L90: Overview of Natural Language Processing

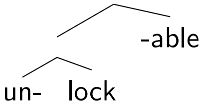
Lecture 4: Phrase Structures and Structured Prediction

Weiwei Sun

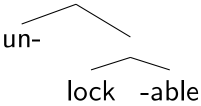
Department of Computer Science and Technology
University of Cambridge

Michaelmas 2020/21

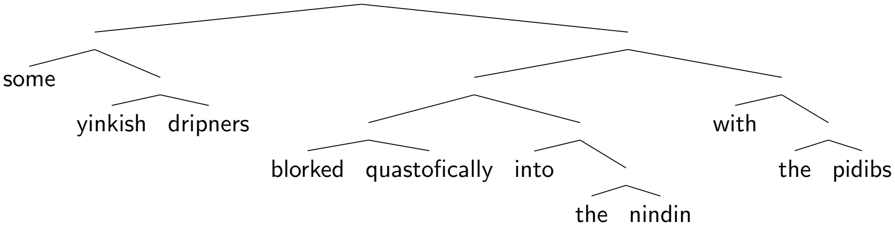
unlockable



Capable of being unlocked.



Not capable of being locked.



Words are organized into nested blocks

Lecture 4: Phrase Structures and Structured Prediction

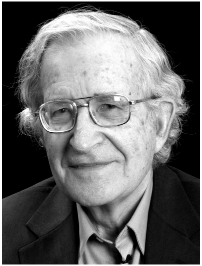
1. Phrase structures
2. Structured prediction
3. Context-free grammars
4. Probabilistic context-free grammars
5. Rethink part-of-speech tagging

Phrase Structures

Interview of Noam Chomsky by Lex Fridman

(1) a. the guy who fixed the car carefully packed his tools
b. carefully the guy who fixed the car packed his tools
c. carefully the guy who fixed the car is tall

I think the deepest property of language and puzzling property that's been discovered is what is sometimes called structure dependence. [...] Linear closeness is an easy computation, but here you're doing a much more, what looks like a more complex computation.



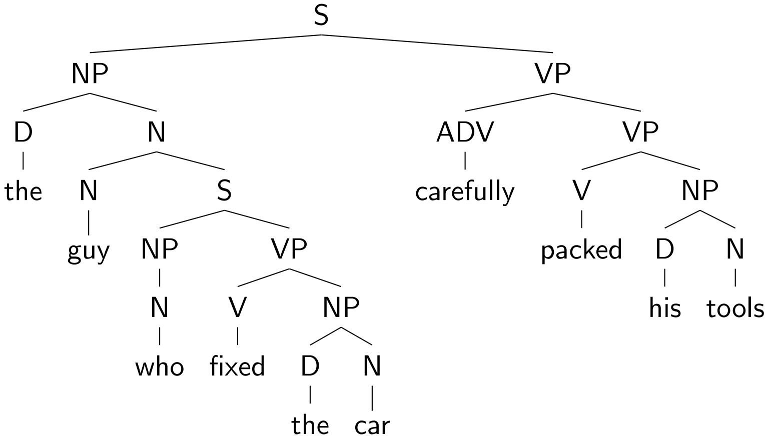
Noam Chomsky: Language, Cognition, and Deep Learning

www.youtube.com/watch?v=cMscNuSUy0I

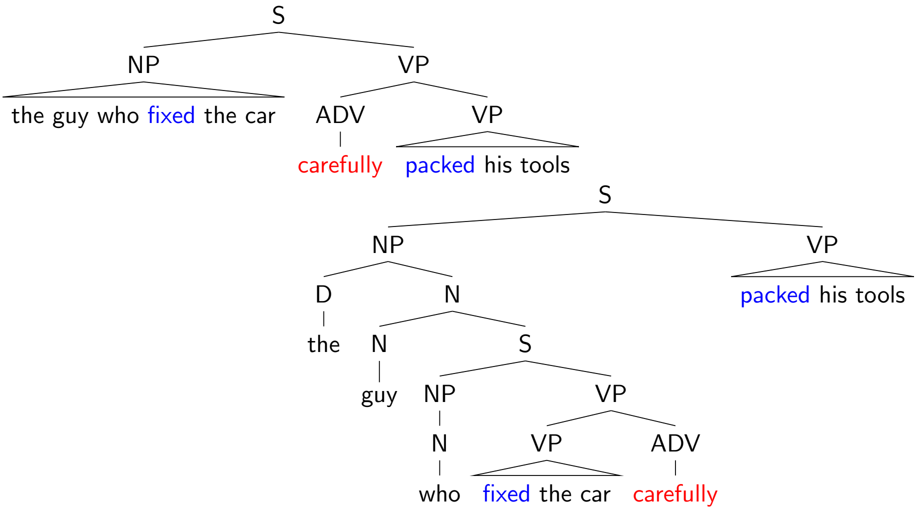
Constituency (phrase structure)

The basic idea

Phrase structure organizes words into *nested* *constituents*, which can be represented as **a tree**.

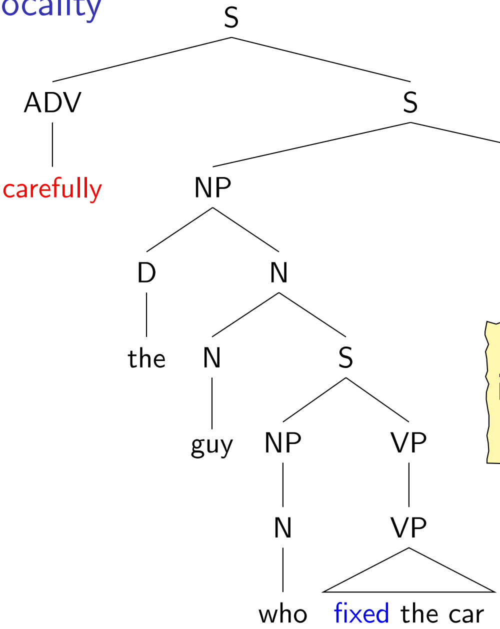


Different structures, different meaning



Results by a cool parser: `http://erg.delph-in.net/logon`

Locality



carefully
is hierarchically closer to
packed / *is tall*

Applications of parsing

Modern parsers are quite accurate

For some languages, automatic syntactic parsing is good enough to help a range of NLP tasks!

- Machine translation
- Information extraction
- Grammar checking
- etc.

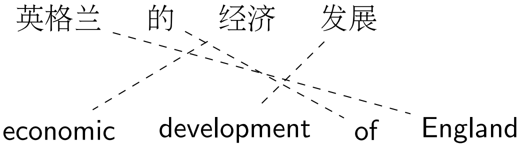
Applications of parsing

Modern parsers are quite accurate

For some languages, automatic syntactic parsing is good enough to help a range of NLP tasks!

- Machine translation
- Information extraction
- Grammar checking
- etc.

Translate “英格兰的经济发展” into English



Applications of parsing

Modern parsers are quite accurate

For some languages, automatic syntactic parsing is good enough to help a range of NLP tasks!

- Machine translation
- Information extraction
- Grammar checking
- etc.

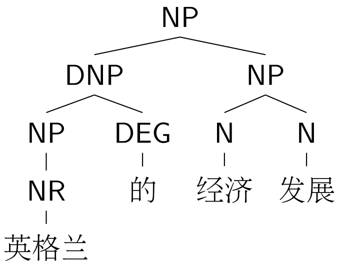
Translate "英格兰的经济发展" into English

英格兰 的 经济 发展

economic development of England

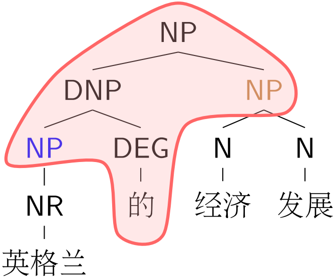
finite-state transducer?

An example: Machine translation



1	NP NR 英格兰	England
2	NP DNP NP NP DEG 的	NP of NP
3	NP N N	N N
4	N 经济	economic
5	N 发展	development

An example: Machine translation

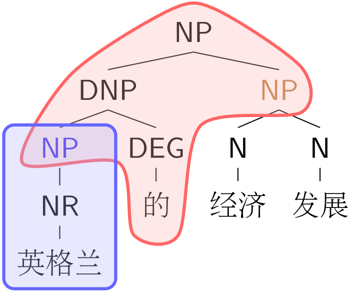


⇒NP of NP

▷R2

1	NP – NR – 英格兰	England
2	NP → DNP NP; DNP → NP DEG; DEG – 的	NP of NP
3	NP → N N	N N
4	N – 经济	economic
5	N – 发展	development

An example: Machine translation

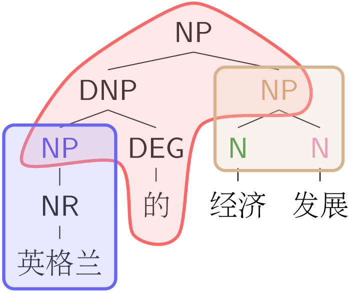


$\Rightarrow$NP of NP ▷R2

$\Rightarrow$NP of England ▷R1

1	NP – NR – 英格兰	England
2	NP → DNP NP; DNP → NP DEG; DEG – 的	NP of NP
3	NP → N N	N N
4	N – 经济	economic
5	N – 发展	development

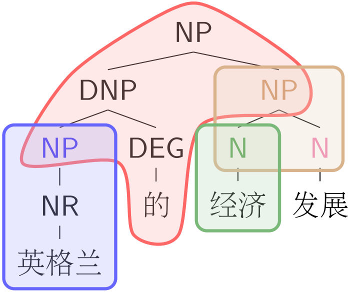
An example: Machine translation



$\Rightarrow$NP of NP ▷R2
$\Rightarrow$NP of England ▷R1
$\Rightarrow$N N of England ▷R3

1	NP – NR – 英格兰	England
2	NP (DNP (NP DEG – 的) NP)	NP of NP
3	NP (N N)	N N
4	N – 经济	economic
5	N – 发展	development

An example: Machine translation



1	NP NR 英格兰	England
2	NP DNP NP NP DEG 的	NP of NP
3	NP N N	N N
4	N 经济	economic
5	N 发展	development

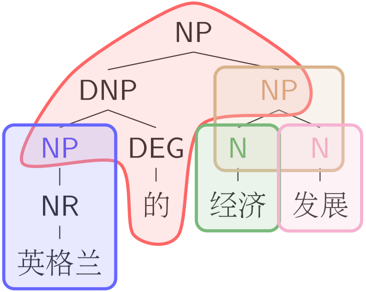
⇒NP of NP ▷R2

⇒NP of England ▷R1

⇒N N of England ▷R3

⇒economic N of England ▷R4

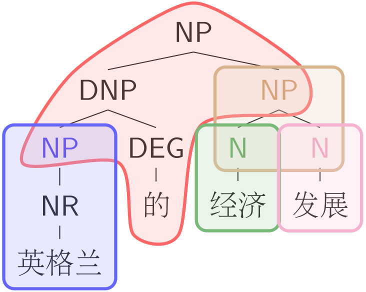
An example: Machine translation



⇒NP of NP ▷R2
⇒NP of England ▷R1
⇒N N of England ▷R3
⇒economic N of England ▷R4
⇒economic development of England ▷R5

1	NP NR 英格兰	England
2	NP DNP NP NP DEG 的	NP of NP
3	NP N N	N N
4	N 经济	economic
5	N 发展	development

An example: Machine translation



⇒NP of NP ▷R2

⇒NP of England ▷R1

⇒N N of England ▷R3

⇒economic N of England ▷R4

⇒economic development of England ▷R5

recursive form transformation

1	NP NR 英格兰	England
2	NP DNP NP NP DEG 的	NP of NP
3	NP N N	N N
4	N 经济	economic
5	N 发展	development

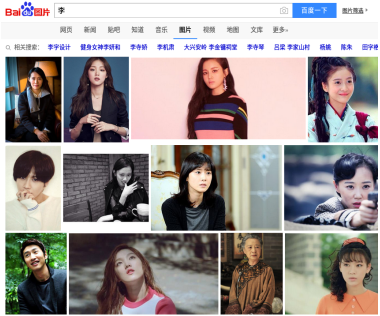
Structured Prediction

watch this video

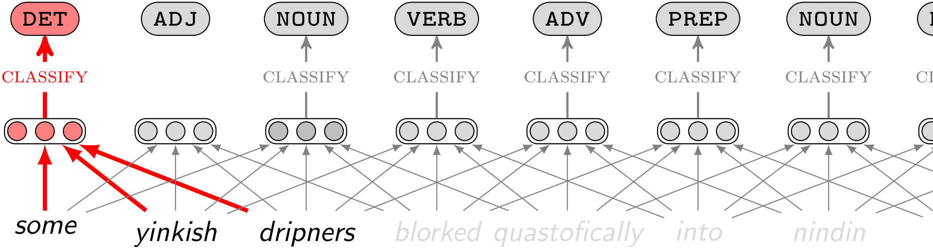
www.youtube.com/watch?v=bjUwSHGsG9o

Muhammad Li

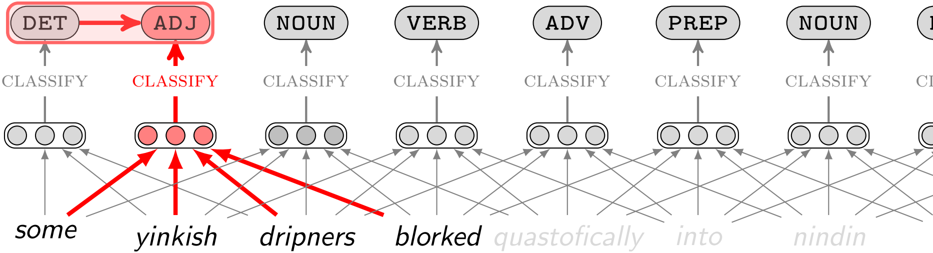
Howard	Who's Muhammad Li?
Sheldon	Muhammad is the most common first name in the world, Li, the most common surname. As I didn't know the answer, I thought that gave me a mathematical edge.



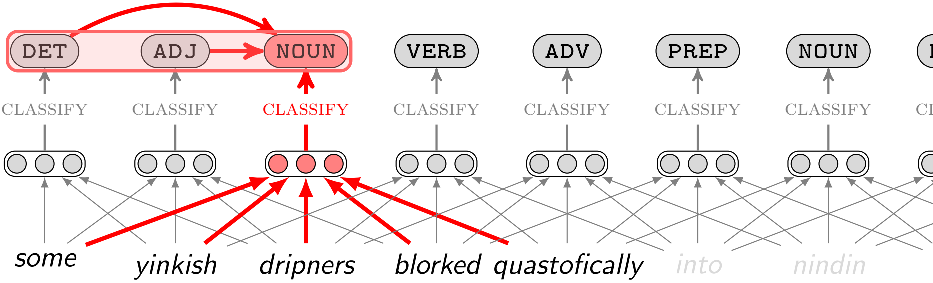
POS tagging and prediction



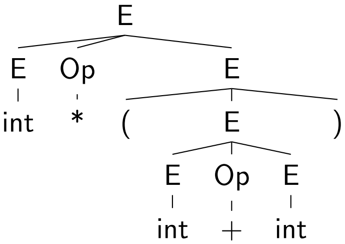
POS tagging and prediction

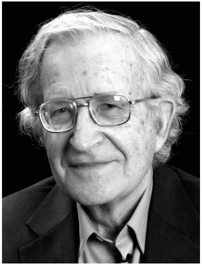


POS tagging and prediction



Two perspectives ≈ Possible vs Probable





[...] Therefore the true logic for this world is the calculus of probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man's mind.

Linguistic structure prediction

As a structured prediction problem

- Search space: Is this analysis possible? ▷CFG (today)
- Measurement: Is this analysis *good*? ▷PCFG (today)

$$\boldsymbol{y}^*(\boldsymbol{x}; \boxed{\mathbf{w}}) = \boxed{\arg\max}_{\boxed{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}} \boxed{\textsc{Score}(\boldsymbol{x}, \boldsymbol{y})}$$

- Decode: find the analysis that obtains the highest score ▷L95
- Parameter estimation: find good parameters ▷L101

Linguistic structure prediction

As a structured prediction problem

- Search space: Is this analysis possible? ▷CFG (today)
- Measurement: Is this analysis *good*? ▷PCFG (today)

$$\boldsymbol{y}^*(\boldsymbol{x}; \mathbf{w}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \textsc{Score}(\boldsymbol{x}, \boldsymbol{y})$$

- Decode: find the analysis that obtains the highest score ▷L95
- Parameter estimation: find good parameters ▷L101

Linguistic structure prediction

As a structured prediction problem

- Search space: Is this analysis possible? ▷CFG (today)
- Measurement: Is this analysis *good*? ▷PCFG (today)

$$\boldsymbol{y}^*(\boldsymbol{x}; \mathbf{w}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \textsc{Score}(\boldsymbol{x}, \boldsymbol{y})$$

- Decode: find the analysis that obtains the highest score ▷L95
- Parameter estimation: find good parameters ▷L101

Linguistic structure prediction

As a structured prediction problem

- Search space: Is this analysis possible? ▷CFG (today)
- Measurement: Is this analysis *good*? ▷PCFG (today)

$$\boldsymbol{y}^*(\boldsymbol{x}; \boxed{\mathbf{w}}) = \boxed{\arg\max}_{\boxed{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}} \boxed{\textsc{Score}(\boldsymbol{x}, \boldsymbol{y})}$$

- Decode: find the analysis that obtains the highest score ▷L95
- Parameter estimation: find good parameters ▷L101

Linguistic structure prediction

As a structured prediction problem

- Search space: Is this analysis possible? ▷CFG (today)
- Measurement: Is this analysis *good*? ▷PCFG (today)

$$\boldsymbol{y}^*(\boldsymbol{x}; \mathbf{w}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \textsc{Score}(\boldsymbol{x}, \boldsymbol{y})$$

- Decode: find the analysis that obtains the highest score ▷L95
- Parameter estimation: find good parameters ▷L101

Context-Free Grammar

A formal language is a set of strings over an alphabet

Strings and languages

- A string of length n over an alphabet Σ is an ordered n-tuple of elements of Σ.
- Σ^* denotes the set of all strings over Σ of finite length.
- Given an alphabet Σ any subset of Σ^* is a formal language over alphabet Σ.

Language models

- Define a particular subset of strings $S \subseteq \Sigma^*$ with finite *rules*.

Regular Expression

$$\mathtt{a}^+(\mathtt{.a}^+)^*\mathtt{@aa}^*\mathtt{.aa}^+$$

Formal grammars

Formally specify a grammar that can generate all and only the acceptable sentences of a natural language. Generative Grammar

Formal grammars

Formally specify a grammar that can generate all and only the acceptable sentences of a natural language. Generative Grammar

A grammar G consists of the following components:

1. A finite set Σ of terminal symbols.
2. A finite set N of nonterminal symbols that is disjoint from Σ.
3. A distinguished nonterminal symbol that is the $\textsc{start}$ symbol.
4. A finite set R of production rules, each rule of the form

   $$(\Sigma \cup N)^+ \rightarrow (\Sigma \cup N)^*$$

   Each production rule maps from one string of symbols to another.

Formal grammars

Formally specify a grammar that can generate all and only the acceptable sentences of a natural language. Generative Grammar

A grammar G consists of the following components:

1. A finite set Σ of terminal symbols.
2. A finite set N of nonterminal symbols that is disjoint from Σ.
3. A distinguished nonterminal symbol that is the $\textsc{start}$ symbol.
4. A finite set R of production rules, each rule of the form

$$(\Sigma \cup N)^{+} \rightarrow (\Sigma \cup N)^{*}$$

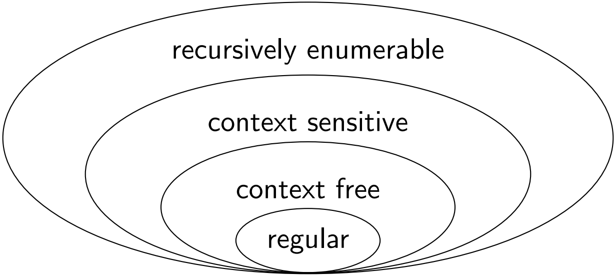
   Each production rule maps from one string of symbols to another.

Weak equivalence grammars generate the same strings
Strong equivalence grammars generate the same strings with same internal structures

Chomsky Hierarchy

Grammar	Languages	Production rules
Type-0	Recursively enumerable	$\alpha \rightarrow \gamma$
Type-1	Context-sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	$A \rightarrow \gamma$
Type-3	Regular	$A \rightarrow a$
		$A \rightarrow aB$

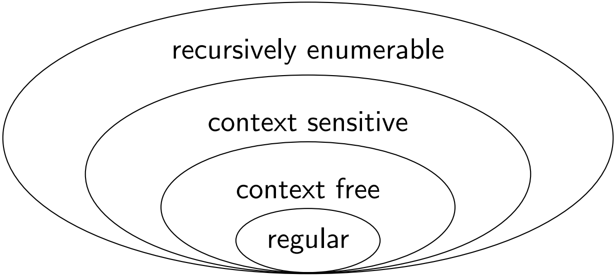
$a \in N;\ \alpha, \beta, \gamma \in (N \cup \Sigma)^*$



Chomsky Hierarchy

Grammar	Languages	Production rules
Type-0	Recursively enumerable	$\alpha \rightarrow \gamma$
Type-1	Context-sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	$A \rightarrow \gamma$
Type-3	Regular	$A \rightarrow a$ $A \rightarrow aB$

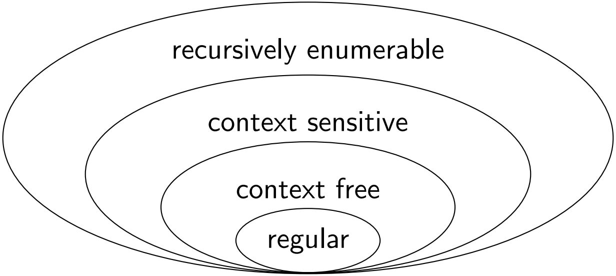
$a \in N;\ \alpha, \beta, \gamma \in (N \cup \Sigma)^*$



Chomsky Hierarchy

Grammar	Languages	Production rules
Type-0	Recursively enumerable	$\alpha \rightarrow \gamma$
Type-1	Context-sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	$A \rightarrow \gamma$
Type-3	Regular	$A \rightarrow a$ $A \rightarrow aB$

$a \in N$; $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$



Context-Free Grammars

1. N: variables
2. Σ: terminals
3. R: productions

$$A \to (N \cup \Sigma)^*$$

$$A \in N$$

4. S: START

Question

What does *context-free* mean?

An example

- $L = \{a^n b^n | n \in \mathbb{N}\}$
- S$\rightarrow$aSb$|\epsilon$

An example

S

- $L = \{a^n b^n | n \in \mathbb{N}\}$
- S$\rightarrow$aSb$|\epsilon$

An example

- $L = \{a^n b^n | n \in \mathbb{N}\}$
- $\mathsf{S} \rightarrow \mathsf{aSb} | \epsilon$

$$
\begin{aligned}
& \mathsf{S} \\
\Rightarrow\ & \mathsf{aSb}
\end{aligned}
$$

An example

- $L = \{a^n b^n | n \in \mathbb{N}\}$
- S→aSb|ϵ

$$\begin{aligned} & \mathsf{S} \\ \Rightarrow\ & \mathsf{aSb} \\ \Rightarrow\ & \mathsf{aaSbb} \end{aligned}$$

An example

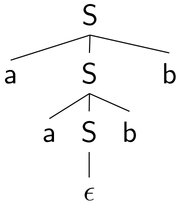
- $L = \{a^n b^n | n \in \mathbb{N}\}$
- S→aSb|ϵ

$$
\begin{aligned}
& \mathsf{S} \\
\Rightarrow\ & \mathsf{aSb} \\
\Rightarrow\ & \mathsf{aaSbb} \\
\Rightarrow\ & \mathsf{aa}\epsilon\mathsf{bb}
\end{aligned}
$$

An example

- $L = \{a^n b^n | n \in \mathbb{N}\}$
- S→aSb|ϵ

$$
\begin{aligned}
& \mathsf{S} \\
\Rightarrow\ & \mathsf{aSb} \\
\Rightarrow\ & \mathsf{aaSbb} \\
\Rightarrow\ & \mathsf{aa}\epsilon\mathsf{bb}
\end{aligned}
$$



A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

S ⇒ NP VP

A linguistic example (1)

$N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}, \text{Adj}, \text{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \text{S}$

We can derive the structure of a string.

S ⇒ NP VP
⇒ N VP

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

S ⇒ NP VP
⇒ N VP
⇒ ideas VP

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

$$
\begin{aligned}
\mathsf{S} &\Rightarrow \mathsf{NP\ VP} \\
&\Rightarrow \mathsf{N\ VP} \\
&\Rightarrow \mathsf{ideas\ VP} \\
&\Rightarrow \mathsf{ideas\ VP\ AdvP}
\end{aligned}
$$

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

S ⇒ NP VP
⇒ N VP
⇒ ideas VP
⇒ ideas VP AdvP
⇒ ideas V AdvP

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

S ⇒ NP VP
⇒ N VP
⇒ ideas VP
⇒ ideas VP AdvP
⇒ ideas V AdvP
⇒ ideas sleep AdvP

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

S ⇒ NP VP
⇒ N VP
⇒ ideas VP
⇒ ideas VP AdvP
⇒ ideas V AdvP
⇒ ideas sleep AdvP
⇒ ideas sleep Adv

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj
Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

We can derive the structure of a string.

S ⇒ NP VP
⇒ N VP
⇒ ideas VP
⇒ ideas VP AdvP
⇒ ideas V AdvP
⇒ ideas sleep AdvP
⇒ ideas sleep Adv
⇒ ideas sleep furiously

A linguistic example (1)

$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{AdjP}, \mathsf{AdvP}\} \cup \{\mathsf{N}, \mathsf{Adj}, \mathsf{Adv}\}$

$\Sigma = \{$*colorless*, *green*, *ideas*, *sleep*, *furiously*$\}$

R

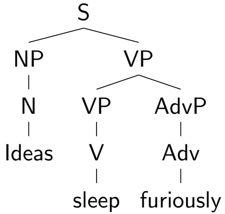
S→NP VP	NP→AdjP NP
VP→VP AdvP	
VP→V	NP→N
AdvP→Adv	AdjP→Adj

Adj→*colorless*	Adj→*green*
N→*ideas*	V→*sleep*
Adv→*furiously*	

$S = \mathsf{S}$

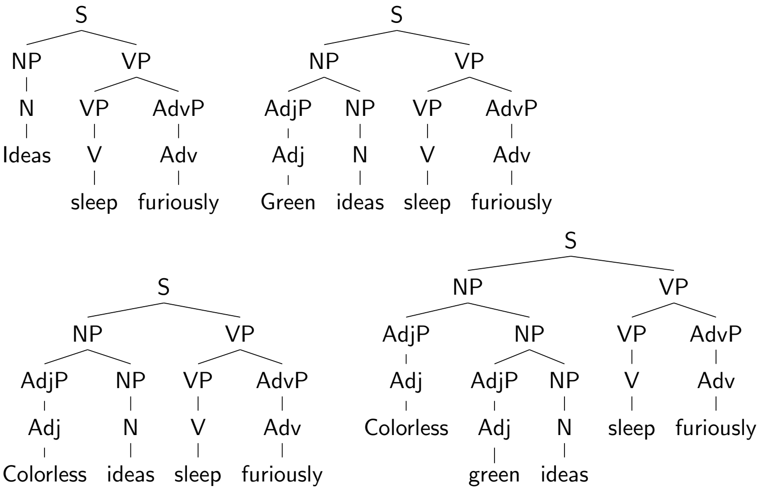
We can derive the structure of a string.

$$
\begin{aligned}
\mathsf{S} &\Rightarrow \mathsf{NP\ VP} \\
&\Rightarrow \mathsf{N\ VP} \\
&\Rightarrow \mathsf{ideas\ VP} \\
&\Rightarrow \mathsf{ideas\ VP\ AdvP} \\
&\Rightarrow \mathsf{ideas\ V\ AdvP} \\
&\Rightarrow \mathsf{ideas\ sleep\ AdvP} \\
&\Rightarrow \mathsf{ideas\ sleep\ Adv} \\
&\Rightarrow \mathsf{ideas\ sleep\ furiously}
\end{aligned}
$$



A linguistic example (2)

We can define the language of a grammar by applying the productions.



Recursion (1)

recursion

place one component inside another component of the same type

Recursion (1)

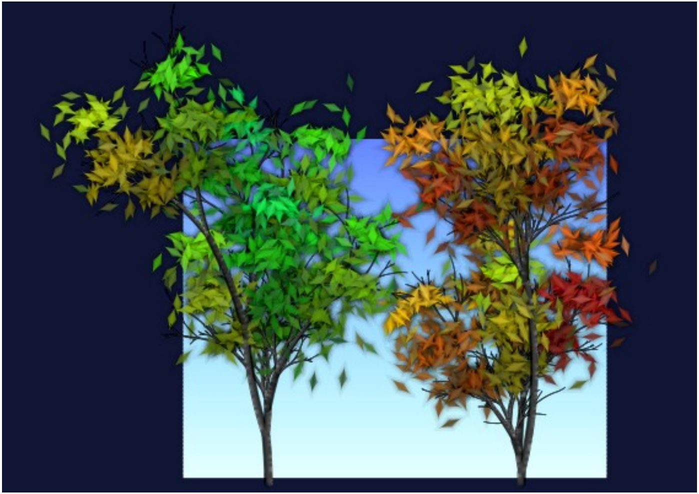
Natural numbers

- $0 \leftarrow \emptyset$
- If n is a natural number, let $n + 1 \leftarrow n \cup \{n\}$

$$0 = \emptyset$$
$$1 = \{0\} = \{\emptyset\}$$
$$2 = \{0, 1\} = \{\emptyset, \{\emptyset\}\}$$
$$3 = \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$$

recursion
place one component inside another component of the same type

Recursion (2)



www.contextfreeart.org

Recursion (3)

> *We hypothesize that FLN (faculty of language in the narrow sense) only includes recursion and is the only uniquely human component of the faculty of language.*
>
> *M Hauser, N Chomsky and W Fitch (2002)*
>
> *science.sciencemag.org/content/298/5598/1569*

(2) a. The dog bit the cat [which chased the mouse [which died]]. (right)

b. [[the dog] 's owner] 's friend (left)

c. The mouse [the cat [the dog bit] chased] died. (center)

Where can I get a grammar?

English Treebank

- Penn Treebank = ca. 50,000 sentences with associated trees
- Usual set-up: ca. 40,000 training sentences, ca. 2,400 test sentences

Probabilistic Context-Free Grammars

Probabilistic CFGs

Probability of a tree t with rules $A_1 \to \beta_1, A_2 \to \beta_2, \ldots$ is

$$p(t) = \prod_{i=1}^{n} q(A_i \to \beta_i)$$

where $q(A_i \to \beta_i)$ is the probability for rule $A_i \to \beta_i$.

- When we expand A_i, how likely is it that we choose $A_i \to \beta_i$?
- For each nonterminal A_i,

$$\sum_{\beta} q(A \to \beta | A) = 1$$

- PCFG generates random derivations of CFG.
- Each event (expanding nonterminal by production rules) is statistically independent of all the others.

An example (1)

S	$\rightarrow$	NP VP	0.8
S	$\rightarrow$	Aux NP VP	0.15
S	$\rightarrow$	VP	0.05
NP	$\rightarrow$	AdjP NP	0.2
NP	$\rightarrow$	D N	0.7
NP	$\rightarrow$	N	0.1
VP	$\rightarrow$	VP AdvP	0.3
VP	$\rightarrow$	V	0.2
VP	$\rightarrow$	V NP	0.3
VP	$\rightarrow$	V NP NP	0.2
AdvP	$\rightarrow$	Adv	1.0
AdjP	$\rightarrow$	Adj	1.0

Adj	$\rightarrow$	*colorless*	0.4
Adj	$\rightarrow$	*green*	0.6
N	$\rightarrow$	*ideas*	1.0
V	$\rightarrow$	*sleep*	1.0
Adv	$\rightarrow$	*furiously*	1.0

An example (2)

S	S→NP VP	0.8

An example (2)

	S	S→NP VP	0.8
$\Rightarrow$	NP VP	NP→N	0.1

An example (2)

	S	S→NP VP	0.8
⇒	NP VP	NP→N	0.1
⇒	N VP	N→*ideas*	1.0

An example (2)

	S	S→NP VP	0.8
⇒	NP VP	NP→N	0.1
⇒	N VP	N→*ideas*	1.0
⇒	ideas VP	VP→VP AdvP	0.3

An example (2)

	S	S→NP VP	0.8
⇒	NP VP	NP→N	0.1
⇒	N VP	N→*ideas*	1.0
⇒	ideas VP	VP→VP AdvP	0.3
⇒	ideas VP AdvP	VP→V	0.2

An example (2)

	S	S→NP VP	0.8
⇒	NP VP	NP→N	0.1
⇒	N VP	N→*ideas*	1.0
⇒	ideas VP	VP→VP AdvP	0.3
⇒	ideas VP AdvP	VP→V	0.2
⇒	ideas V AdvP	V→*sleep*	1.0

An example (2)

	S	S→NP VP	0.8
⇒	NP VP	NP→N	0.1
⇒	N VP	N→*ideas*	1.0
⇒	ideas VP	VP→VP AdvP	0.3
⇒	ideas VP AdvP	VP→V	0.2
⇒	ideas V AdvP	V→*sleep*	1.0
⇒	ideas sleep AdvP	AdvP→Adv	1.0

An example (2)

	S	S→NP VP	0.8
⇒	NP VP	NP→N	0.1
⇒	N VP	N→*ideas*	1.0
⇒	ideas VP	VP→VP AdvP	0.3
⇒	ideas VP AdvP	VP→V	0.2
⇒	ideas V AdvP	V→*sleep*	1.0
⇒	ideas sleep AdvP	AdvP→Adv	1.0
⇒	ideas sleep Adv	Adv→*furiously*	1.0

An example (2)

	S	S→NP VP	0.8
⇒	NP VP	NP→N	0.1
⇒	N VP	N→*ideas*	1.0
⇒	ideas VP	VP→VP AdvP	0.3
⇒	ideas VP AdvP	VP→V	0.2
⇒	ideas V AdvP	V→*sleep*	1.0
⇒	ideas sleep AdvP	AdvP→Adv	1.0
⇒	ideas sleep Adv	Adv→*furiously*	1.0

$$0.8 \times 0.1 \times 1.0 \times 0.3 \times 0.2 \times 1.0 \times 1.0 \times 1.0$$

Properties of PCFGs

- Assigns a probability to each parse-tree, allowed by the underlying CFG
- Say we have a sentence s, set of derivations for that sentence is $\mathcal{T}(s)$, as defined by a CFG. Then a PCFG assigns a probability $p(t)$ to each member of $\mathcal{T}(s)$.

- We now have a $\textsc{Score}$ function (probability) that can ranks trees.
- The most likely parse tree for a sentence s is

$$\boxed{\arg\max_{t \in \mathcal{T}(s)} p(t)}$$

"correct" means more probable parse tree

"language" means set of grammatical sentences

Deriving a PCFG from a Treebank

Given a set of example trees (a treebank), the underlying CFG can simply be all rules seen in the corpus

Maximum Likelihood Estimates

$$q_{ML}(\alpha \rightarrow \beta) = \frac{\text{COUNT}(\alpha \rightarrow \beta)}{\text{COUNT}(\alpha)}$$

The counts are taken from a training set of example trees.

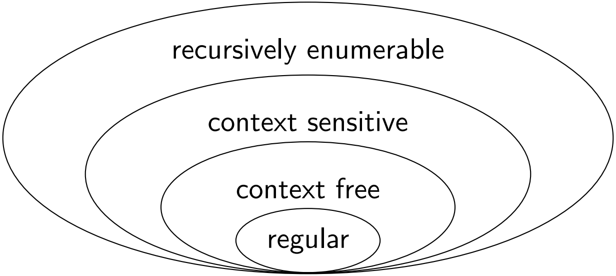
If the training data is generated by a PCFG, then as the training data size goes to infinity, the maximum-likelihood PCFG will converge to the same distribution as the "true" PCFG.

Rethink Part-of-Speech Tagging

Chomsky Hierarchy

Grammar	Languages	Production rules
Type-0	Recursively enumerable	$\alpha \rightarrow \gamma$
Type-1	Context-sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	$A \rightarrow \gamma$
Type-3	Regular	$A \rightarrow a$
		$A \rightarrow aB$

$a \in N;\ \alpha, \beta, \gamma \in (N \cup \Sigma)^*$



An example

Max bitted the cat [which chased the mouse [which died]].

A toy grammar

- VP$\rightarrow$ bitted|chased|... DP
- VP$\rightarrow$ died
- DP$\rightarrow$ the|a|this|... NP
- NP$\rightarrow$ dog|cat|mouse|... RC
- RC$\rightarrow$ which|that|... VP

An example

Max bitted the cat [which chased the mouse [which died]].

A toy grammar

- VP$\rightarrow$ bitted|chased|... DP
- VP$\rightarrow$ died
- DP$\rightarrow$ the|a|this|... NP
- NP$\rightarrow$ dog|cat|mouse|... RC
- RC$\rightarrow$ which|that|... VP

<u>VP</u>

An example

Max bitted the cat [which chased the mouse [which died]].

A toy grammar

- VP$\rightarrow$ bitted|chased|... DP
- VP$\rightarrow$ died
- DP$\rightarrow$ the|a|this|... NP
- NP$\rightarrow$ dog|cat|mouse|... RC
- RC$\rightarrow$ which|that|... VP

<u>VP</u>
$\Rightarrow$ *bit <u>DP</u>*

An example

Max bitted the cat [which chased the mouse [which died]].

A toy grammar

- VP$\rightarrow$ bitted|chased|... DP
- VP$\rightarrow$ died
- DP$\rightarrow$ the|a|this|... NP
- NP$\rightarrow$ dog|cat|mouse|... RC
- RC$\rightarrow$ which|that|... VP

VP
$\Rightarrow$ *bit DP* $\Rightarrow$ bit *the NP*

An example

Max bitted the cat [which chased the mouse [which died]].

A toy grammar

- VP$\rightarrow$ bitted|chased|... DP
- VP$\rightarrow$ died
- DP$\rightarrow$ the|a|this|... NP
- NP$\rightarrow$ dog|cat|mouse|... RC
- RC$\rightarrow$ which|that|... VP

<u>VP</u>
$\Rightarrow$ *bit <u>DP</u>* $\Rightarrow$ bit *the <u>NP</u>* $\Rightarrow$ bit the *cat <u>RC</u>*

An example

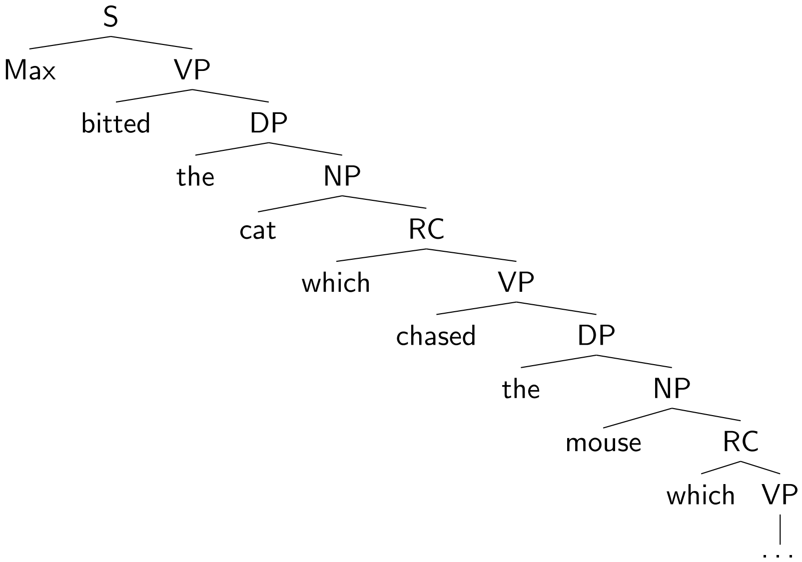
Max bitted the cat [which chased the mouse [which died]].

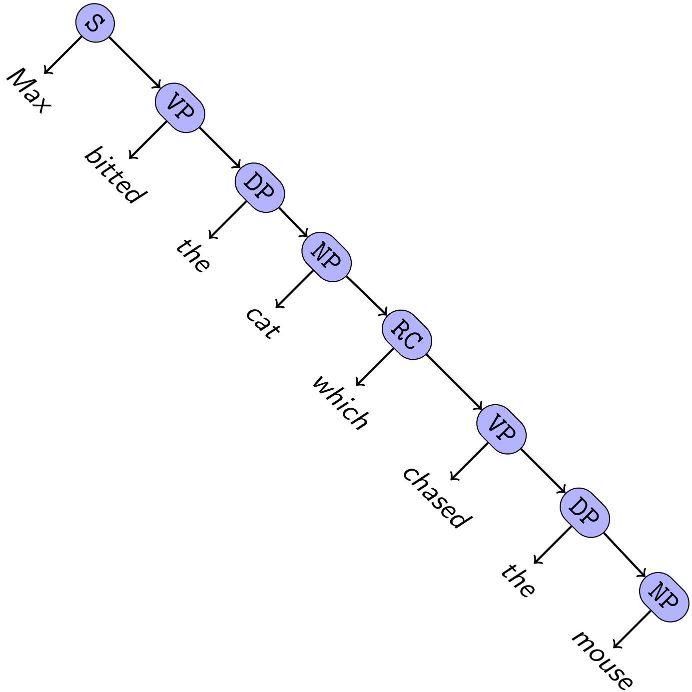
A toy grammar

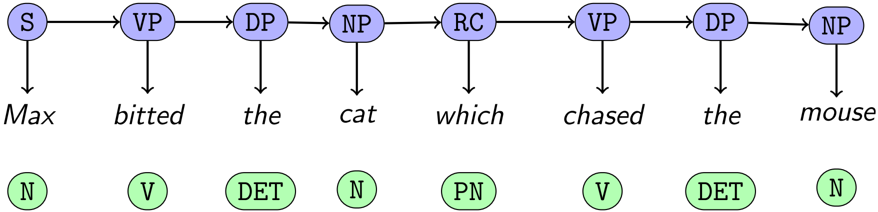
- VP$\rightarrow$ bitted|chased|... DP
- VP$\rightarrow$ died
- DP$\rightarrow$ the|a|this|... NP
- NP$\rightarrow$ dog|cat|mouse|... RC
- RC$\rightarrow$ which|that|... VP

<u>VP</u>
$\Rightarrow$ *bit* <u>*DP*</u> $\Rightarrow$ bit *the* <u>*NP*</u> $\Rightarrow$ bit the *cat* <u>*RC*</u>
$\Rightarrow$ bit the cat *which* <u>*VP*</u>

Finite state machines?




S
Max
VP
bitted
DP
the
NP
cat
RC
which
VP
chased
DP
the
NP
mouse



word tagging is very powerful

Harmonic word order

Morphology

- Postpositional and head-final languages use suffixes and no prefixes.
- Prepositional and head-initial languages use not only prefixes but also suffixes.

Greenberg's word order universals

- Universal 3: Languages with dominant VSO order are always prepositional.
- Universal 4: With overwhelmingly greater than chance frequency, languages with normal SOV order are postpositional.
- Universal 5: If a language has dominant SOV order and the genitive follows the governing noun, then the adjective likewise follows the noun.
- Universal 17: With overwhelmingly more than chance frequency, languages with dominant order VSO have the adjective after the noun.

Empirical data can be found at `https://wals.info`.

Challenge

Cross-serial dependencies in Dutch

... dat Wim Jan Marie de kinderen zag helpen leren zwemmen
... that Wim Jan Marie the children saw help teach swim
... that Wim saw Jan help Marie teach the children to swim

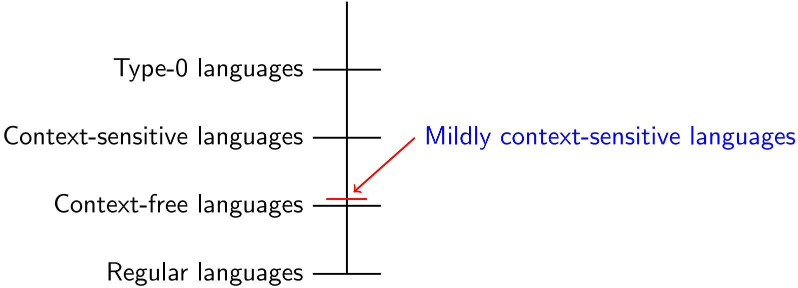
Cross-serial dependencies in Swiss German

... das mer em Hans es huus hälfed aastriiche
... that we Hans$_{Dat}$ house$_{Acc}$ helped paint
... that we helped Hans paint the house

... das mer d'chind em Hans es huus lönd hälfe aastriiche
... that we the children$_{Acc}$ Hans$_{Dat}$ house$_{Acc}$ let help paint
... that we let the children help Hans paint the house

Mildly Context-Sensitive Languages

Natural languages are provably non-context-free.
Natural languages = mildly context-sensitive languages?



Reading

- Ann's lecture notes.
  `https://www.cl.cam.ac.uk/teaching/1920/NLP/materials.html`
- D Jurafsky and J Martin. *Speech and Language Processing*.
  - Chapter 12. Constituency Grammars.
    `https://web.stanford.edu/~jurafsky/slp3/12.pdf`
  - Chapter 14. Statistical Constituency Parsing.
    `https://web.stanford.edu/~jurafsky/slp3/14.pdf`