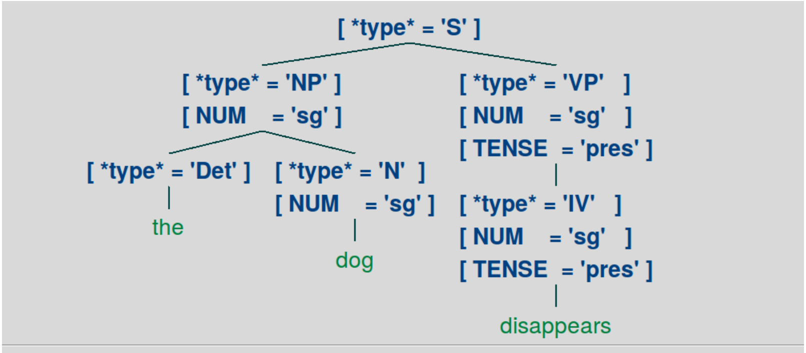
Analysis: **Kim likes children**

Analysis: **Kim likes children**



Analysis: **The dog disappears**

Analysis: **The dog disappears**



NLTK: Unification-based Grammars

- Notebook in egela
- Apply the grammars

Index

Dependency Parsing

TBC