Foundations for Natural Language Processing

Syntax and Parsing

Ivan Titov
(with slides from Ivan Titov, Alex Lascarides, Sharon Goldwater, Mark Steedman, and Marco Kuhlmann)

School of informatics

Today

- Last few words about sequence modeling
- Basics of Syntax, Context-Free Grammars
- Classes of Syntactic Parsing Algorithms
- Start with the CKY algorithm

Recap: Sequence labeling problems

- Definition:
  - Input: sequences of variable length $\mathbf{x} = (x_1, x_2, \ldots, x_{|x|}), x_i \in \mathcal{X}$
  - Output: every position is associated with a label $\mathbf{y} = (y_1, y_2, \ldots, y_{|x|}), y_i \in \{1, \ldots, N\}$
- An example:
  - Part-of-speech tagging

$\mathbf{x} =$	**John**	**carried**	**a**	**tin**	**can**	**.**
$\mathbf{y} =$	NNP	VBD	DT	NN	NN	.

Back to generative modeling – HMMs, what else?

- Using Viterbi, we can find the best tags for a sentence (decoding), and get $P(y, x|\theta)$ with HMM
  - or $P(\mathbf{y} \mid \mathbf{x})$ if you use the conditional modeling
- We might also want to
  - compute the likelihood, i.e., the probability of a sentence regardless of its tags (a language model!) $P(x|\theta)$
  - learn the best set of parameters $\hat{\theta}$ given only an unannotated corpus of sentences.

Computing the likelihood

- From the probability theory we know

$$P(x|\theta) = \sum_y P(x, y|\theta)$$

- But there are an exponential number of sequences *y*
- Again, by computing and storing partial results, we can solve efficiently.

Viterbi

Initialization: $v_j^1 = a_{START,j} b_{j,x^1}, \quad j = 1, \ldots, N;$

Recomputation: $v_j^t = \left(\max_i v_i^{t-1} a_{ij} \right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x|$

Final: $v_{STOP}^{|\mathbf{x}|+1} = \max_i v_i^{|\mathbf{x}|} a_{i,STOP}$

Forward algorithm

Initialization: $v_j^1 = a_{START,j} b_{j,x^1}, \quad j = 1, \ldots, N;$

Recomputation: $v_j^t = \left(\sum_i v_i^{t-1} a_{ij} \right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x|$

Final: $v_{STOP}^{|\mathbf{x}|+1} = \sum_i v_i^{|\mathbf{x}|} a_{i,STOP}$

Forward algorithm

Initialization: $v_j^1 = a_{START,j} b_{j,x^1}, \quad j = 1, \ldots, N;$

Recomputation: $v_j^t = \left(\sum_i v_i^{t-1} a_{ij} \right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x|$

Final: $v_{STOP}^{|\mathbf{x}|+1} = \sum_i v_i^{|\mathbf{x}|} a_{i,STOP}$

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{\exp\left(\sum_{t=1}^{|\mathbf{x}|} g^t(\mathbf{x}, y_{t-1}, y_t)\right)}{\sum_{\mathbf{y}'} \exp\left(\sum_{t=1}^{|\mathbf{x}|} g^t(\mathbf{x}, y'_{t-1}, y'_t)\right)}$$

This is also the algorithm used to compute the denominator for Markov Random Field, we considered last time

Today

- ~~Last few words about sequence modeling~~
- Basics of Syntax, Context-Free Grammars
- Classes of Syntactic Parsing Algorithms
- Start with the CKY algorithm

Modelling word behaviour

- We've seen various ways to model word behaviour.
  - Bag-of-words models: ignore word order entirely
  - N-gram models: capture a fixed-length history to predict word sequences.
  - HMMs: also capture fixed-length history, using latent variables.
  - RNNs/Transformers: few restrictions, reliance on flexibility neural networks
- We will see how provide linguistic priors into models

Long-range dependencies

- The form of one word often depends on (agrees with) another, even when arbitrarily long material intervenes.

  Sam/Dogs sleeps/sleep soundly
  Sam, who is my cousin, sleeps soundly
  Dogs often stay at my house and sleep soundly
  Sam, the man with red hair who is my cousin, sleeps soundly

- We want models that can capture these dependencies.

Phrasal categories

- We may also want to capture substitutability at the phrasal level.
  - POS categories indicate which words are *substitutable*. For example, substituting **adjectives**:

I saw a red cat
I saw a former cat
I saw a billowy cat

  - Phrasal categories indicate which phrases are substitutable. For example, substituting **noun phrase**:

Dogs sleep soundly
My next-door neighbours sleep soundly
Green ideas sleep soundly

This is one example of "**constituency test**"

Example constituency tests: coordination

- Only constituents (of the same type) can be **coordinated** using conjunction words like *and*, *or*, and *but*
- Pass the test:

Her friends from Peru went to the show.
Mary *and* her friends from Peru went to the show.

Should I go through the tunnel?
Should I go through the tunnel *and* over the bridge?

- Fail the test

We peeled the potatoes.
*We peeled the and washed the potatoes.

Example constituency tests: clefting

- Only a constituent can appear in the frame "____ *is/are who/what/where/when/why/how* …"
- Pass the test:

They put the boxes in the basement.
In the basement *is where* they put the boxes.

- Fail the test

They put the boxes in the basement.
*Put the boxes is what they did in the basement.

Theories of syntax

- A theory of syntax should explain which sentences are well-formed (grammatical) and which are not.
  - Note that well-formed is distinct from meaningful.
  - Famous example from Chomsky:

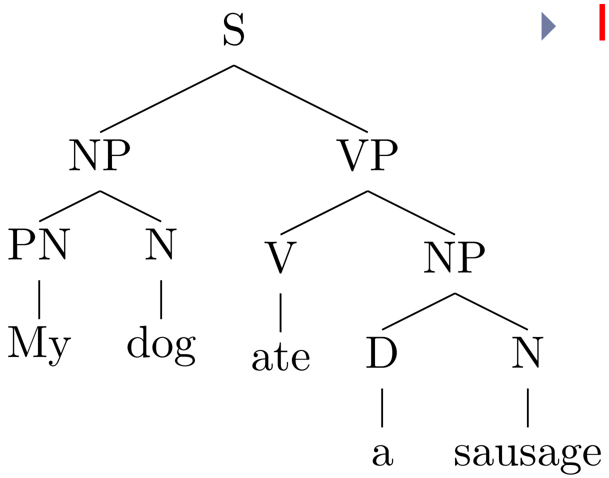
    Colorless green ideas sleep furiously
- (Even if the reason we care about syntax is mainly for interpreting meaning.)

Theories of syntax

- We'll look at one theory of syntax:
  - Constituency (aka phrase) structures

- A theory of syntax can be viewed as a model of language behaviour. As with other models, we will look at
  - What the model can capture, and what it cannot.
  - Algorithms that provide syntactic analyses for sentences using the model (i.e., syntactic parser).

Constituent trees



- Internal nodes correspond to phrases

  S – a sentence

  NP (Noun Phrase): My dog, a sandwich, lakes, ..

  VP (Verb Phrase): ate a sausage, barked, …

  PP (Prepositional phrases): with a friend, in a car, …

- Nodes immediately above words are PoS tags

  PN – pronoun
  D – determiner
  V – verb
  N – noun
  P – preposition

Context-Free Grammar

- Context-free grammar is a tuple of 4 elements $G = (V, \Sigma, R, S)$
  - V - the set of non-terminals
    In our case: phrase categories (VP, NP, ..) and PoS tags (N, V, .. – aka preterminals)
  - Σ - the set of terminals
    Words
  - R - the set of rules of the form $X \rightarrow Y_1, Y_2, \ldots, Y_n$, where $n \geq 0$, $X \in V,\ Y_i \in V \cup \Sigma$
    $$S \rightarrow NP\ \ VP$$
    $$NP \rightarrow D\ \ N$$
    $$NP \rightarrow PN$$
    $$NP \rightarrow NP\ \ PP$$
    $$PP \rightarrow P\ \ NP$$
    $$\ldots$$
    $$N \rightarrow girl$$
    $$N \rightarrow telescope$$
    $$V \rightarrow saw$$
    $$V \rightarrow eat$$
    $$\ldots$$
  - S is a dedicated start symbol

An example grammar

$$V = \{S, VP, NP, PP, N, V, PN, P\}$$

$$\Sigma = \{girl, telescope, sandwich, I, saw, ate, with, in, a, the\}$$

$$S = \{S\}$$

R :

Inner rules

$S \rightarrow NP \ \ VP$ (NP A girl) (VP ate a sandwich)

$VP \rightarrow V$

$VP \rightarrow V \ \ NP$ (V ate) (NP a sandwich)

$VP \rightarrow VP \ \ PP$ (VP saw a girl) (PP with a telescope)

$NP \rightarrow NP \ \ PP$ (NP a girl) (PP with a sandwich)

$NP \rightarrow D \ \ N$ (D a) (N sandwich)

$NP \rightarrow PN$

$PP \rightarrow P \ \ NP$ (P with) (NP with a sandwich)

Preterminal rules

$N \rightarrow girl$

$N \rightarrow telescope$

$N \rightarrow sandwich$

$PN \rightarrow I$

$V \rightarrow saw$

$V \rightarrow ate$

$P \rightarrow with$

$P \rightarrow in$

$D \rightarrow a$

$D \rightarrow the$

CFGs

S

$$\boxed{S \rightarrow NP\ \ VP}$$

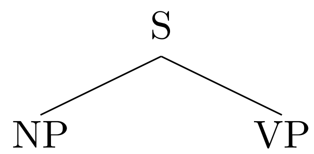
$$VP \rightarrow V$$
$$VP \rightarrow V\ \ NP$$
$$VP \rightarrow VP\ \ PP$$

$$NP \rightarrow NP\ \ PP$$
$$NP \rightarrow D\ \ N$$
$$NP \rightarrow PN$$

$$PP \rightarrow P\ \ NP$$

$$N \rightarrow girl$$
$$N \rightarrow telescope$$
$$N \rightarrow sandwich$$
$$PN \rightarrow I$$
$$V \rightarrow saw$$
$$V \rightarrow ate$$
$$P \rightarrow with$$
$$P \rightarrow in$$
$$D \rightarrow a$$
$$D \rightarrow the$$

CFGs



$$S \rightarrow NP \;\; VP$$

$$VP \rightarrow V$$

$$VP \rightarrow V \;\; NP$$

$$VP \rightarrow VP \;\; PP$$

$$NP \rightarrow NP \;\; PP$$

$$NP \rightarrow D \;\; N$$

$$\boxed{NP \rightarrow PN}$$

$$PP \rightarrow P \;\; NP$$

$$N \rightarrow girl$$

$$N \rightarrow telescope$$

$$N \rightarrow sandwich$$

$$PN \rightarrow I$$

$$V \rightarrow saw$$

$$V \rightarrow ate$$

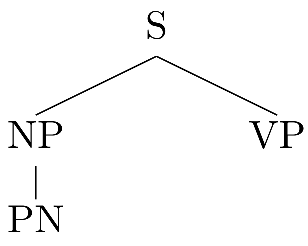
$$P \rightarrow with$$

$$P \rightarrow in$$

$$D \rightarrow a$$

$$D \rightarrow the$$

CFGs



$$S \rightarrow NP \;\; VP$$

$$VP \rightarrow V$$

$$VP \rightarrow V \;\; NP$$

$$VP \rightarrow VP \;\; PP$$

$$NP \rightarrow NP \;\; PP$$

$$NP \rightarrow D \;\; N$$

$$NP \rightarrow PN$$

$$PP \rightarrow P \;\; NP$$

$$N \rightarrow girl$$

$$N \rightarrow telescope$$

$$N \rightarrow sandwich$$

$$\boxed{PN \rightarrow I}$$

$$V \rightarrow saw$$

$$V \rightarrow ate$$

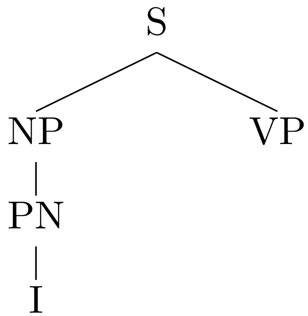
$$P \rightarrow with$$

$$P \rightarrow in$$

$$D \rightarrow a$$

$$D \rightarrow the$$

CFGs



$$S \rightarrow NP \;\; VP$$

$$VP \rightarrow V$$

$$\boxed{VP \rightarrow V \;\; NP}$$

$$VP \rightarrow VP \;\; PP$$

$$NP \rightarrow NP \;\; PP$$

$$NP \rightarrow D \;\; N$$

$$NP \rightarrow PN$$

$$PP \rightarrow P \;\; NP$$

$$N \rightarrow girl$$

$$N \rightarrow telescope$$

$$N \rightarrow sandwich$$

$$PN \rightarrow I$$

$$V \rightarrow saw$$

$$V \rightarrow ate$$

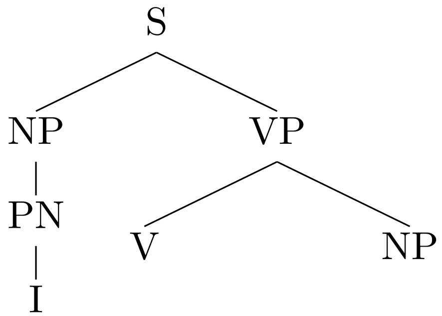
$$P \rightarrow with$$

$$P \rightarrow in$$

$$D \rightarrow a$$

$$D \rightarrow the$$

CFGs



$$S \rightarrow NP \;\; VP$$

$$VP \rightarrow V$$

$$VP \rightarrow V \;\; NP$$

$$VP \rightarrow VP \;\; PP$$

$$NP \rightarrow NP \;\; PP$$

$$NP \rightarrow D \;\; N$$

$$NP \rightarrow PN$$

$$PP \rightarrow P \;\; NP$$

$$N \rightarrow girl$$

$$N \rightarrow telescope$$

$$N \rightarrow sandwich$$

$$PN \rightarrow I$$

$$\boxed{V \rightarrow saw}$$

$$V \rightarrow ate$$

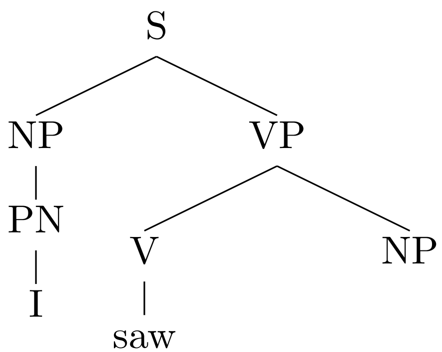
$$P \rightarrow with$$

$$P \rightarrow in$$

$$D \rightarrow a$$

$$D \rightarrow the$$

CFGs



$$S \rightarrow NP \;\; VP$$

$$VP \rightarrow V$$

$$VP \rightarrow V \;\; NP$$

$$VP \rightarrow VP \;\; PP$$

$$\boxed{NP \rightarrow NP \;\; PP}$$

$$NP \rightarrow D \;\; N$$

$$NP \rightarrow PN$$

$$PP \rightarrow P \;\; NP$$

$$N \rightarrow girl$$

$$N \rightarrow telescope$$

$$N \rightarrow sandwich$$

$$PN \rightarrow I$$

$$V \rightarrow saw$$

$$V \rightarrow ate$$

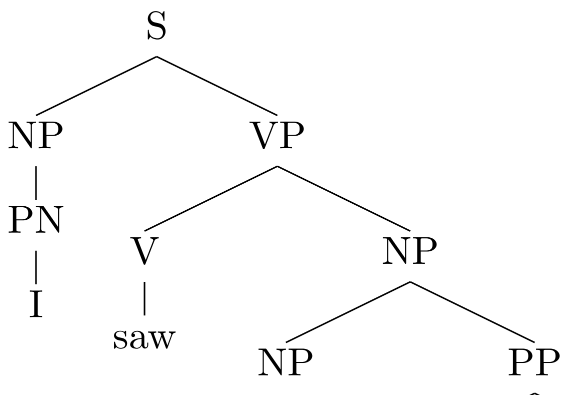
$$P \rightarrow with$$

$$P \rightarrow in$$

$$D \rightarrow a$$

$$D \rightarrow the$$

CFGs



$$S \to NP\ \ VP$$

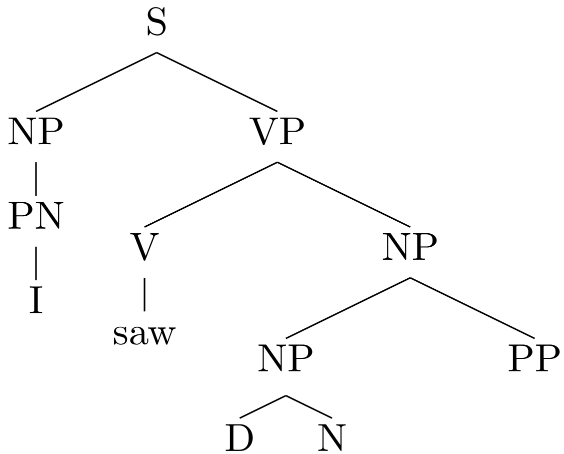
$$VP \to V$$
$$VP \to V\ \ NP$$
$$VP \to VP\ \ PP$$

$$NP \to NP\ \ PP$$
$$\boxed{NP \to D\ \ N}$$
$$NP \to PN$$

$$PP \to P\ \ NP$$

$$N \to girl$$
$$N \to telescope$$
$$N \to sandwich$$
$$PN \to I$$
$$V \to saw$$
$$V \to ate$$
$$P \to with$$
$$P \to in$$
$$D \to a$$
$$D \to the$$

CFGs



$$S \rightarrow NP \ \ VP$$

$$VP \rightarrow V$$

$$VP \rightarrow V \ \ NP$$

$$VP \rightarrow VP \ \ PP$$

$$NP \rightarrow NP \ \ PP$$

$$NP \rightarrow D \ \ N$$

$$NP \rightarrow PN$$

$$PP \rightarrow P \ \ NP$$

$$N \rightarrow girl$$

$$N \rightarrow telescope$$

$$N \rightarrow sandwich$$

$$PN \rightarrow I$$

$$V \rightarrow saw$$

$$V \rightarrow ate$$

$$P \rightarrow with$$

$$P \rightarrow in$$

$$D \rightarrow a$$

$$D \rightarrow the$$

CFGs

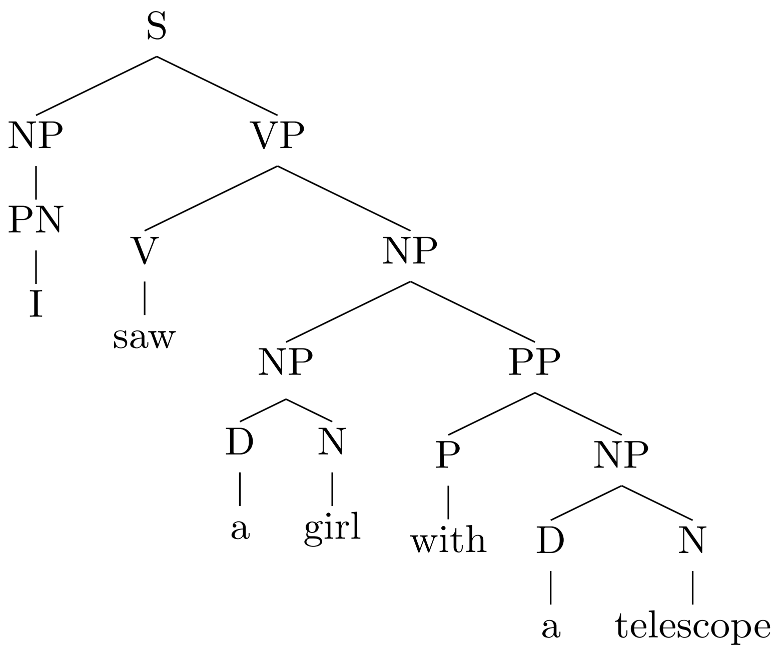
$$S \to NP\ \ VP$$

$$VP \to V$$
$$VP \to V\ \ NP$$
$$VP \to VP\ \ PP$$

$$NP \to NP\ \ PP$$
$$NP \to D\ \ N$$
$$NP \to PN$$

$$PP \to P\ \ NP$$

$$N \to girl$$
$$N \to telescope$$
$$N \to sandwich$$
$$PN \to I$$
$$V \to saw$$
$$V \to ate$$
$$P \to with$$
$$P \to in$$
$$D \to a$$
$$D \to the$$

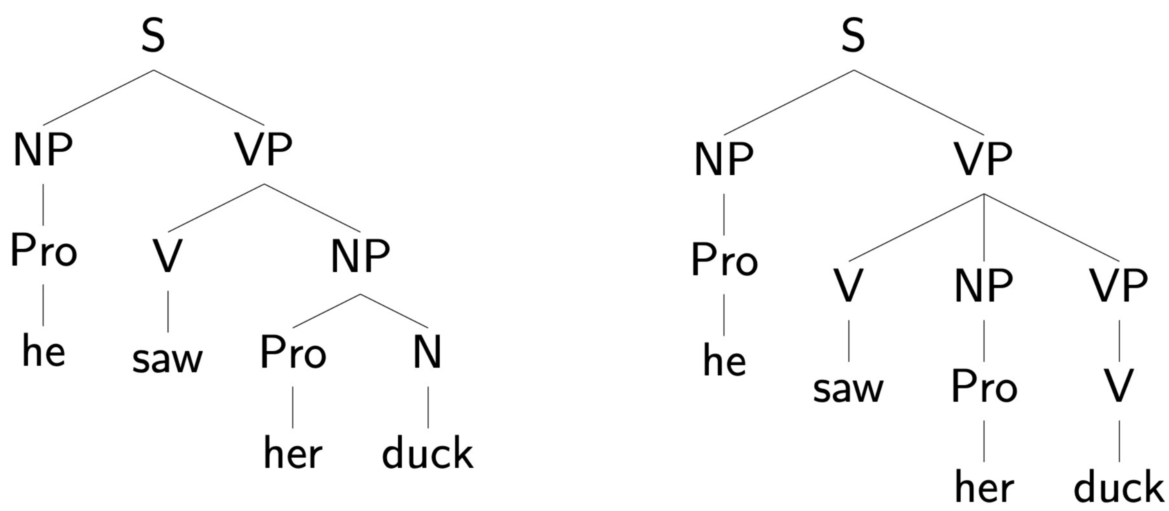


CFG defines both:

- a set of strings (a language)
- structures used to represent sentences (constituent trees)

Structural ambiguity

- Some sentences have more than one parse: structural ambiguity.



- Here, the structural ambiguity is caused by PoS ambiguity in several of the words. (Both are types of syntactic ambiguity.)

Structural ambiguity

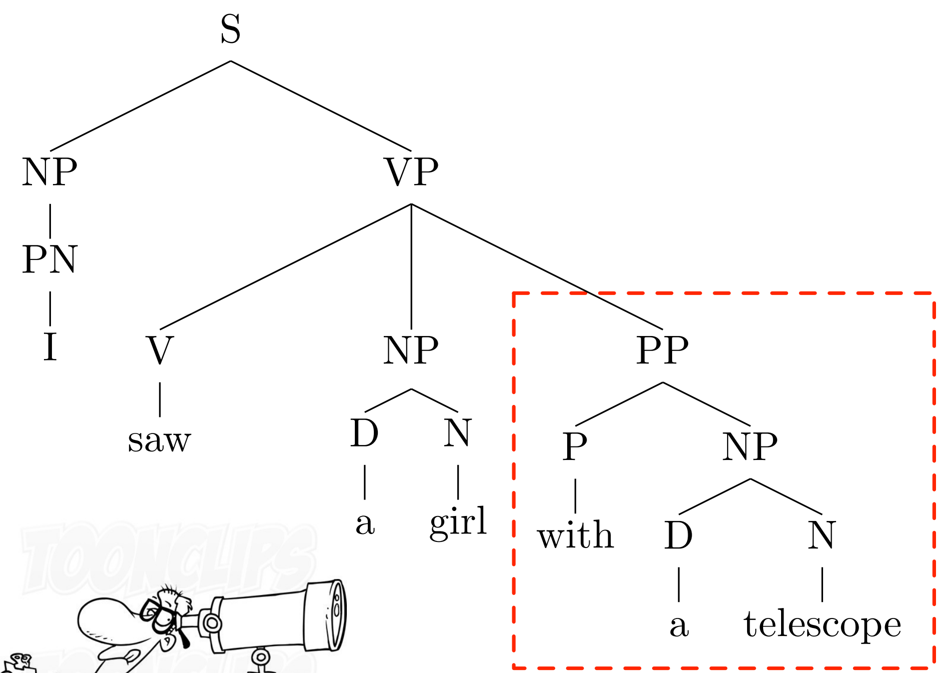
- Some sentences have structural ambiguity even **without** part-of-speech ambiguity. This is called attachment ambiguity.
  - Depends on where different phrases attach in the tree.
  - Different attachments have different meanings:

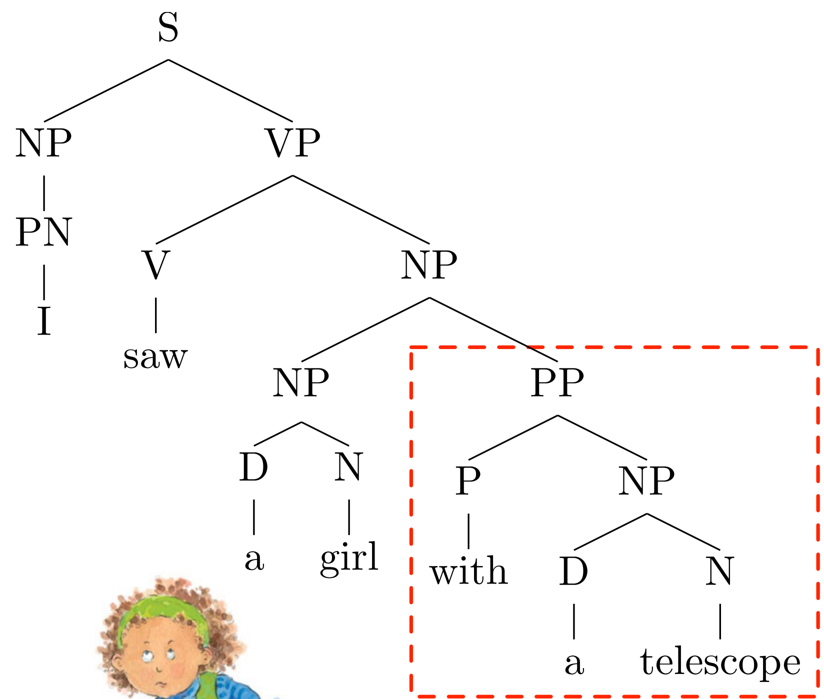
I saw a girl with a telescope
She ate the pizza on the floor
Good boys and girls get presents from Santa

- Next slide shows trees for the first example: prepositional phrase (PP) attachment ambiguity.

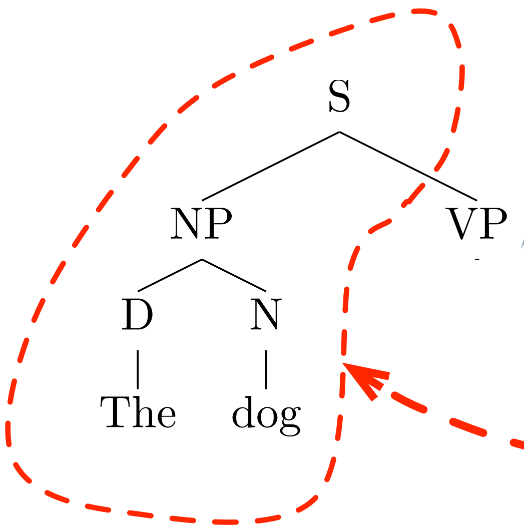
Prepositional Phrase (PP-) Attachment Ambiguity

- Prepositional phrase attachment ambiguity



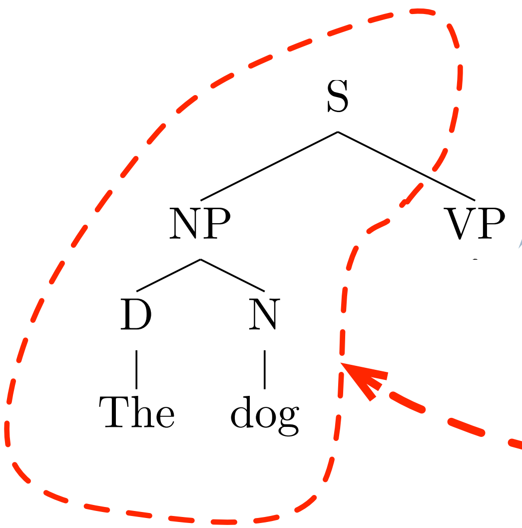


Why context-free?

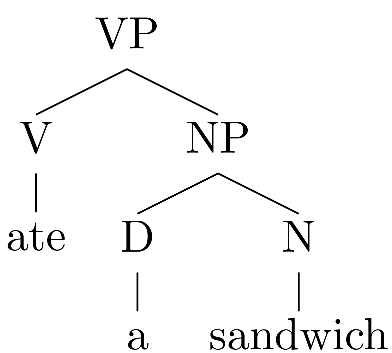


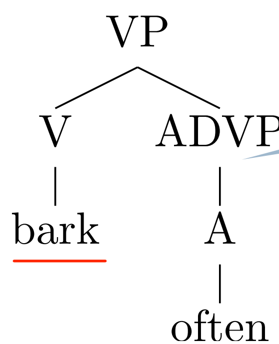
What can be a sub-tree is only affected by what the phrase type is (VP) but not the context

Why context-free?



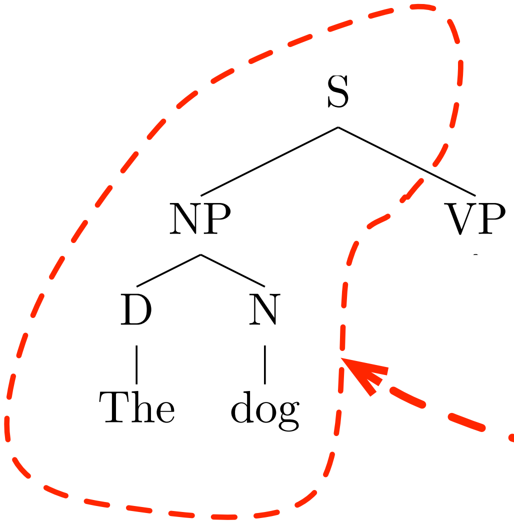
What can be a sub-tree is only affected by what the phrase type is (VP) but not the context



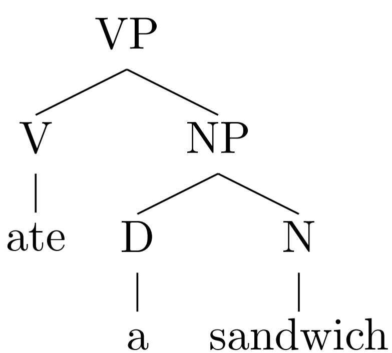


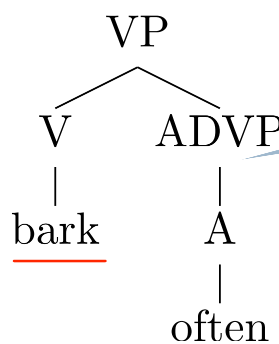
Not grammatical

Why context-free?



What can be a sub-tree is only affected by what the phrase type is (VP) but not the context





Not grammatical

Matters if we want to generate language (e.g., language modeling) but is this relevant to parsing?

Key problems

- Recognition problem: does the sentence belong to the language defined by CFG?
  - That is: is there a derivation which yields the sentence?
- Parsing problem: what is a (most plausible) derivation (tree) corresponding the sentence?

- Parsing problem encompasses the recognition problem

Today

- Basics of Syntax and Context-Free Grammars
- Classes of Syntactic Parsing Algorithms
- Start with the CKY algorithm

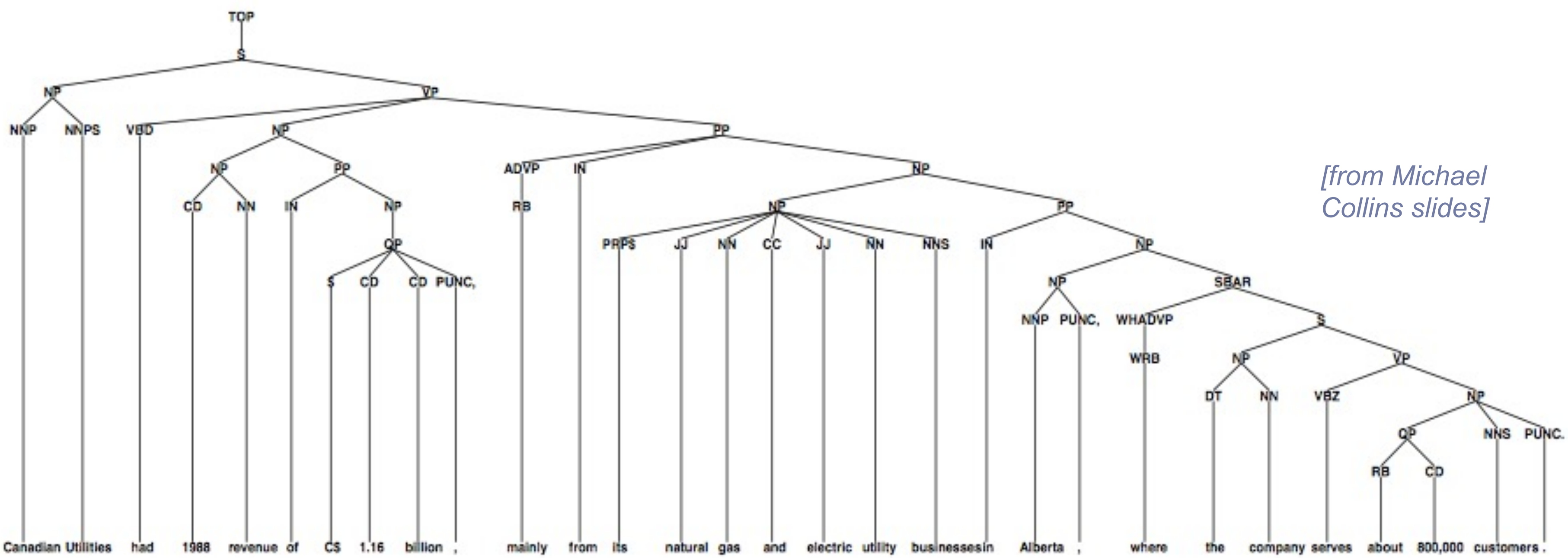
Parsing algorithms

- Goal: compute the structure(s) for an input string given a grammar.
  - (we may want to use the structure to interpret meaning)
  - As usual, ambiguity is a huge problem.

- For correctness: need to find the right structure to get the right meaning.
- For efficiency: searching all possible structures can be very slow
  - want to use parsing for large-scale language tasks

Parsing is hard

- A typical tree from a standard dataset (Penn treebank WSJ)



[from Michael Collins slides]

Canadian Utilities had 1988 revenue of $ 1.16 billion , mainly from its natural gas and electric utility businesses in Alberta , where the company serves about 800,000 customers .

Parser properties

All parsers have two fundamental properties:

- Directionality: the sequence in which the structures are constructed.
  - Top-down: start with root category (S), choose expansions, build down to words.
  - Bottom-up: build subtrees over words, build up to S.
  - Mixed strategies also possible (e.g., left corner parsers)
- Search strategy: the order in which the search space of possible analyses is explored.

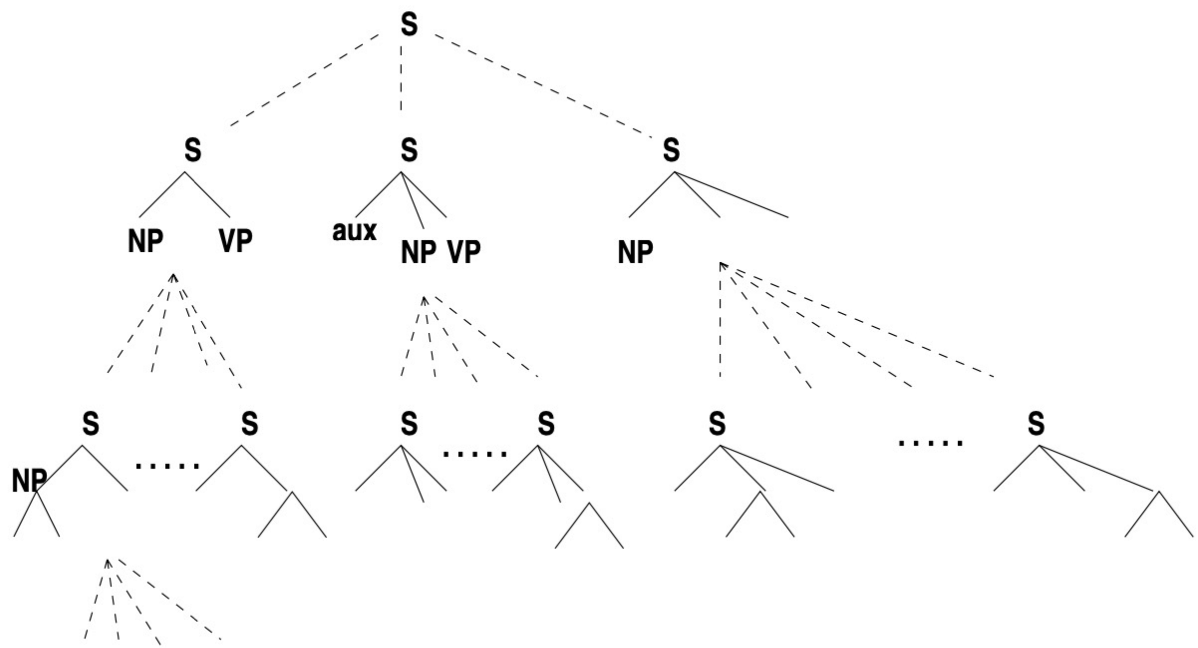
Parser properties

All parsers have two fundamental properties:

- Directionality: the sequence in which the structures are constructed.
  - Top-down: start with root category (S), choose expansions, build down to words.
  - Bottom-up: build subtrees over words, build up to S.
  - Mixed strategies also possible (e.g., left corner parsers)
- Search strategy: the order in which the search space of possible analyses is explored.

Search space for a top-down parser

- Start with S node.
- Choose one of many possible expansions.
- Each of which has children with many possible expansions...
- etc



Search strategies

All parsers have two fundamental properties:

- Depth-first search: explore one branch of the search space at a time, as far as possible. If this branch is a dead-end, parser needs to backtrack.

- Breadth-first search: expand all possible branches in parallel (or simulated parallel). Requires storing many incomplete parses in memory at once.

- Best-first search: score each partial parse and pursue the highest-scoring options first.

We will now consider a bottom-up parser which uses dynamic programming to explore the space

Today

- Basics of Syntax and Context-Free Grammars
- Classes of Syntactic Parsing Algorithms
- Start with the CKY algorithm

CKY algorithm (aka CYK)

- Cocke-Kasami-Younger algorithm
  - Independently discovered in late 60s / early 70s
- An efficient bottom-up parsing algorithm for CFGs
  - can be used both for the recognition and parsing problems
- Very important in NLP (and beyond)
- As we will see, it is generalizable to probabilistic modeling / PCFGs

Constraints on the grammar

- The basic CKY algorithm supports only rules in the *Chomsky Normal Form* (CNF):

$$C \rightarrow x$$

Unary preterminal rules, generation of words given PoS tags
$D \rightarrow the \quad N \rightarrow telescope$

$$C \rightarrow C_1 C_2$$

Binary inner rules (e.g., $S \rightarrow NP\ VP,\ NP \rightarrow D\ \ N$)

Constraints on the grammar

- The basic CKY algorithm supports only rules in the *Chomsky Normal Form (CNF)*:

$$C \rightarrow x$$

$$C \rightarrow C_1 C_2$$

Unary preterminal rules, generation of words given PoS tags
$D \rightarrow the \quad N \rightarrow telescope$

Binary inner rules (e.g., $S \rightarrow NP\ VP,\ NP \rightarrow D\ \ N$)

- Any CFG can be converted to an equivalent CNF
  - Equivalent means that they define the same language
  - However (syntactic) trees will look differently
  - It is possible to address it but defining such transformations that allows for easy reverse transformation

Transformation to CNF form

- What one need to do to convert to CNF form
  - Get rid of empty (aka epsilon) productions: $C \rightarrow \epsilon$
  - Get rid of unary rules: $C \rightarrow C_1$
  - N-ary rules: $C \rightarrow C_1 \; C_2 \ldots C_n \quad (n > 2)$

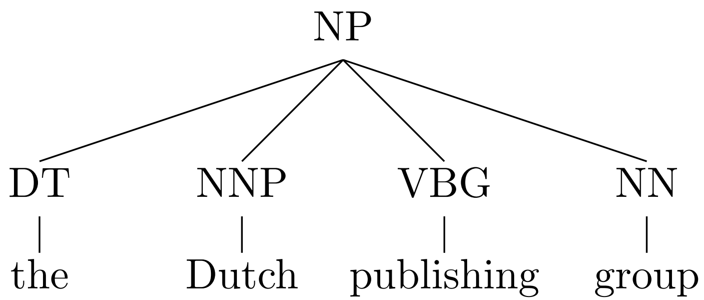
Generally not a problem as there are no empty production in the standard treebanks (or they can be disregarded)

Not a problem, as our CKY algorithm will support unary rules

Crucial to process them, as required for efficient parsing

Transformation to CNF form: binarization

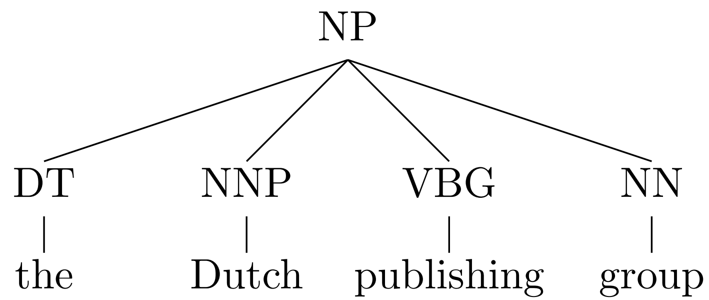
- Consider $NP \rightarrow DT\ \ NNP\ \ VBG\ \ NN$



- How do we get a set of binary rules which are equivalent?

Transformation to CNF form: binarization

- Consider $NP \rightarrow DT \;\; NNP \;\; VBG \;\; NN$



- How do we get a set of binary rules which are equivalent?

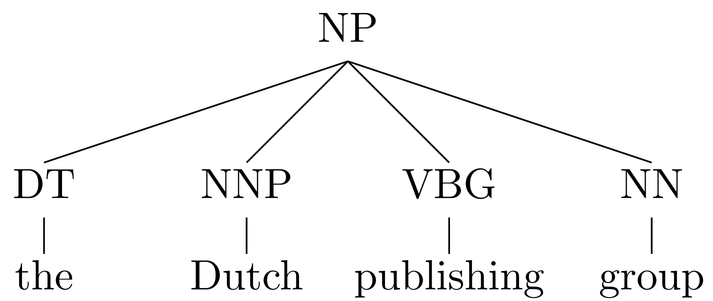
$$NP \rightarrow DT \;\; X$$

$$X \rightarrow NNP \;\; Y$$

$$Y \rightarrow VBG \;\; NN$$

Transformation to CNF form: binarization

- Consider $NP \rightarrow DT \;\; NNP \;\; VBG \;\; NN$



- How do we get a set of binary rules which are equivalent?

$NP \rightarrow DT \;\; X$

$X \rightarrow NNP \;\; Y$

$Y \rightarrow VBG \;\; NN$

- A more systematic way to refer to new non-terminals

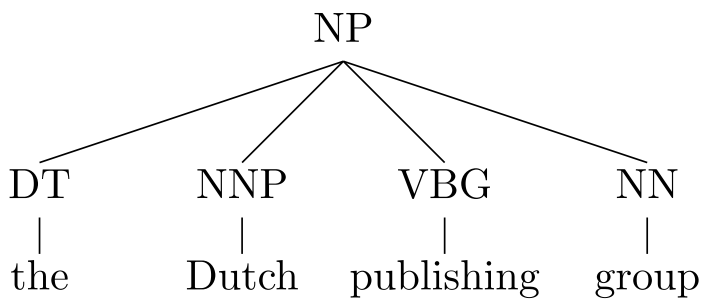
$NP \rightarrow DT \;\; @NP|DT$

$@NP|DT \rightarrow NNP \;\; @NP|DT\_NNP$

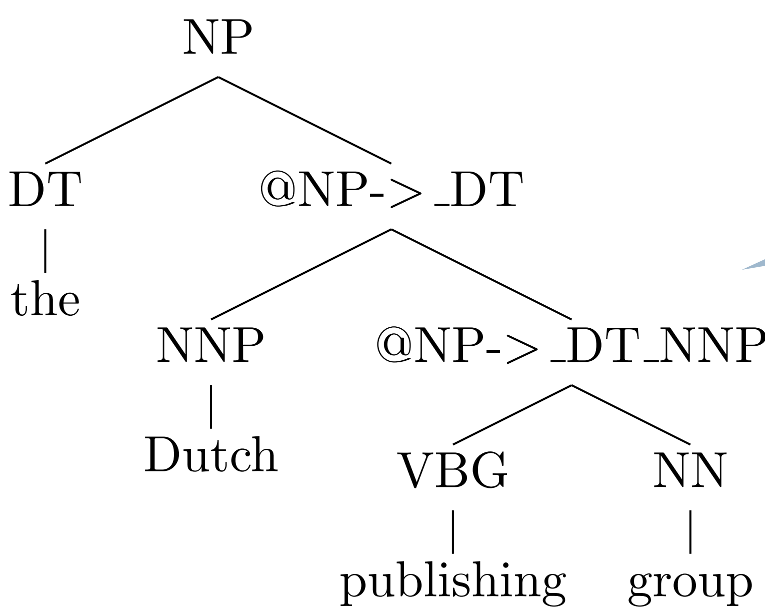
$@NP|DT\_NNP \rightarrow VBG \;\; NN$

Transformation to CNF form: binarization

- Instead of binarizing tules we can binarize trees on preprocessing:



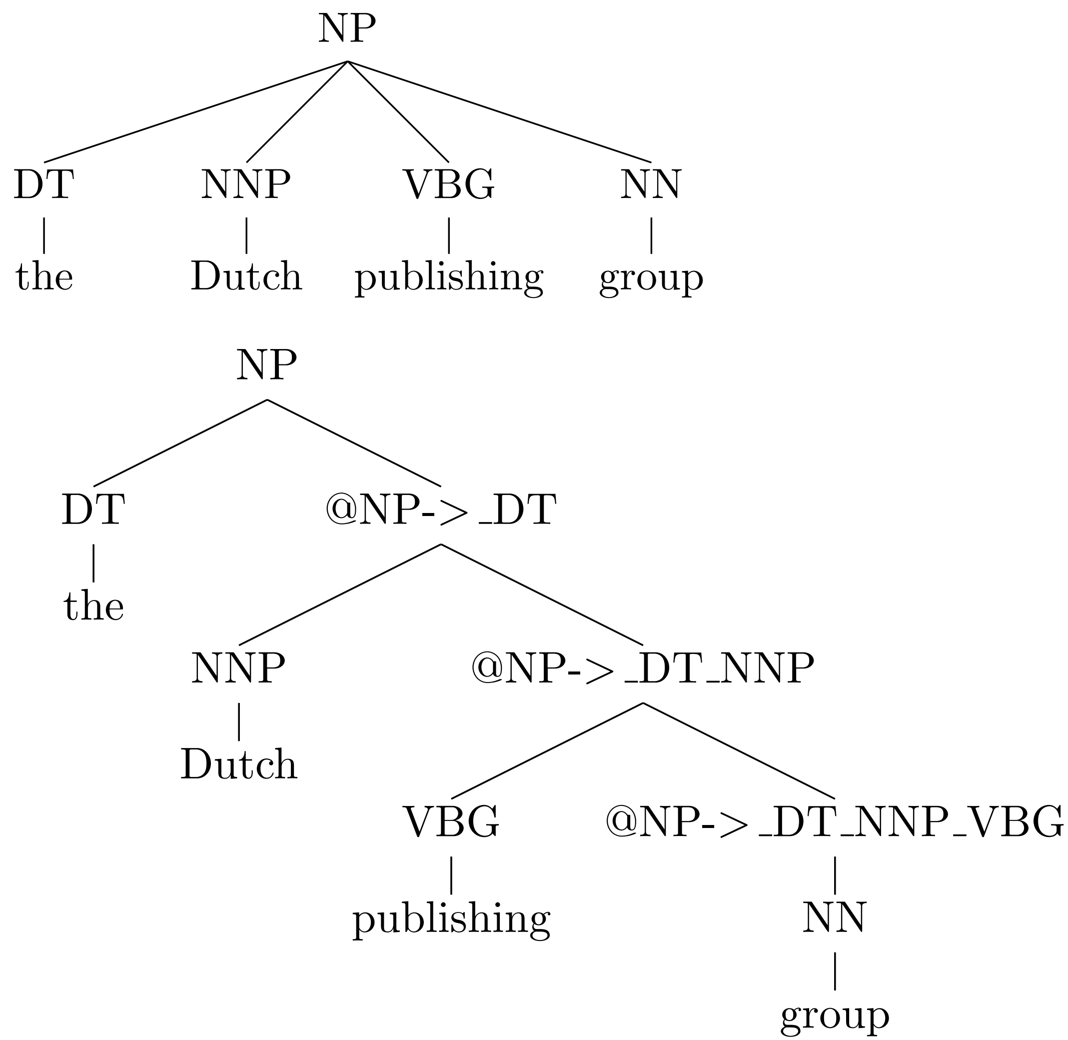
Also known as **lossless Markovization** in the context of PCFGs



Can be easily reversed on postprocessing

Transformation to CNF form: binarization

- Instead of binarizing tules we can binarize trees on preprocessing:



Summary

- Basics of Syntax and Context-Free Grammars
- Classes of Syntactic Parsing Algorithms
- Started with the CKY algorithm – preparing the grammar

Next lectures(s):

- CKY algorithm
- Probabilistic parsing with PCFGs
- PCFG parsing beyond treebank grammars
- Neuralized PCFG algorithms