NLP1

Syntax (and Morphology)

Wilker Aziz
probabll.github.io

w.aziz@uva.nl

ILLC

Where are we at?

Week 1

- HC1a: text classification
- HC1b: language modelling

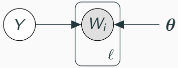
Week 2

- HC2a: sequence labelling
- Today: **syntax (and morphology)**

Modelling Language Thus Far

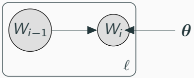
Bag-of-words (e.g., NB)

- ignore word order entirely



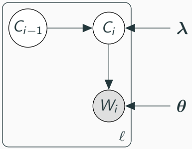
Markov models (e.g., bigram LM)

- memorise short observed phrases



HMM

- capture shallow syntactic patterns through adjacent word classes
- no semantic dependency amongst words



If you want to learn more about how to read these diagrams, you can check a link from our earlier classes: introduction to PGMs

Phrase Categories

Much like we abstracted from words to their (syntactic) categories, we can abstract from phrases to their syntactic categories.

On Mentimeter...

Generalising POS categories

POS categories indicate which words are substitutable:

I saw a [ADJ] cat.

Phrasal categories indicate which *phrases* are substitutable:

$[_{\mathrm{NP}} \ldots]$ sleep soundly.

Phrasal categories; noun phrase (NP), verb phrase (VP), prepositional phrase (PP), etc.

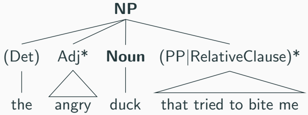
Heads and Phrases

The class that a word belongs to is closely linked to the name of the phrase it customarily appears in.

In English,

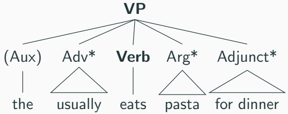
NPs are commonly of the form

- (Det) Adj* **Noun** (PP|RelClause)*



VPs are commonly of the form

- (Aux) Adv* **Verb** Arg* Adjunct*



Constituency

Syntactic constituency is the idea that groups of words can behave as single units, or constituents.

Example: noun phrases (NPs)

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

One evidence for their existence is that they appear in similar syntactic environments (e.g., NPs tend to appear before a verb).

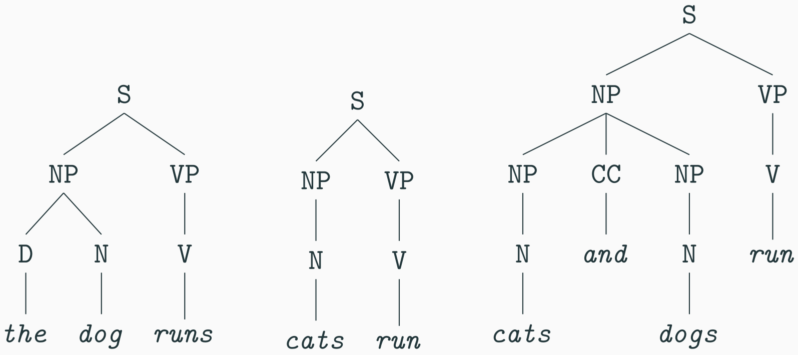
three parties from Brooklyn *arrive*…
a high-class spot such as Mindy's *attracts*…
the Broadway coppers *love*…
they *sit*

Figures from Section 18.1 of textbook.

Nesting

Constituents can be hierarchically embedded in other constituents.

You can view the result as a tree-like structure.



Context-Free Grammars

Context-Free Grammar (CFG)

A rewriting system with two types of *symbols* and a set of *symbol-rewriting rules*.

Symbols

- *Terminals* (or *constants*): words;
- *Nonterminals* (or *variables*): word and phrasal categories.

Rules

- $\mathtt{X} \rightarrow \beta$ where $\mathtt{X}$ is a nonterminal, and β is any string of terminal and nonterminal symbols.

Example

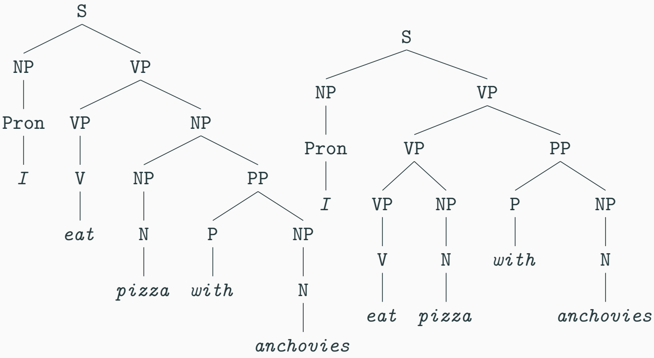
Nonterminals: S, NP, VP, PP, Pron, N, V, P

Terminals: *I, eat, pizza, with, anchovies*

- S $\rightarrow$ NP VP
- NP $\rightarrow$ NP PP
- VP $\rightarrow$ VP PP
- VP $\rightarrow$ VP NP
- PP $\rightarrow$ P NP
- VP $\rightarrow$ V
- NP $\rightarrow$ Pron
- NP $\rightarrow$ N
- V $\rightarrow$ *eat*
- Pron $\rightarrow$ *I*
- P $\rightarrow$ *with*
- N $\rightarrow$ *pizza*
- N $\rightarrow$ *anchovies*

Parses

Text: *I eat pizza with anchovies*



CFG Formally

Σ is a finite set of terminal symbols

$\mathcal{V}$ is a finite set of nonterminal symbols, with a distinguished start symbol $\mathsf{S} \in \mathcal{V}$ and $\mathcal{V} \cap \Sigma = \emptyset$

$\mathcal{R}$ is a finite set of rules of the kind: $\mathtt{X} \to \beta$ with $\mathtt{X} \in \mathcal{V}$ and $\beta \in (\Sigma \cup \mathcal{V})^*$.

A CFG is the tuple $\langle \Sigma, \mathcal{V}, \mathsf{S}, \mathcal{R} \rangle$

CFG Terminology

Arity (length of rule's RHS)

- unary: $\mathtt{A} \rightarrow \mathtt{B}$
- binary: $\mathtt{X} \rightarrow \mathtt{B}\,\mathtt{C}$
- *n*-ary: $\mathtt{X} \rightarrow \mathtt{X}_1, \ldots, \mathtt{X}_n$
- if the longest rule has arity *a*, we say the grammar has arity *a*

No matter the CFG, we can always re-express the same set of strings in *Chomsky normal form* (CNF), which gives us a grammar of arity 2.

Derivation

We can use CFGs to derive strings

A derivation is a sequence of strings

- we start from the string $\langle \mathtt{S} \rangle$
- and at each step we rewrite the leftmost nonterminal $\mathtt{X}$ by application of a rule $\mathtt{X} \to \beta$
- until only terminals remain

If a string $w_1 \cdots w_\ell$ is derivable from $\mathtt{S}$ we write: $\mathsf{S} \overset{*}{\Rightarrow} w_1 \cdots w_\ell$.

Example

Rule	Derivation
$\mathtt{BoS} \rightarrow \mathtt{S}$	$\langle \mathtt{S} \rangle$

Example

Rule	Derivation
$\texttt{BoS} \rightarrow \texttt{S}$	$\langle\texttt{S}\rangle$
$\texttt{S} \rightarrow \texttt{NP VP}$	$\langle\texttt{NP VP}\rangle$

Example

Rule	Derivation
$\mathtt{BoS} \rightarrow \mathtt{S}$	$\langle \mathtt{S} \rangle$
$\mathtt{S} \rightarrow \mathtt{NP\ VP}$	$\langle \mathtt{NP\ VP} \rangle$
$\mathtt{NP} \rightarrow \mathtt{D\ N}$	$\langle \mathtt{D\ N\ VP} \rangle$

Example

Rule	Derivation
$\mathtt{BoS} \rightarrow \mathtt{S}$	$\langle \mathtt{S} \rangle$
$\mathtt{S} \rightarrow \mathtt{NP\ VP}$	$\langle \mathtt{NP\ VP} \rangle$
$\mathtt{NP} \rightarrow \mathtt{D\ N}$	$\langle \mathtt{D\ N\ VP} \rangle$
$\mathtt{D} \rightarrow \textit{the}$	$\langle \textit{the}\ \mathtt{N\ VP} \rangle$

Example

Rule	Derivation
$\texttt{BoS} \rightarrow \texttt{S}$	$\langle\texttt{S}\rangle$
$\texttt{S} \rightarrow \texttt{NP VP}$	$\langle\texttt{NP VP}\rangle$
$\texttt{NP} \rightarrow \texttt{D N}$	$\langle\texttt{D N VP}\rangle$
$\texttt{D} \rightarrow \textit{the}$	$\langle\textit{the}\ \texttt{N VP}\rangle$
$\texttt{N} \rightarrow \textit{dog}$	$\langle\textit{the dog}\ \texttt{VP}\rangle$

Example

Rule	Derivation
$\texttt{BoS} \rightarrow \texttt{S}$	$\langle \texttt{S} \rangle$
$\texttt{S} \rightarrow \texttt{NP VP}$	$\langle \texttt{NP VP} \rangle$
$\texttt{NP} \rightarrow \texttt{D N}$	$\langle \texttt{D N VP} \rangle$
$\texttt{D} \rightarrow \textit{the}$	$\langle \textit{the}\ \texttt{N VP} \rangle$
$\texttt{N} \rightarrow \textit{dog}$	$\langle \textit{the dog}\ \texttt{VP} \rangle$
$\texttt{VP} \rightarrow \texttt{V}$	$\langle \textit{the dog}\ \texttt{V} \rangle$

Example

Rule	Derivation
$\texttt{BoS} \rightarrow \texttt{S}$	$\langle \texttt{S} \rangle$
$\texttt{S} \rightarrow \texttt{NP VP}$	$\langle \texttt{NP VP} \rangle$
$\texttt{NP} \rightarrow \texttt{D N}$	$\langle \texttt{D N VP} \rangle$
$\texttt{D} \rightarrow \textit{the}$	$\langle \textit{the}\ \texttt{N VP} \rangle$
$\texttt{N} \rightarrow \textit{dog}$	$\langle \textit{the dog}\ \texttt{VP} \rangle$
$\texttt{VP} \rightarrow \texttt{V}$	$\langle \textit{the dog}\ \texttt{V} \rangle$
$\texttt{V} \rightarrow \textit{barks}$	$\langle \textit{the dog barks} \rangle$

We can denote the derivation (i.e., sequence of rule applications) by δ.

The fact that δ derives a specific string (e.g., *the dog barks*) can be denoted by $\texttt{S} \stackrel{\delta}{\Rightarrow} w_1 \cdots w_\ell$.

So we write some rules. What could go wrong?

On Mentimeter...

Probabilistic Context-Free Grammars

Probabilistic CFG (PCFG)

A probability distribution over the space of all derivations (including their yields) supported by a grammar.

Factorisation

Factorisation

A random derivation $D = \langle R_1, \ldots, R_M \rangle$ is a sequence of M random rule applications. A valid derivation rewrites $\mathtt{S}$ into a sequence of random words $X = \langle W_1, \ldots, W_L \rangle$.

The CFG backbone gives us a mechanism to factorise probabilities. We can assign probability mass to $r_{1:m}$ via chain rule

$$P_D(r_{1:m}) = \prod_{j=1}^{m} P_{R|H}(r_j | r_{<j})$$

$r_{<j}$ is the *history* of rule applications relative to the jth rule in the derivation.

Markov Model over Steps

A random derivation $D = \langle R_1, \ldots, R_M \rangle$ is a sequence of M random rule applications.

A random rule is a pair (N, S) of a random LHS nonterminal and a random RHS string.

We can assign probability mass to $r_{1:m}$ via chain rule under a Markov assumption:

$$P_D(r_{1:m}) \overset{\text{ind.}}{=} \prod_{j=1}^{m} P_R(r_j) = \prod_{j=1}^{m} P_{S|N}(\beta_j | v_j)$$

with $v_j \in \mathcal{V}$ and $\beta_j \in (\mathcal{V} \cup \Sigma)^*$

Note: pretend every derivation starts with $r_0 = \texttt{BoS} \to \texttt{S}$.

Example

Rule Probability	Derivation
1	$\langle \mathtt{S} \rangle$
$P_{S\|N}(\mathtt{NP\ VP}\|\mathtt{S})$	$\langle \mathtt{NP\ VP} \rangle$
$P_{S\|N}(\mathtt{D\ N}\|\mathtt{NP})$	$\langle \mathtt{D\ N\ VP} \rangle$
$P_{S\|N}(\mathit{the}\|\mathtt{D})$	$\langle \mathit{the}\ \mathtt{N\ VP} \rangle$
$P_{S\|N}(\mathit{dog}\|\mathtt{N})$	$\langle \mathit{the}\ \mathit{dog}\ \mathtt{VP} \rangle$
$P_{S\|N}(\mathtt{V}\|\mathtt{VP})$	$\langle \mathit{the}\ \mathit{dog}\ \mathtt{V} \rangle$
$P_{S\|N}(\mathit{barks}\|\mathtt{V})$	$\langle \mathit{the}\ \mathit{dog}\ \mathit{barks} \rangle$

Generative Story

1. Start with $D = \langle \mathtt{S} \rangle$
2. If all symbols in D are terminal, stop. Else, go to (3).
3. Condition on the left-most nonterminal symbol ν in the derivation, and draw a RHS string β with probability $P_{S|N}(\beta|v)$, replace ν in D by β. Repeat from (2).

This corresponds to a **depth-first** expansion of nonterminals. See my commented Colab demo.

Parameterisation

If we can rewrite a nonterminal variable ν into K different ways, associate a K-dimensional Categorical distribution with $S|N = \nu$:

Examples:

$$S|N = \texttt{S} \sim \text{Categorical}(\theta_{\texttt{S} \to \mathsf{NP\ VP}}, \theta_{\texttt{S} \to \mathsf{VP}})$$
$$S|N = \texttt{N} \sim \text{Categorical}(\theta_{\texttt{N} \to \mathit{cat}}, \theta_{\texttt{N} \to \mathit{dog}}, \theta_{\texttt{N} \to \mathit{bird}})$$

The pmf assigns mass $\prod_{j=1}^{m} \theta_{\nu_j \to \beta_j}$ to a derivation $r_{1:m} = \langle \nu_1 \to \beta_1, \ldots, \nu_m \to \beta_m \rangle$

Parameter Estimation via MLE

Relative frequency of observed rule application

$$\theta_{\nu \to \beta} = \frac{\mathsf{count}(\nu \to \beta)}{\sum_{(\nu \to \gamma) \in \mathcal{R}} \mathsf{count}(\nu \to \gamma)}$$

Evaluation

On Mentimeter...

Probability of Sentence

Due to structural ambiguities, there are potentially many derivations for any one sentence $w_{1:\ell}$.

The PCFG assigns **marginal probability**

$$P_X(w_{1:\ell}) = \sum_{\delta : \mathrm{S} \overset{\delta}{\Rightarrow} w_{1:\ell}} P_D(\delta)$$

equal to the sum of the probabilities of all derivations whose yield is the sentence we want a marginal probability for.

The sum is over the space of all derivations that have $w_{1:\ell}$ as yield.

Evaluation as LM

Assess perplexity of model using marginal probability of sentences in heldout dataset of valid sentences.

Evaluation as Syntactic Parser

Obtain the **most probable derivation** subject to its yield being the sentence we want to parse:

$$\hat{\delta} = \underset{\delta : \mathrm{S} \overset{\delta}{\Rightarrow} w_{1:\ell}}{\arg\max}\ P_D(\delta)$$

Compare model prediction to a human annotated tree. Think of it in terms of span classification (i.e., did we classify $w_{i:j}$ correctly as an NP?), report the constituent label precision, recall, F_1.

See section 18.8 of textbook.

Parse Forest

The key to both uses of PCFG (as an LM or as a parser) is to find all derivations of a given sentence $w_{1:\ell}$, a set we refer to as a **parse forest** for $w_{1:\ell}$.

Parse Forest

The key to both uses of PCFG (as an LM or as a parser) is to find all derivations of a given sentence $w_{1:\ell}$, a set we refer to as a **parse forest** for $w_{1:\ell}$.

Enumeration is bad. We typically work with binary-branching trees (arity=2), then the number of trees for a sentence of L words is the Catalan number $C_L = \frac{(2L)!}{(L+1)!L!}$.

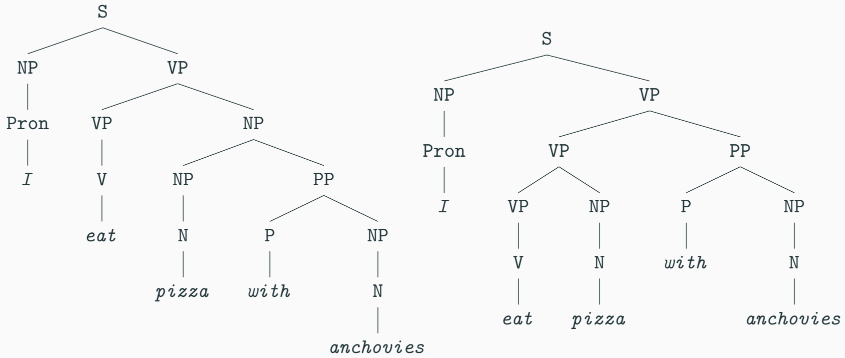
CKY

But to find the sum of probabilities or the maximum probability we do not need to **enumerate** the trees, we can exploit the Markov assumption in yet another dynamic programme.

The CKY algorithm is a compact representation of a forest. It can be used to find marginal probability (Inside algorithm) and maximum probability (Viterbi algorithm).

CKY - Intuition

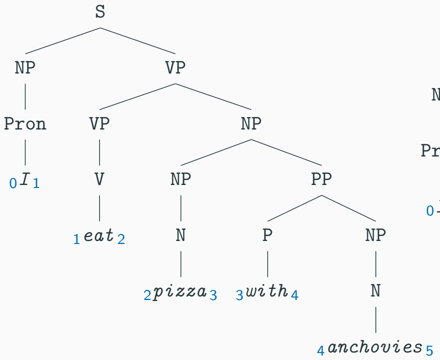
A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.



CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

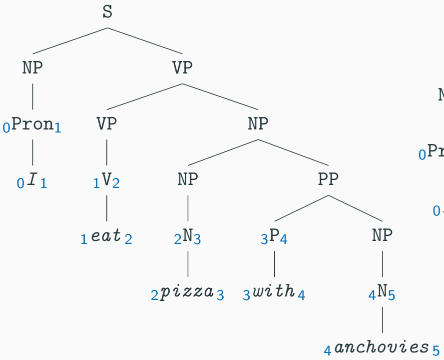
A span is identified by a pair of positions from 0 to ℓ



CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

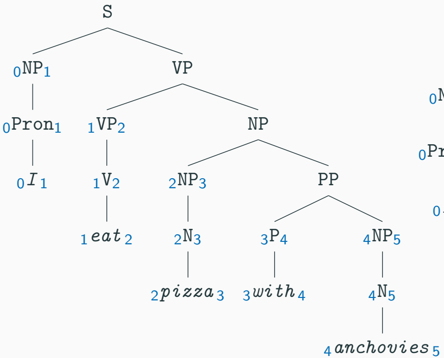
POS tags span single words

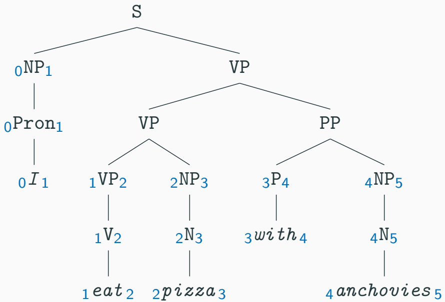


CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

Some phrase categories are derived by unary rules

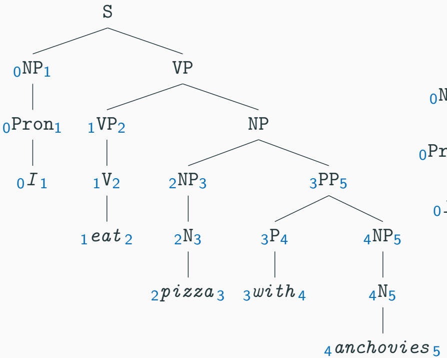


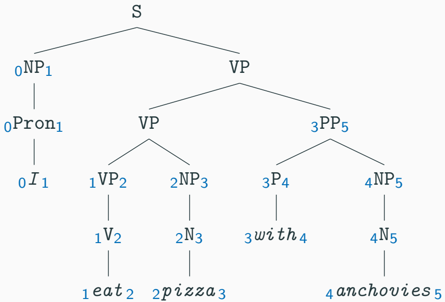


CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

Some phrase categories merge two spans

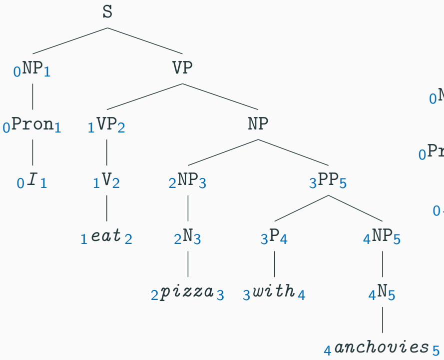


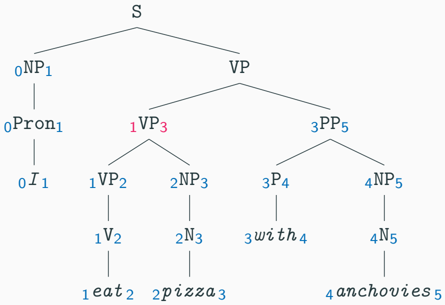


CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

Some spans are common across derivations, while others

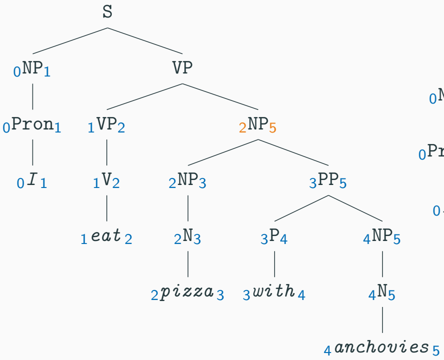




CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

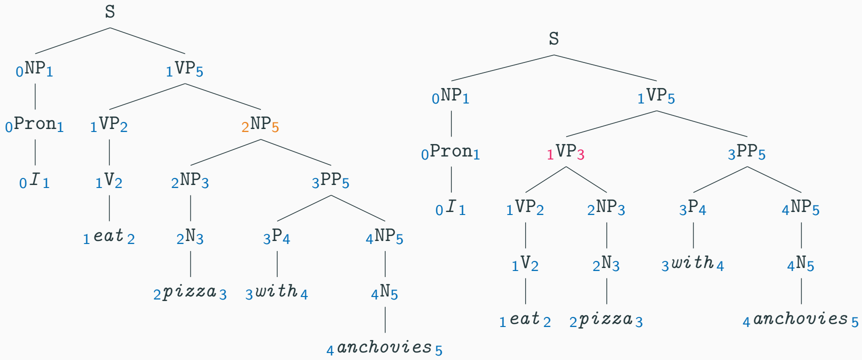
Some spans are common across derivations, while others aren't



CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

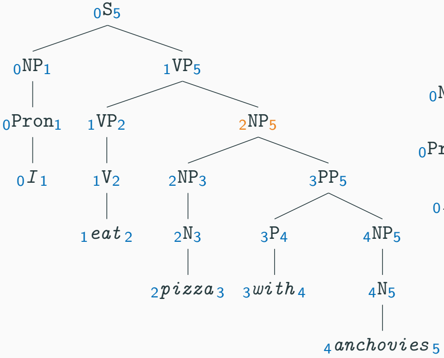
We may derive the same phrase category in different ways

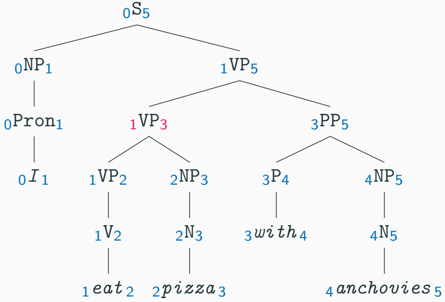


CKY - Intuition

A *forest* is made of elementary building blocks that are reused: *phrase categories over input spans*.

We may derive the same phrase category in different ways



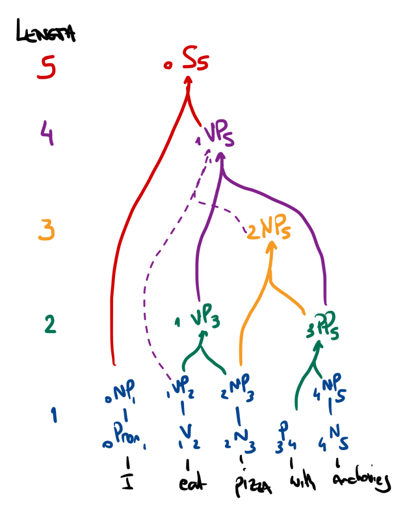


CKY - Intuition

We can construct a graph-like view of the forest, compressed to size that is cubic in sentence length.

Resources:

- my video tutorial
- Appendix C of textbook can be useful.



Limitations

Limited use of linguistic context due to generative formulation.

The context-free assumption is not enough in general: some linguistic constructions violate it.

Dynamic programming for PCFGs takes time that is cubic in sentence length.

Extensions

Like the HMM, the PCFG is a generative model. If all we care about is a mechanism to predict parse trees, then we can parameterise the model conditionally using an expressive feature function. Examples: transition-based parsers, CRF parsers.

We may care about relations between words, more so than constituency, for that we develop **dependency grammars**. Optional reading: Chapter 19 of textbook.

What Next

Required self-study

- Watch the video on dynamic programming for PCFGs
- Watch Katia's class on Morphology
  (first 43 minutes of the video)

Other, useful (but optional) material

- Check the colab demo on sampling from PCFGs
- Chapter 18 of textbook
- Appendix C of textbook

References