Introduction to Natural Language Processing

Part VII: NLP using Context-Free Grammars

Henning Wachsmuth

https://ai.uni-hannover.de

Learning Objectives

Concepts

- Hierarchical patterns in language
- Probabilitic extensions of grammars
- Use of context-free grammars in NLP

Methods

- Conversion of context-free grammars into Chomsky Normal Form
- Syntact parsing of sentences with the extended CKY algorithm
- Extensions and variations of syntactic parsing

Covered tasks

- Constituency parsing
- Dependency parsing

Outline of the Course

Introduction

Grammar

Formal grammars (recap)

- A grammar is a description of the valid structures of a language.
- A formal grammar specifies a set of rules consisting of terminal and non-terminal symbols.

Grammar (Σ, N, S, R)

Σ An alphabet, i.e., a finite set of terminal symbols, $\Sigma = \{v_1, v_2, \ldots\}$

N A finite set of non-terminal symbols, $N = \{W_1, W_2, \ldots\}$

S A start non-terminal symbol, $S \in N$

R A finite set of production rules, $R \subseteq (\Sigma \cup N)^+ \setminus \Sigma^* \times (\Sigma \cup N)^*$

Context-free grammar (CFG)

- A grammar (Σ, N, S, R) is *context-free* if all rules in R are of the form $U \rightarrow V$ with $U \in N$ and $V \in (N \cup \Sigma)^*$.
- A language is context-free, if there is a CFG that defines it.

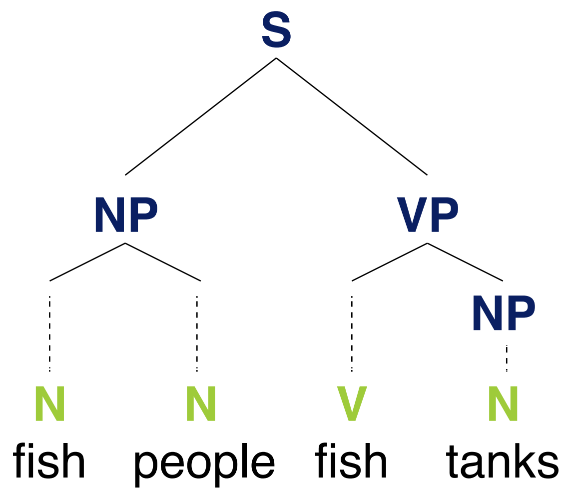
NLP using Context-Free Grammars

Use of CFGs in NLP

- CFGs tend to be effective for hierarchical structures of language.
- Probabilistic extensions (PCFGs) capture the likeliness of structures.
- CFGs usually define the basis of syntactic parsing.

Syntactic parsing (aka full parsing)

- The text analysis that determines the syntactic structure of a sentence
- Used in NLP as preprocessing for many tasks, e.g., relation extraction



Constituencies vs. dependencies

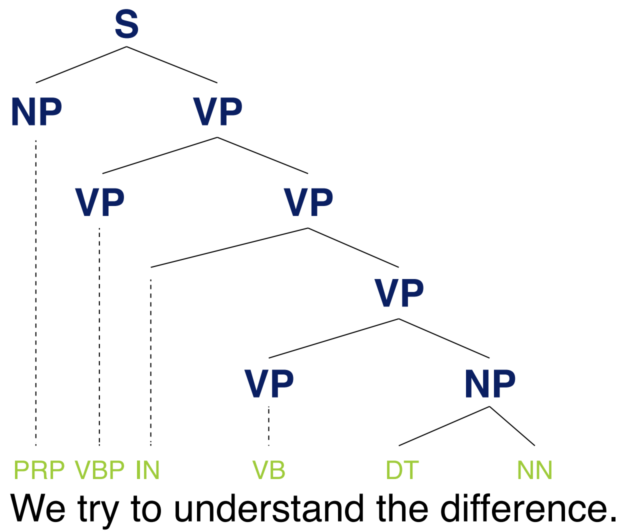
- Constituency parsing. Infers the structure of the phrases in a sentence
- Dependency parsing. Infers the structure of the words' dependencies

NLP using Context-Free Grammars

Phrase vs. Dependency Structure (Recap)

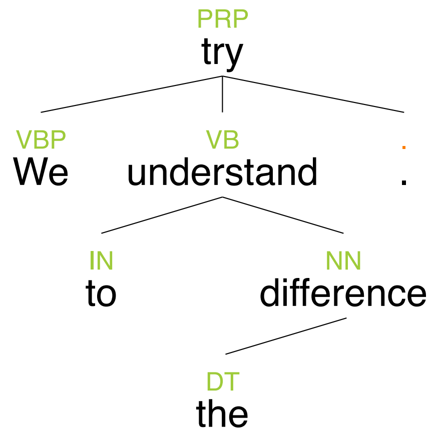
Phrase structure grammar

- Models the constituents of a sentence and how they are composed of other constituents and words
- Constituency (parse) tree. Inner nodes are non-terminals, leafs are terminals



Dependency grammar

- Models the dependencies between the words in a sentence
- Dependency (parse) tree. All nodes are terminals, the root is nearly always the main verb (of the first main clause).



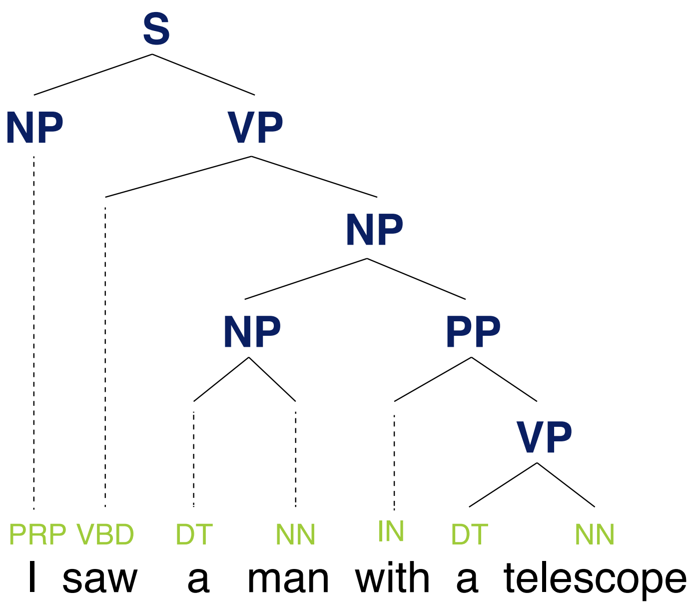
NLP using Context-Free Grammars

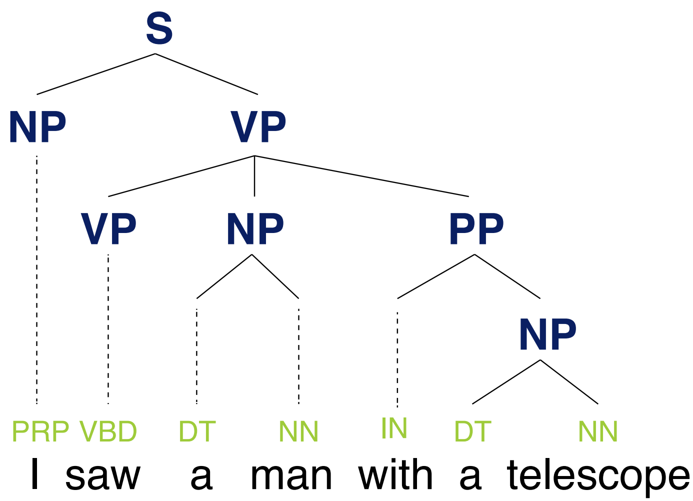
Attachment Ambiguity

Example “I saw a man with a telescope”

https://pxfuel.com

https://pxfuel.com





Probabilistic Context-Free Grammars

Context-Free Grammars

Context-free gammars (CFGs) in NLP

- CFGs are particularly used to model the *phrase structure* of sentences.
- A phrase structure grammar is just a CFG with a specific interpretation.
  We will mostly simply speak of CFGs here.
- For NLP, CFGs are extended by probabilities, as we will see below.

Phrase structure interpretation of non-terminals $N = N_{phr} \cup N_{pos}$

N_{phr} Phrase types. A finite set of structural non-terminal symbols

N_{pos} Part-of-speech tags. A finite set of lexical “pre-terminal” symbols

$N_{phr} \cap N_{pos} = \emptyset$.

Phrase structure interpretation of rules $R = R_{phr} \cup R_{pos}$

R_{phr} A finite set of structure rules of the form $U \rightarrow V$ with $U \in N_{phr}$ and $V \in (N_{phr} \cup N_{pos})^*$

R_{pos} A finite set of lexicon rules of the form $U \rightarrow v$ with $U \in N_{pos}$ and $v \in \Sigma$

In addition to S, CFGs in NLP usually include an extra symbol ROOT.

Context-Free Grammars

Example

Example CFG, represented by its rules

Structural rules			Lexical rules	
s1	S $\rightarrow$ NP VP		l1	N $\rightarrow$ people
s2	VP $\rightarrow$ V NP		l2	N $\rightarrow$ fish
s3	VP $\rightarrow$ V NP PP		l3	N $\rightarrow$ tanks
s4	NP $\rightarrow$ NP NP		l4	N $\rightarrow$ rods
s5	NP $\rightarrow$ NP PP	// binary	l5	V $\rightarrow$ people
s6	NP $\rightarrow$ N	// unary	l6	V $\rightarrow$ fish
s7	NP $\rightarrow$ ε	// empty	l7	V $\rightarrow$ tanks
s8	PP $\rightarrow$ P NP		l8	P $\rightarrow$ with

Example sentences created by the grammar

"people fish tanks"

"people fish with rods"

Chomsky Normal Form (CNF)

- A CFG is in Chomsky Normal Form if all rules in R are of the forms $U \rightarrow VW$ and $U \rightarrow v$, where $U, V, W \in N$ and $v \in \Sigma^*$.

Tansformation into normal form

- Cleaning. Empties and unaries are removed recursively.
- Binarization. n-ary rules are divided by using new non-terminals, $n > 2$.

- Any CFG can be transformed into CNF without changing the language.
- This may result in different parse trees for words the language.

Why transforming?

- Restricting a CFG in such a way is key to efficient parsing.
- Binarization is crucial for cubic time.
- Cleaning is not mandatory, but makes parsing quicker and cleaner.

  More on this further below.

Chomsky Normal Form

Pseudocode

Signature

- Input. The production rules R of a CFG
- Output. The production rules R^* of the normalized version of the CFG

toChomskyNormalForm(Production rules R)

while an empty $(U \rightarrow \varepsilon) \in R$ **do**
    $R \leftarrow R \setminus \{U \rightarrow \varepsilon\}$
    for each rule $(V \rightarrow V_1 \ldots V_k\, U\, W_1 \ldots W_l) \in R$ **do** // $k, l \geq 0$
        $R \leftarrow R \cup \{V \rightarrow V_1 \ldots V_k \;\; W_1 \ldots W_l\}$
while a unary $(U \rightarrow V) \in R$ **do**
    $R \leftarrow R \setminus \{U \rightarrow V\}$
    if $U \neq V$ **then**
        for each $(V \rightarrow V_1 \ldots V_k) \in R$ **do** $R \leftarrow R \cup \{U \rightarrow V_1 \ldots V_k\}$
        if not $(W \rightarrow V_1 \ldots V_k \;\; V\, W_1 \ldots W_l) \in R$ **then**
            for each $(V \rightarrow V_1 \ldots V_k) \in R$ **do** $R \leftarrow R \setminus \{V \rightarrow V_1 \ldots V_k\}$
while an n-ary $(U \rightarrow V_1 \ldots V_n) \in R$ **do** // $n \geq 3$
    $R \leftarrow (R \setminus \{U \rightarrow V_1 \ldots V_n\}) \cup \{U \rightarrow V_1\, U\_V_1,\; U\_V_1 \rightarrow V_2 \ldots V_n\}$
return R

Chomsky Normal Form

Example: Empties (Removal)

Structural rules		Lexical rules	
s1	S $\rightarrow$ NP VP	l1	N $\rightarrow$ people
s2	VP $\rightarrow$ V NP	l2	N $\rightarrow$ fish
s3	VP $\rightarrow$ V NP PP	l3	N $\rightarrow$ tanks
s4	NP $\rightarrow$ NP NP	l4	N $\rightarrow$ rods
s5	NP $\rightarrow$ NP PP	l5	V $\rightarrow$ people
s6	NP $\rightarrow$ N	l6	V $\rightarrow$ fish
s7	NP $\rightarrow$ ε	l7	V $\rightarrow$ tanks
s8	PP $\rightarrow$ P NP	l8	P $\rightarrow$ with

Removal of empties

- Add new rules for each rule where NP occurs on the right side.

  Pseudocode lines 2–4.

Chomsky Normal Form

Example: Empties (Addition)

Structural rules		Lexical rules	
s1	S $\rightarrow$ NP VP	l1	N $\rightarrow$ people
s1'	S $\rightarrow$ VP	l2	N $\rightarrow$ fish
s2	VP $\rightarrow$ V NP	l3	N $\rightarrow$ tanks
s2'	VP $\rightarrow$ V	l4	N $\rightarrow$ rods
s3	VP $\rightarrow$ V NP PP	l5	V $\rightarrow$ people
s3'	VP $\rightarrow$ V PP	l6	V $\rightarrow$ fish
s4	NP $\rightarrow$ NP NP	l7	V $\rightarrow$ tanks
s4'	NP $\rightarrow$ NP	l8	P $\rightarrow$ with
s5	NP $\rightarrow$ NP PP		
s5'	NP $\rightarrow$ PP		
s6	NP $\rightarrow$ N		
s8	PP $\rightarrow$ P NP		
s8'	PP $\rightarrow$ P		

Chomsky Normal Form

Example: Unaries (Removal)

Structural rules		Lexical rules	
s1	S $\rightarrow$ NP VP	l1	N $\rightarrow$ people
s1’	S $\rightarrow$ VP	l2	N $\rightarrow$ fish
s2	VP $\rightarrow$ V NP	l3	N $\rightarrow$ tanks
s2’	VP $\rightarrow$ V	l4	N $\rightarrow$ rods
s3	VP $\rightarrow$ V NP PP	l5	V $\rightarrow$ people
s3’	VP $\rightarrow$ V PP	l6	V $\rightarrow$ fish
s4	NP $\rightarrow$ NP NP	l7	V $\rightarrow$ tanks
s4’	NP $\rightarrow$ NP	l8	P $\rightarrow$ with
s5	NP $\rightarrow$ NP PP		
s5’	NP $\rightarrow$ PP		
s6	NP $\rightarrow$ N		
s8	PP $\rightarrow$ P NP		
s8’	PP $\rightarrow$ P		

Chomsky Normal Form

Example: Unaries (Addition)

Structural rules		Lexical rules	
s1	S $\rightarrow$ NP VP	l1	N $\rightarrow$ people
s2	VP $\rightarrow$ V NP	l2	N $\rightarrow$ fish
s2"	S $\rightarrow$ V NP	l3	N $\rightarrow$ tanks
s2'	VP $\rightarrow$ V	l4	N $\rightarrow$ rods
s2"'	S $\rightarrow$ V	l5	V $\rightarrow$ people
s3	VP $\rightarrow$ V NP PP	l6	V $\rightarrow$ fish
s3"	S $\rightarrow$ V NP PP	l7	V $\rightarrow$ tanks
s3'	VP $\rightarrow$ V PP	l8	P $\rightarrow$ with
s3"'	S $\rightarrow$ V PP		
s4	NP $\rightarrow$ NP NP		
s4'	NP $\rightarrow$ NP		
s5	NP $\rightarrow$ NP PP		
s5'	NP $\rightarrow$ PP		
s6	NP $\rightarrow$ N		
s8	PP $\rightarrow$ P NP		
s8'	PP $\rightarrow$ P		

Chomsky Normal Form

Example: Unaries 2 (Removal)

Structural rules		Lexical rules	
s1	S → NP VP	l1	N → people
s2	VP → V NP	l2	N → fish
s2”	S → V NP	l3	N → tanks
s2’	VP → V	l4	N → rods
s2”’	S → V	l5	V → people
s3	VP → V NP PP	l6	V → fish
s3”	S → V NP PP	l7	V → tanks
s3’	VP → V PP	l8	P → with
s3”’	S → V PP		
s4	NP → NP NP		
s4’	NP → NP		
s5	NP → NP PP		
s5’	NP → PP		
s6	NP → N		
s8	PP → P NP		
s8’	PP → P		

Chomsky Normal Form

Example: Unaries 2 (Addition)

Structural rules		Lexical rules	
s1	S → NP VP	l1	N → people
s2	VP → V NP	l2	N → fish
s2”	S → V NP	l3	N → tanks
s2”’	S → V	l4	N → rods
s3	VP → V NP PP	l5	V → people
s3”	S → V NP PP	l5’	VP → people
s3’	VP → V PP	l6	V → fish
s3”’	S → V PP	l6’	VP → fish
s4	NP → NP NP	l7	V → tanks
s4’	NP → NP	l7’	VP → tanks
s5	NP → NP PP	l8	P → with
s5’	NP → PP		
s6	NP → N		
s8	PP → P NP		
s8’	PP → P		

Chomsky Normal Form

Example: Unaries 3 (Removal)

Structural rules		Lexical rules	
s1	S → NP VP	l1	N → people
s2	VP → V NP	l2	N → fish
s2"	S → V NP	l3	N → tanks
s2"'	S → V	l4	N → rods
s3	VP → V NP PP	l5	V → people
s3"	S → V NP PP	l5'	VP → people
s3'	VP → V PP	l6	V → fish
s3"'	S → V PP	l6'	VP → fish
s4	NP → NP NP	l7	V → tanks
s4'	NP → NP	l7'	VP → tanks
s5	NP → NP PP	l8	P → with
s5'	NP → PP		
s6	NP → N		
s8	PP → P NP		
s8'	PP → P		

Chomsky Normal Form

Example: Unaries 3 (Addition)

Structural rules		Lexical rules	
s1	S → NP VP	l1	N → people
s2	VP → V NP	l2	N → fish
s2”	S → V NP	l3	N → tanks
s3	VP → V NP PP	l4	N → rods
s3”	S → V NP PP	l5	V → people
s3’	VP → V PP	l5’	VP → people
s3”’	S → V PP	l5”	S → people
s4	NP → NP NP	l6	V → fish
s4’	NP → NP	l6’	VP → fish
s5	NP → NP PP	l6”	S → fish
s5’	NP → PP	l7	V → tanks
s6	NP → N	l7’	VP → tanks
s8	PP → P NP	l7”	S → tanks
s8’	PP → P	l8	P → with

Chomsky Normal Form

Example: Unaries 4–7 (Removal)

Structural rules		**Lexical rules**	
s1	S → NP VP	l1	N → people
s2	VP → V NP	l2	N → fish
s2”	S → V NP	l3	N → tanks
s3	VP → V NP PP	l4	N → rods
s3”	S → V NP PP	l5	V → people
s3’	VP → V PP	l5’	VP → people
s3”’	S → V PP	l5”	S → people
s4	NP → NP NP	l6	V → fish
s4’	NP → NP	l6’	VP → fish
s5	NP → NP PP	l6”	S → fish
s5’	NP → PP	l7	V → tanks
s6	NP → N	l7’	VP → tanks
s8	PP → P NP	l7”	S → tanks
s8’	PP → P	l8	P → with

Chomsky Normal Form

Example: Unaries 4–7 (Addition)

Structural rules		**Lexical rules**	
s1	S → NP VP	l1	NP → people
s2	VP → V NP	l2	NP → fish
s2”	S → V NP	l3	NP → tanks
s3	VP → V NP PP	l4	NP → rods
s3”	S → V NP PP	l5	V → people
s3’	VP → V PP	l5’	VP → people
s3”’	S → V PP	l5”	S → people
s4	NP → NP NP	l6	V → fish
s5	NP → NP PP	l6’	VP → fish
s5”	NP → P NP	l6”	S → fish
s8	PP → P NP	l7	V → tanks
		l7’	VP → tanks
		l7”	S → tanks
		l8	P → with
		l8’	PP → with
		l8”	NP → with

Chomsky Normal Form

Example: n-aries 1–2 (Removal)

Structural rules		Lexical rules	
s1	S $\rightarrow$ NP VP	l1	NP $\rightarrow$ people
s2	VP $\rightarrow$ V NP	l2	NP $\rightarrow$ fish
s2"	S $\rightarrow$ V NP	l3	NP $\rightarrow$ tanks
s3	VP $\rightarrow$ V NP PP	l4	NP $\rightarrow$ rods
s3"	S $\rightarrow$ V NP PP	l5	V $\rightarrow$ people
s3'	VP $\rightarrow$ V PP	l5'	VP $\rightarrow$ people
s3"'	S $\rightarrow$ V PP	l5"	S $\rightarrow$ people
s4	NP $\rightarrow$ NP NP	l6	V $\rightarrow$ fish
s5	NP $\rightarrow$ NP PP	l6'	VP $\rightarrow$ fish
s5"	NP $\rightarrow$ P NP	l6"	S $\rightarrow$ fish
s8	PP $\rightarrow$ P NP	l7	V $\rightarrow$ tanks
		l7'	VP $\rightarrow$ tanks
		l7"	S $\rightarrow$ tanks
		l8	P $\rightarrow$ with
		l8'	PP $\rightarrow$ with
		l8"	NP $\rightarrow$ with

Chomsky Normal Form

Example: n-aries 1–2 (Addition) $\rightarrow$ Results in Chomsky normal form!

Structural rules		**Lexical rules**	
s1	S $\rightarrow$ NP VP	l1	NP $\rightarrow$ people
s2	VP $\rightarrow$ V NP	l2	NP $\rightarrow$ fish
s2”	S $\rightarrow$ V NP	l3	NP $\rightarrow$ tanks
s3””	VP $\rightarrow$ V VP_V	l4	NP $\rightarrow$ rods
s3””’	VP_V $\rightarrow$ NP PP	l5	V $\rightarrow$ people
s3”””	S $\rightarrow$ V S_V	l5’	VP $\rightarrow$ people
s3”””’	S_V $\rightarrow$ NP PP	l5”	S $\rightarrow$ people
s3’	VP $\rightarrow$ V PP	l6	V $\rightarrow$ fish
s3”’	S $\rightarrow$ V PP	l6’	VP $\rightarrow$ fish
s4	NP $\rightarrow$ NP NP	l6”	S $\rightarrow$ fish
s5	NP $\rightarrow$ NP PP	l7	V $\rightarrow$ tanks
s5”	NP $\rightarrow$ P NP	l7’	VP $\rightarrow$ tanks
s8	PP $\rightarrow$ P NP	l7”	S $\rightarrow$ tanks
		l8	P $\rightarrow$ with
		l8’	PP $\rightarrow$ with
		l8”	NP $\rightarrow$ with

Probabilistic Context-Free Grammars

Probabilistic context-free grammar (PCFG)

- A CFG where each production rule is assigned a probablility

PCFG (Σ, N, S, R, P)

P A probability function $R \rightarrow [0,1]$ from production rules to probabilities, such that

$$\forall U \in N : \sum_{(U \rightarrow V) \in R} P(U \rightarrow V) = 1$$

(Σ, N, S, R as before)

Probabilities

- Trees. The probability $P(t)$ of a tree t is the product of the probabilities of the rules used to generate it.
- Strings. The probability $P(s)$ of a string s is the sum of the probabilities of the trees which yield s.

Probabilistic Context-Free Grammars

Example

Example PCFG

Structural rules			Lexical rules		
s1	S → NP VP	1.0	l1	N → people	0.5
s2	VP → V NP	0.6	l2	N → fish	0.2
s3	VP → V NP PP	0.4	l3	N → tanks	0.2
s4	NP → NP NP	0.1	l4	N → rods	0.1
s5	NP → NP PP	0.2	l5	V → people	0.1
s6	NP → N	0.7	l6	V → fish	0.6
s7	PP → P NP	1.0	l7	V → tanks	0.3
			l8	P → with	1.0

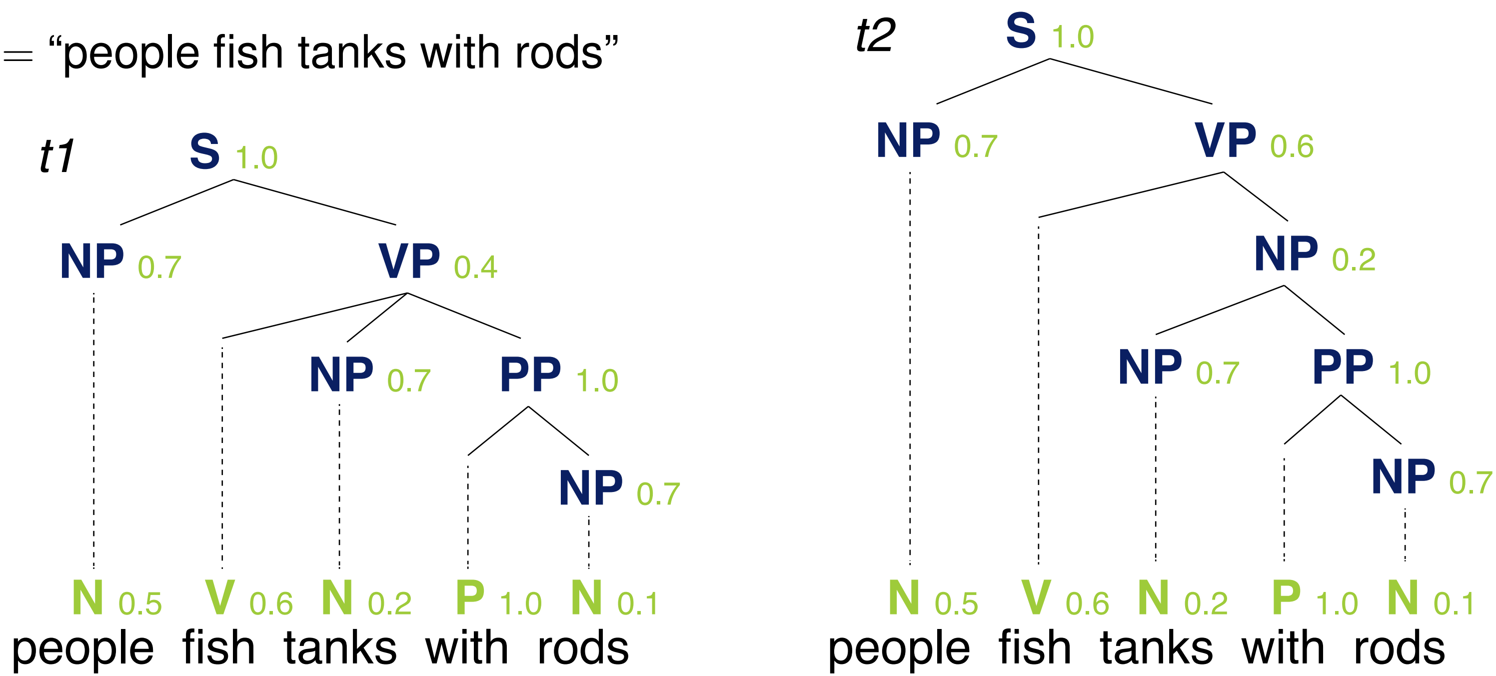
Notice

- For parsing, a PCFG should be transformed to Chomsky Normal Form or at least binarized.
- The origin of the probabilities is clarified below.

Probabilistic Context-Free Grammars

Example Probabilities

$s =$ “people fish tanks with rods”



Probabilities

$$
\begin{aligned}
P(t_1) &= 1.0 \cdot 0.7 \cdot 0.4 \cdot 0.5 \cdot 0.6 \cdot 0.7 \cdot 1.0 \cdot 0.2 \cdot 1.0 \cdot 0.7 \cdot 0.1 \quad = 0.0008232 \\
P(t_2) &= 1.0 \cdot 0.7 \cdot 0.6 \cdot 0.5 \cdot 0.6 \cdot 0.2 \cdot 0.7 \cdot 1.0 \cdot 0.2 \cdot 1.0 \cdot 0.7 \cdot 0.1 \quad = 0.00024696 \\
P(s) &= P(t_1) + P(t_2) \quad = 0.0008232 + 0.00024696 \quad = 0.00107016
\end{aligned}
$$

Constituency Parsing

Constituency Parsing

Constituency parsing

- The text analysis that determines the phrase structure of a sentence with respect to a given grammar
- Often used in NLP as preprocessing where syntax is important
- Parsing works robust across domains of well-formatted texts.

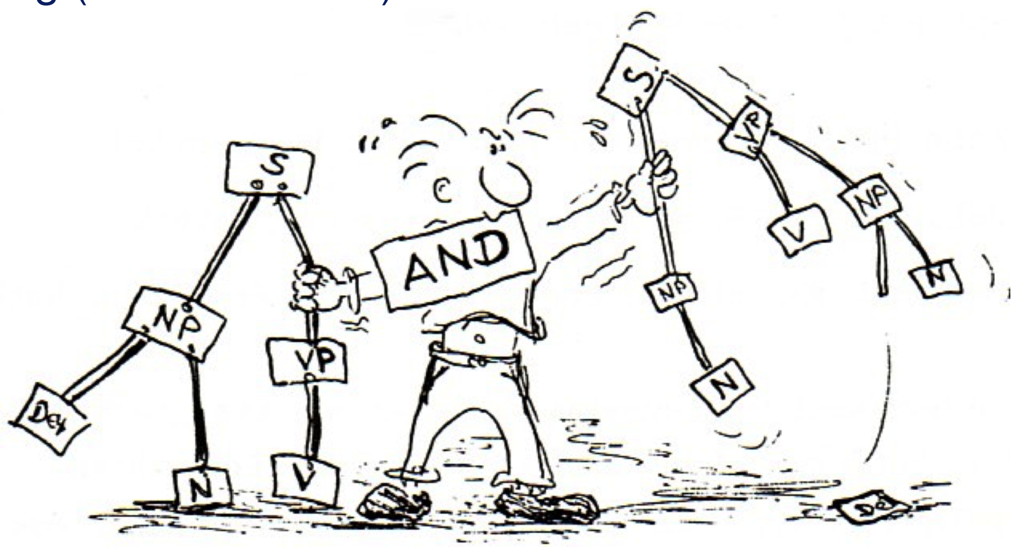
Downstream tasks based on parsing

- Named entity recognition in complex domains (e.g., biology)
- Relationship extraction, both for semantic and temporal relations
- Coreference resolution, to identify candidate matching references
- Opinion mining regarding aspects of products or similar
- Machine translation, to analyze the source sentence
- Question answering, particularly in high-precision scenarios

... and so forth

Constituency Parsing

Classical Parsing (before ~ 1990)



Classical parsing

- Hand-crafted CFG, along with a lexicon
- Usage of CFG-based systems to prove parses from words
- This scales badly and fails to give high language coverage.

Example “Fed raises interest rates 0.5% in effort to control inflation”

- Minimal grammar. 36 possibly parse trees
- Real-size broad-coverage grammar. Millions of parse trees

Constituency Parsing

Classical Parsing: Problems and Solutions

Grammars with categorical constraints

- Limitation of the chance for unlikely parse trees of sentences
- But constraints reduce the coverage of a grammar.
- In classical parsing, typically $\sim$30% of all sentences cannot be parsed.

Less constrained grammars

- Can parse more sentences
- But simple sentences end up with even more parse trees.
- No way to choose between different parse trees

Solution: Statistical parsing

- Very loose grammars that admit millions of parse trees for sentences
- Mechanisms that find the most likely parse tree of a sentence quickly
- Nowadays, most parsers are based on statistics (probabilities).

Constituency Parsing

Statistical Parsing

How to build a statistical parser?

- Statistical parsers are based on PCFGs (or varations thereof).
- The rules and probabilities of the PCFGs are derived from *treebanks*.

Treebanks

- A treebank is corpus with tree-structured annotations

  One of the most used treebanks is the *Penn Treebank*, PTB (Marcus et al., 1993).
- Building a treebank is an expensive manual process done by experts.
- Slower than building a grammar, but the benefits outweigh the costs

Benefits of treebanks

- Statistics. Frequencies and distributional information
- Development. Reusable for many parsers, POS taggers, etc.
- Evaluation. Basis for evaluating a developed system
- Language. Valuable resource for linguistics in general

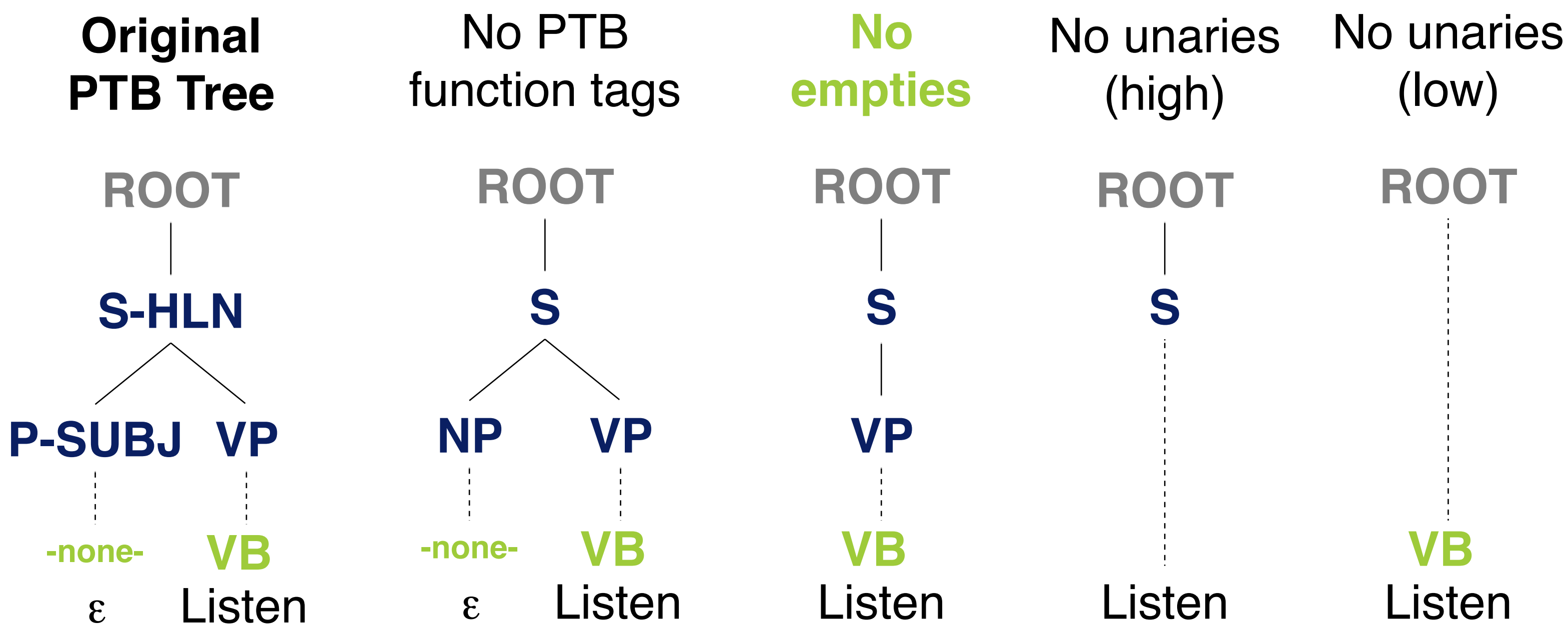
Constituency Parsing

Example PTB Sentence Representation

```
( (S
    (NP-SBJ (DT The) (NN move))
    (VP (VBD followed)
      (NP
        (NP (DT a) (NN round))
        (PP (IN of)
          (NP
            (NP (JJ similar) (NNS increases))
            (PP (IN by)
              (NP (JJ other) (NNS lenders)))
            (PP (IN against)
              (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
      (, ,)
      (S-ADV
        (NP-SBJ (-NONE- *))
        (VP (VBG reflecting)
          (NP
            (NP (DT a) (VBG continuing) (NN decline))
            (PP-LOC (IN in)
              (NP (DT that) (NN market)))))))
    (. .)))
```

Constituency Parsing

From Treebank to Chomsky Normal Form



Notice

- No unaries. Low preferred over high, since it keeps lexical information
- No empties. Enough for parsing and makes a reconstruction of the original parse tree easier

Constituency Parsing

Attachment Ambiguity

Key parsing problem

- Correct attachment of the various constituents in a sentence, such as prepositional phrases, adverbial phrases, infinitives, ...

“The board approved	its acquisition	$\rightarrow$ attaches to “approved”
	by Royal Trustco Ltd.	$\rightarrow$ attaches to “its acquisition”
	of Toronto	$\rightarrow$ attaches to “by Royal Trustco Ltd.”
	for $27 a share	$\rightarrow$ attaches to “its acquisition”
	at its monthly meeting.”	$\rightarrow$ attaches to “approved ... for $27 a share”

How to find the correct attachment?

- Potential attachments grow exponentially with number n of constituents
- The problem is *AI complete*.

“I saw a man with a telescope.”

- Words predict attachment well.

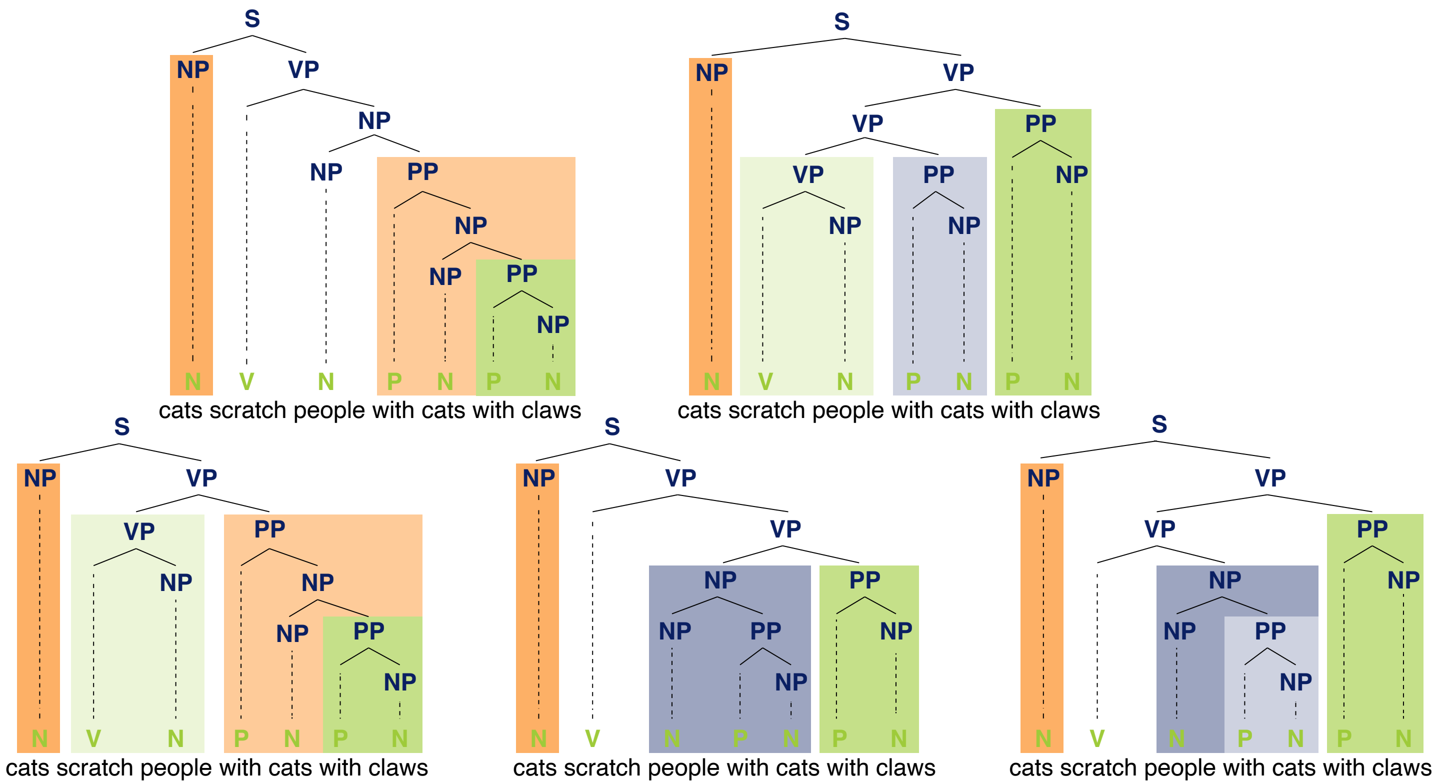
“Moscow sent more than 100,000 soldiers into Ukraine.”

Constituency Parsing

Attachment Ambiguity in Statistical Parsing

Two problems to solve in statistical parsing

1. Choose the most likely parse (according to statistics).
2. Avoid to do repeated work (algorithmically).



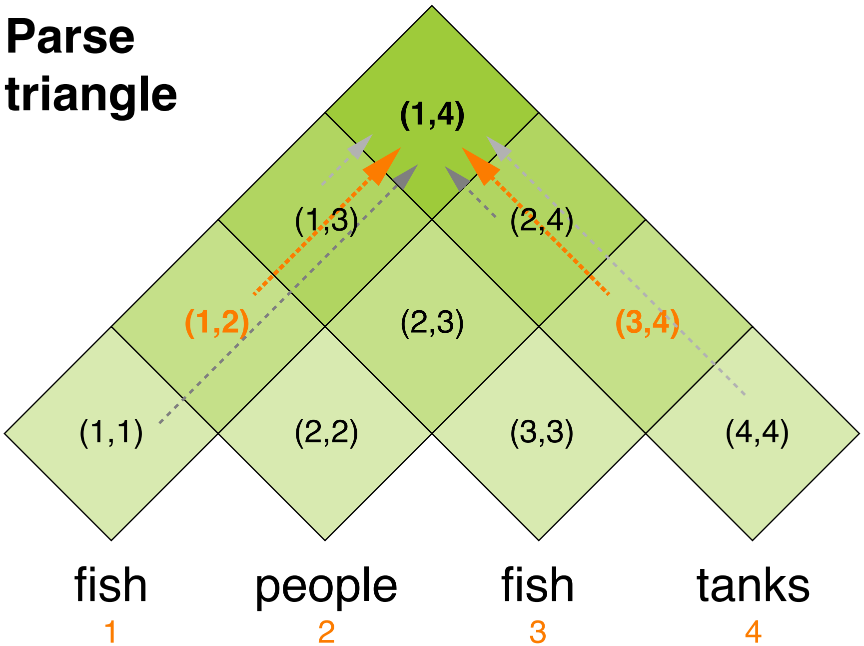
CKY Parsing

Cocke-Kasami-Younger (CKY) parsing (aka CYK parsing)

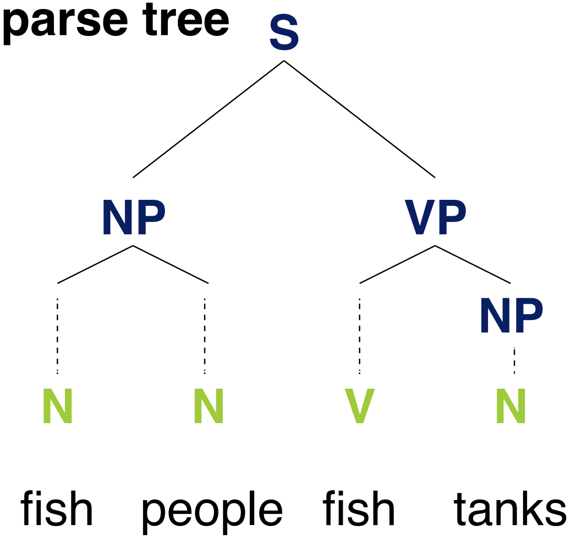
- A dynamic programming parsing algorithm based on PCFG in CNF
- Goal. Get the most likely constituency parse tree for a sentence.
- Asympotically strong: cubic time, quadratic space

  With respect to the length of the sentence and the number of non-terminals

Parse triangle



Most likely parse tree



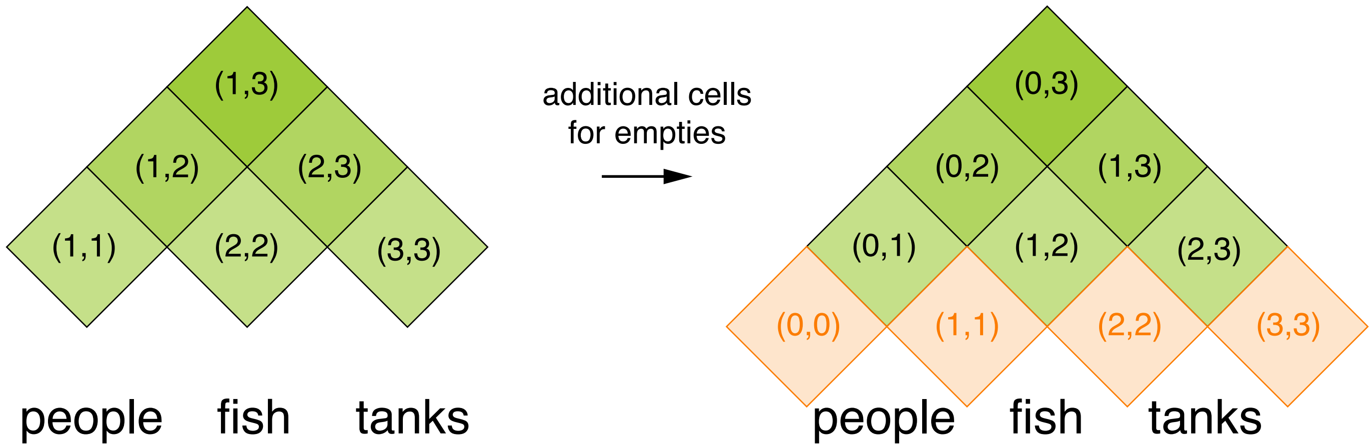
CKY Parsing

Extension

Extended CKY Parsing

- A parser based on a *binarized* PCFG, introduced below
- Includes unaries without increasing asymptotic complexity

  Makes the algorithm more complicated, but keeps the grammar smaller
- Empties are treated like unaries, after adding a cell row.



Binarization is needed for cubic time

- Without, CKY parsing does not work by concept. Why not?

  Other parsers may not require a binarized grammar, but then do binarization implicitly.

CKY Parsing

Pseudocode (1 out of 2)

Signature

- Input. A sentence (represented by a list of tokens), a binarized PCFG
- Output. The most likely parse tree of the sentence

extendedCKYParsing(List<Token> tokens, PCFG (Σ, N, S, R, P))

```
double [][][] probs ← new double[#tokens][#tokens][#N]
for int i ← 1 to #tokens do // Lexicon rules (and unaries)
    for each U ∈ N do
        if (U→tokens[i]) ∈ P then
            probs[i][i][U] ← P(U→tokens[i])
    boolean added ← 'true' // As of here: Handle unaries
    while added = 'true' do
        added ← 'false'
        for each U,V ∈ N do
            if probs[i][i][V]>0 and (U → V) ∈ P then
                double prob ← P(U → V) · probs[i][i][V]
                if prob > probs[i][i][U] then
                    probs[i][i][U] ← prob
                    added ← 'true'
// ... continued on next slide...
```

CKY Parsing

Pseudocode (2 out of 2)

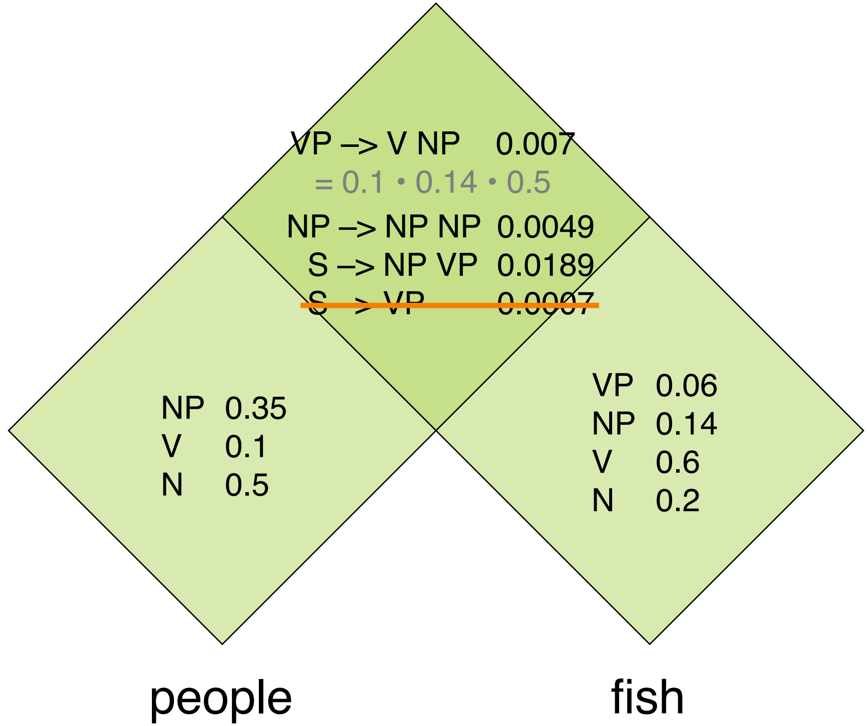
```
        // ... lines 1-14 on previous slide...
15.     for int length ← 2 to #tokens do // Structural rules
16.         for int beg ← 1 to #tokens - length + 1 do
17.             int end ← beg + length - 1
18.             for int split ← beg to end-1 do
19.                 for int U,V,W ∈ N do
20.                     int prob ← probs[beg][split][V] ·
                                  probs[split+1][end][W] · P(U → V W)
21.                     if prob > probs[beg][end][U] then
22.                         probs[beg][end][U] ← prob
23.             boolean added ← 'true' // As of here: Handle unaries
24.             while added do
25.                 added ← 'false'
26.                 for U,V ∈ N do
27.                     prob ← P(U → V) · probs[beg][end][V];
28.                     if prob > probs[beg][end][U] then
29.                         probs[beg][end][U] ← prob
30.                         added ← 'true'
31.     return buildTree(probs) // Reconstruct tree from triangle
```

CKY Parsing

Example

A binarized PCFG

Structural rules		
s1	S → NP VP	0.9
s1’	S → VP	0.1
s2	VP → V NP	0.5
s2’	VP → V	0.1
s3’	VP → V VP_V	0.3
s3”	VP → V PP	0.1
s3”’	VP_V → NP PP	1.0
s4	NP → NP NP	0.1
s5	NP → NP PP	0.2
s6	NP → N	0.7
s7	PP → P NP	1.0



Filling cells

- Compute probabilities for each cell.
- Keep only highest for each left side.

CKY Parsing

Run-time Complexity

Run-time of pseudocode part 1

- $\mathcal{O}(n)$ times for-loop in lines 1–14, $n = \#$ tokens
- $\mathcal{O}(|N|)$ times for-loop in lines 3–5
- $\mathcal{O}(|N|^2)$ times while-loop in lines 7–14

$\mathcal{O}(n \cdot |N|^2)$
for part 1 in total

Run-time of pseudocode part 2

- $\mathcal{O}(n)$ times for-loop in lines 15–30
- $\mathcal{O}(n)$ times for-loop in lines 16–30
- $\mathcal{O}(n)$ times for-loop in lines 18–22
- $\mathcal{O}(|N|^3)$ times for-loop in lines 19–22
- $\mathcal{O}(|N|^2)$ times while-loop in lines 24–30
- $\mathcal{O}(n^2)$ for building the tree in line 31

$\mathcal{O}(n^3 \cdot |N|^3)$
for part 2 in total

Overall run-time

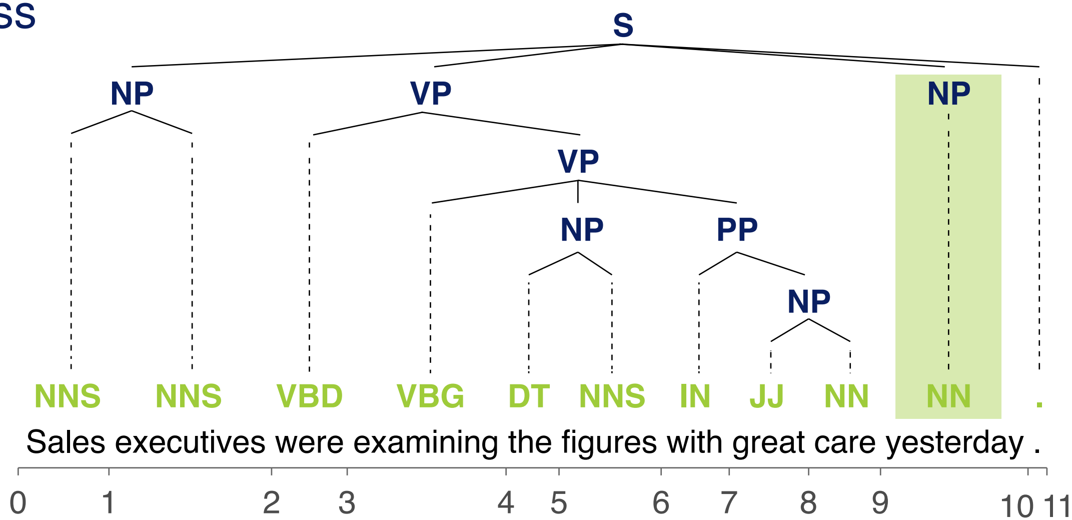
- Extended CKY parsing has a run-time of $\mathcal{O}(n^3 \cdot |N|^3)$.
- Several optimizations possible, but asymptotic complexity remains

CKY Parsing

Evaluation of Effectiveness

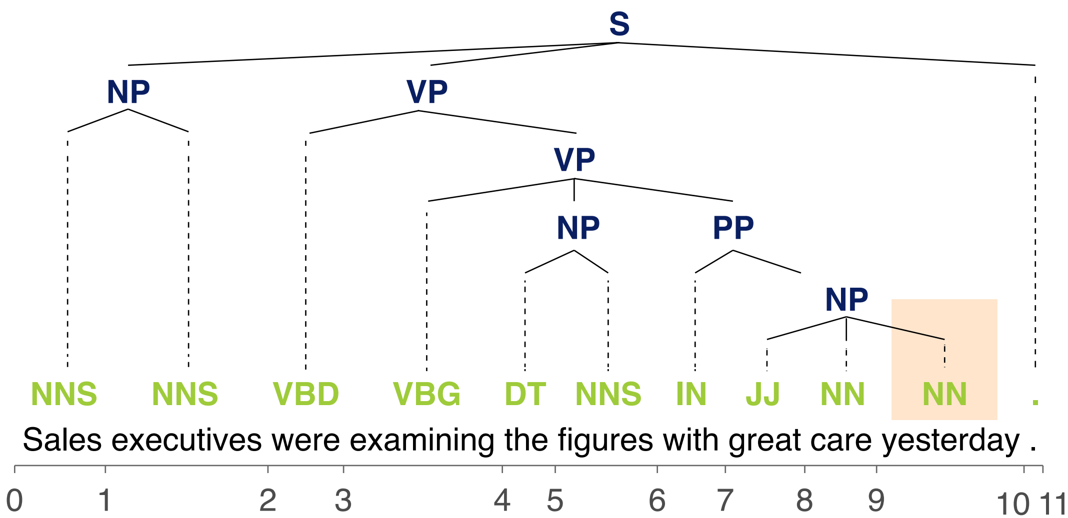
Gold standard brackets

S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9), NP-(4:6), PP-(6:9), NP-(7:9), NP-(9:10)



Candidate brackets

S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:6), PP-(6:10), NP-(7:10)



CKY Parsing

Evaluation of Effectiveness

8 gold standard brackets

S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9), NP-(4:6), PP-(6:9), NP-(7:9), NP-(9:10)

7 candidate brackets

S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:6), PP-(6:10), NP-(7:10)

Effectiveness in the example

- Labeled precision (LP). $0.429 = 3 / 7$
- Labeled recall (LR). $0.375 = 3 / 8$
- Labeled F_1-score. $0.400 = 2 \cdot LP \cdot LR / (LP + LR)$
- POS tagging accuracy. $1.000 = 11 / 11$

Effectiveness of CKY in general (Charniak, 1997)

- Labeled F_1 $\sim$0.87 when trained and tested on Penn Treebank
- CKY is robust, i.e., it parses almost anything (but with low probabilities).

Lexicalized Parsing

Limitations of standard PCFGs

- PCFGs assume that the plausibility of structures is independent of the words used, i.e., each rule has a fixed probability.

$$P(VP \rightarrow V\ NP\ NP) = 0.00151$$

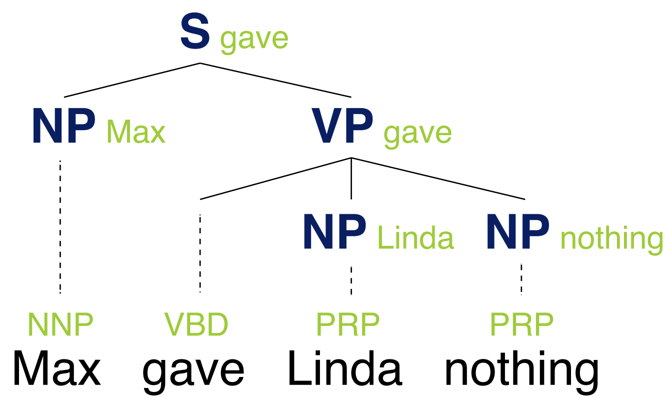
- However, specific words may make certain rules particularly (un)likely.

Lexicalization of PCFGs (Collins, 1999)

- Condition the probability of a rule on the *head word* of the given phrase.

$$P(VP \rightarrow V\ NP\ NP \mid \text{“said”}) = 0.00001$$
$$P(VP \rightarrow V\ NP\ NP \mid \text{“gave”}) = 0.01980$$

- Rationale. The head word represents a phrase’s structure and meaning well.



Lexicalized parsing

- Constituency parsing based on a lexicalized PCFG

Lexicalized Parsing

“Unlexicalization” of PCFGs

Hypothesis

- Lexical selection between content words is not crucial for parsing.
- More important are grammatical features, such as verb form, presence of a verb auxiliary, ...

Unlexicalized parsing (Klein and Manning, 2003)

- Rules are not systematically specified down to the level of lexical items.
- No semantic lexicalization for nouns, such as “$\text{NP}_{\text{stocks}}$”
- Instead: Structural “lexicalization”, such as “$\text{NP}^{\text{S}}_{\text{CC}}$”
  Meaning: Parent node is “S” and noun phrase is coordinating.
- Keep functional lexicalization of closed-class words, such as “VB-have”

Learned unlexicalized parsing (Petrov and Knight, 2007)

- Learn extra information for a non-terminal based on training data.
- Parse based on unlexicalized PCFG.

Constituency Parsing

Evaluation Results

Comparison of the different approaches

- All in exactly the same setting on the Penn Treebank

Approach	Source	Labeled F_1
Extended CKY parsing	Charniak (1997)	0.87
Lexicalized parsing	Collins (1999)	0.89
Unlexicalized parsing	Klein and Manning (2003)	0.86
Learned unlexicalized parsing	Petrov and Klein (2007)	0.90
Combining parsers	Fossum and Knight (2009)	0.92
Transformer-based span parsing	Tian et al. (2020)	**0.96**

Notice

- Neural parsers nowadays outperform probabilistic parsers significantly.
- Still, many build on core ideas of CKY parsing, like Tian et al. (2020).
  The details are beyond the scope of this course.

Dependency Parsing

Dependency Grammars

Dependency grammar

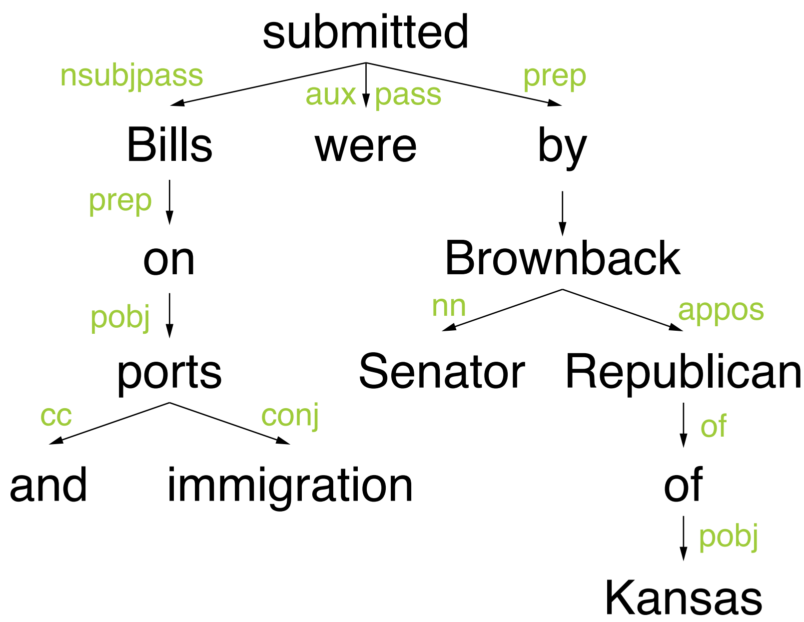
- A grammar that models the syntactic structure of a sentence by linking its tokens with binary asymmetric relations
- Relations, called *dependencies*, define grammatical connections.

  Subject, prepositional object, apposition, etc.

Graph representation

- Each node is a token.
- An edge connects a *head* with a *dependent* node.
- The nodes and edges form a fully connected tree with a single head.

  If available, the main verb of the first main clause is the head.

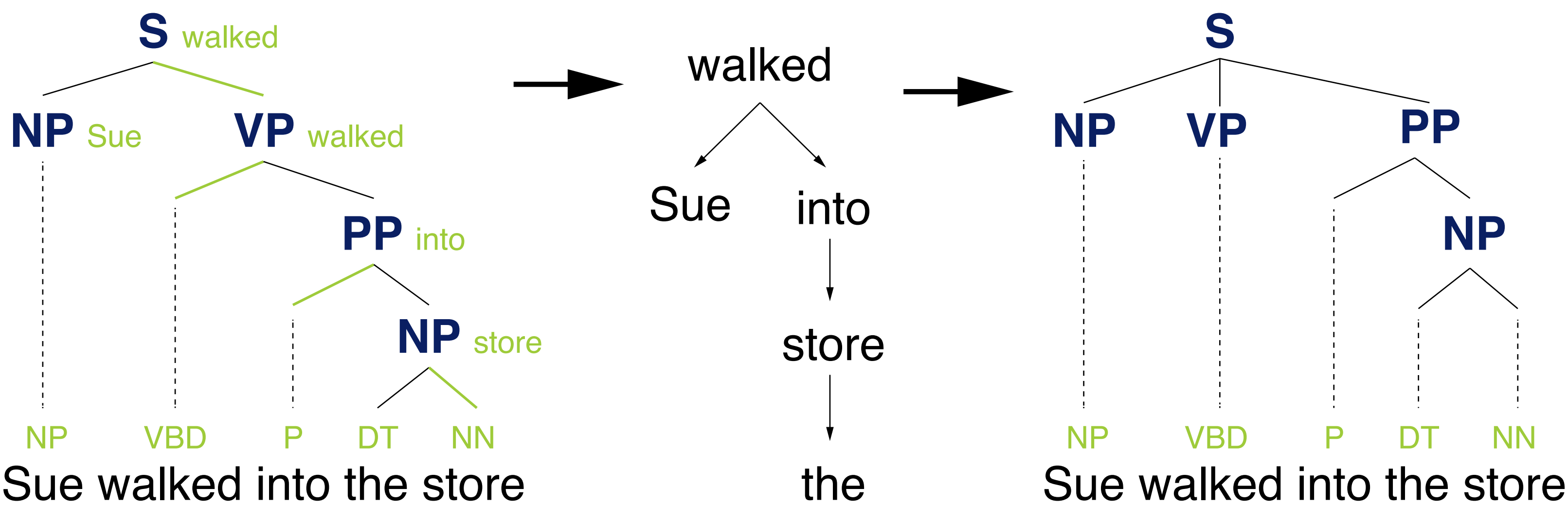


Dependency Grammars

Dependency Grammars vs. Phrase Structure Grammars

Dependency vs. phrase structure

- CFGs do not have the notion of a head — officially.
- Modern statistical parsers usually include phrasal "head rules".
  For example, the head of an NP is a noun, number, adjective, ...
- Given head rules, constituencies can be converted to dependencies.
- Dependencies can be converted back to constituencies, but a word's dependents will be on the same level.



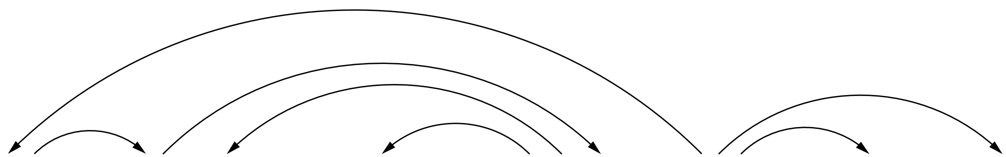
Dependency Grammars

Identification of Dependencies

Selected features of dependencies

- Breaks. Dependencies rarely span intervening verbs or punctuation.
- Valency. Heads have usual numbers of dependents on each side.
- Affinities. Some dependencies are more plausible than others.

  For example “issues → the” rather than “the → issues”.



Discussion of the outstanding issues was completed .

Example “Retail sales drop in April cools afternoon market trading.”

“sales”	dependent of?	→ “drop”
“April”	dependent of?	→ “in”
“afternoon”	dependent of?	→ “trading”
“trading”	dependent of?	→ “cools”

Dependency Grammars

Parsing Methods

Dynamic programming (Eisner, 1996)

- Lexicalized PCFG parsing, similar to CKY would need $\mathcal{O}(n^5)$ steps.
- When forcing parse structures with heads at the ends, $\mathcal{O}(n^3)$ is possible.

Graph algorithms (McDonald et al., 2005)

- Build a maximum spanning tree for a sentence. Score dependencies independently using machine learning. $\rightarrow \mathcal{O}(n^3)$.
- More accurate on long dependencies and dependencies near the root.

Transition-based parsing (Nivre et al., 2008)

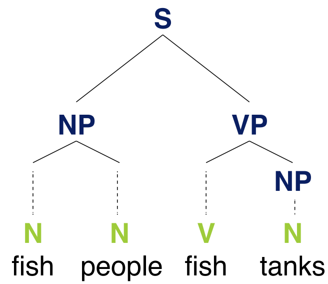
- Shift from left to right over a sentence. Make greedy attachment choices guided by a machine learning classifier. $\rightarrow \mathcal{O}(n)$
- More accurate on short dependencies and disambiguation of core grammatical functions.

Conclusion

Conclusion

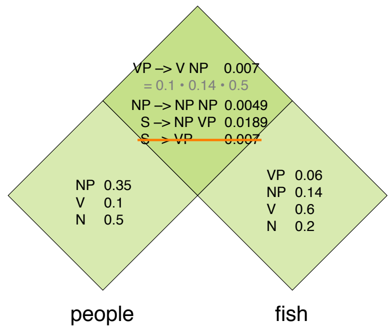
NLP using context-free grammars (CFGs)

- CFGs model hierarchical structure.
- PCFGs extend CFGs by probabilities (via statistics).
- In NLP, PCFGs used for phrase structure of sentences



Syntactic parsing based on grammars

- PCFGs used for CKY constituency parsing
- Extensions include lexicalization and unlexicalization
- Dependency grammars for dependency parsing



Benefits and limitations

- Statistical grammars are a core technique of NLP.
- Creation of large-scale treebanks is very expensive.
- CFGs just model the ways syntax is constructed.

```
( (S
  (NP-SBJ (DT The) (NN move))
  (VP (VBD followed)
    (NP
      (NP (DT a) (NN round))
      (PP (IN of)
        (NP
          (NP (JJ similar) (NNS increases))
          (PP (IN by)
            (NP (JJ other) (NNS lenders)))
          (PP (IN against)
            (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
    (, ,)
    (S-ADV
      (NP-SBJ (-NONE- *))
      (VP (VBG reflecting)
        (NP
          (NP (DT a) (VBG continuing) (NN decline))
          (PP-LOC (IN in)
            (NP (DT that) (NN market)))))))
  (. .)))
```

References

Much content and many examples taken from

- **Jurafsky and Manning (2016).** Daniel Jurafsky and Christopher D. Manning. Natural Language Processing. Lecture slides from the Stanford Coursera course, 2016. https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html.
- **Jurafsky and Manning (2009).** Daniel Jurafsky and James H. Martin. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. Prentice-Hall, 2nd edition, 2009.
- **Meyer auf der Heide (2010).** Friedhelm Meyer auf der Heide. Einführung in Berechenbarkeit, Komplexität und Formale Sprachen. Begleitmaterial zur Vorlesung, 2010. https://www.hni.uni-paderborn.de/fileadmin/Fachgruppen/Algorithmen/Lehre/Vorlesungsarchiv/WS_2009_10/Einfuehrung_in_die_Berechenbarkeit_K_u_f_S/skript.pdf
- **Wachsmuth (2015).** Henning Wachsmuth: Text Analysis Pipelines — Towards Ad-hoc Large-scale Text Mining. LNCS 9383, Springer, 2015.

References

Other references

- **Charniak (1997).** Statistical parsing with a context-free grammar and word statistics. In AAAI-97, Menlo Park. AAAI Press.
- **Collins (1999).** Michael J. Collins. Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia, 1999.
- **Eisner (1996).** Jason M. Eisner. Three New Probabilistic Models for Dependency Parsing: An Exploration. In COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics, 1996.
- **Fossum and Knight (2009).** Victoria Fossum and Kevin Knight. Combining Constituent Parsers. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, pages 253–256, 2009.
- **Klein and Manning (2003).** Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, pages 423–430. 2003.
- **Marcus et al. (1993).** Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics, 19(2):313–330, 1993.

References

Other references

- **McDonald et al. (2005).** Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pages 523–530, 2005.
- **Nivre (2008).** Joakim Nivre. Algorithms for Deterministic Incremental Dependency Parsing. Computational Linguistics, 34(4):513–553, 2008.
- **Petrov and Klein (2007).** Slav Petrov and Dan Klein. Improved Inference for Unlexicalized Parsing. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pages 404–411, 2007.
- **Tian et al. (2020).** Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. Improving Constituency Parsing with Span Attention. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1691–1703, 2020.