SYNTAX

Matt Post
IntroHLT class
10 September 2020

Johns Hopkins University

and stupor his the Fred with pain from ease couldn’t would a set he cigarette out the that for in wife Jones was during caring a often drugs house but screaming the crying at for didn’t fear worn sleep ablaze day the from that she night

Fred Jones was worn out from caring for his often screaming and crying wife during the day but he couldn't sleep at night for fear that she in a stupor from the drugs that didn't ease the pain would set the house ablaze with a cigarette

- 46 words, 46! permutations of those words, the vast majority of them ungrammatical and meaningless
- How is that we can
  - process and understand this sentence?
  - discriminate it from the sea of ungrammatical permutations it floats in?

Today we will cover

Linguistics

what is syntax?

where do grammars come from?

how can a computer find a sentence's structure?

Computer Science

Goals for today

- After today, you should be able to
  - Give a working definition of syntax and describe how linguists think about it
  - Describe two well-known grammar formalisms and projects supporting them
  - Discuss issues related to universal language features
  - Describe the formal language hierarchy
  - Describe algorithms for parsing the two grammar formalisms

Outline

what is syntax?

where do grammars come from?

how can a computer find a sentence's structure?

Linguistic fields of study

- Phonetics: sounds

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure
- **Syntax**: external word structure (sentences)

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure
- **Syntax**: external word structure (sentences)
- Semantics: sentence meaning

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure
- **Syntax**: external word structure (sentences)
- Semantics: sentence meaning
- Pragmatics: contextualized meaning and communicative goals

Aside

- Much of our focus is on *written language*, but language is first and foremost spoken
- Why does this matter?

Aside

- Much of our focus is on *written language*, but language is first and foremost spoken
- Why does this matter?
- Which of these is easier for a computer to work with?

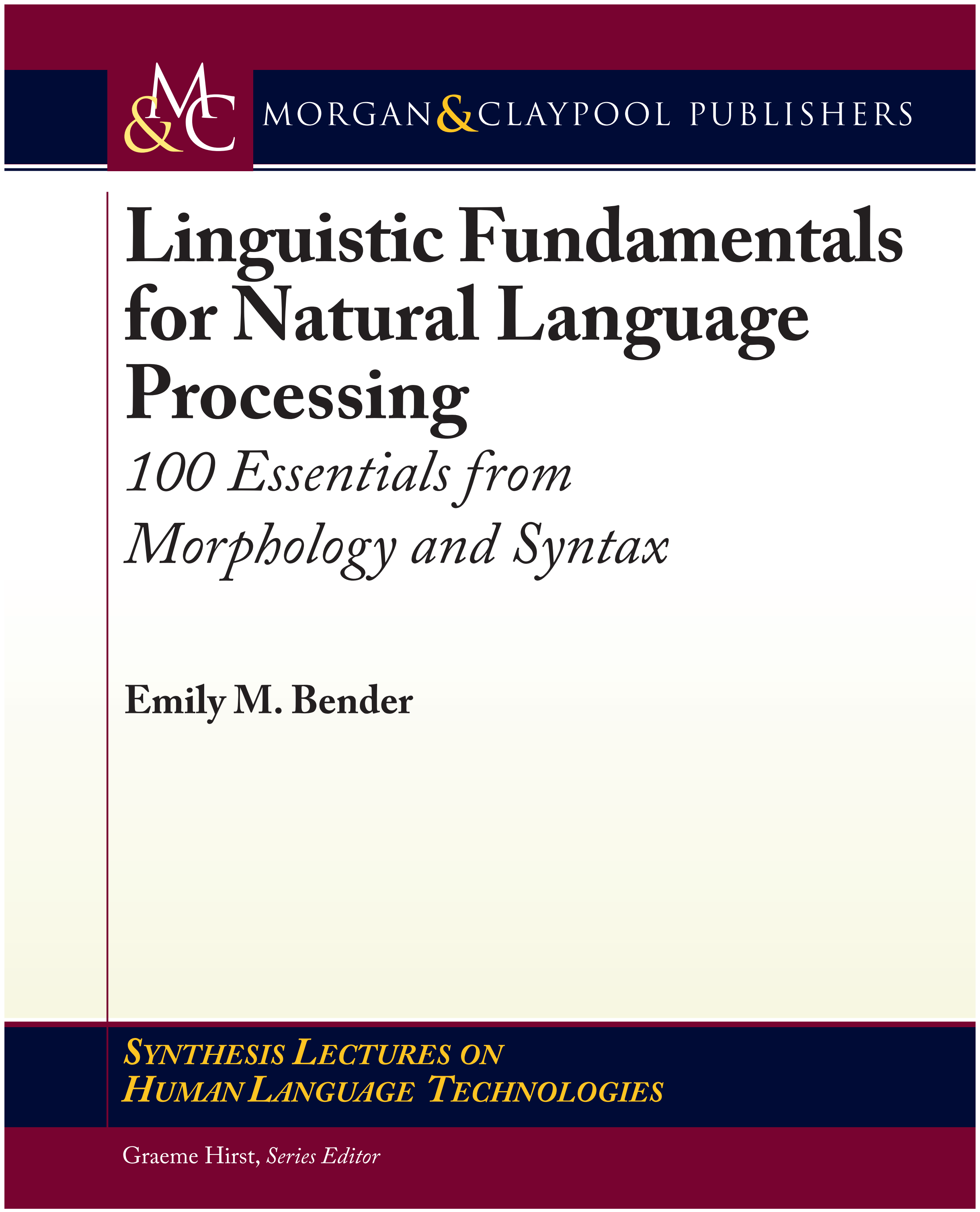
Aside

- Much of our focus is on *written language*, but language is first and foremost spoken
- Why does this matter?
- Which of these is easier for a computer to work with?
  - (written) *Dipanjan asked a question*

Aside

- Much of our focus is on *written language*, but language is first and foremost spoken
- Why does this matter?
- Which of these is easier for a computer to work with?
  - (written) *Dipanjan asked a question*
  - (spoken) *Dipanjan, uh, he, uh, um, was wondering, uh, he had a question*

Today's focus



What is syntax?

- A set of constraints on the possible sentences in the language
  - *A set of <u>constraint</u> on the possible <u>sentence.</u>
  - *Dipanjan had <u>[a]</u> question.
  - *You are <u>on</u> class.
- At a coarse level, we can divide all possible sequences of words into two groups: *valid* and *invalid* (or *grammatical* and *ungrammatical*)

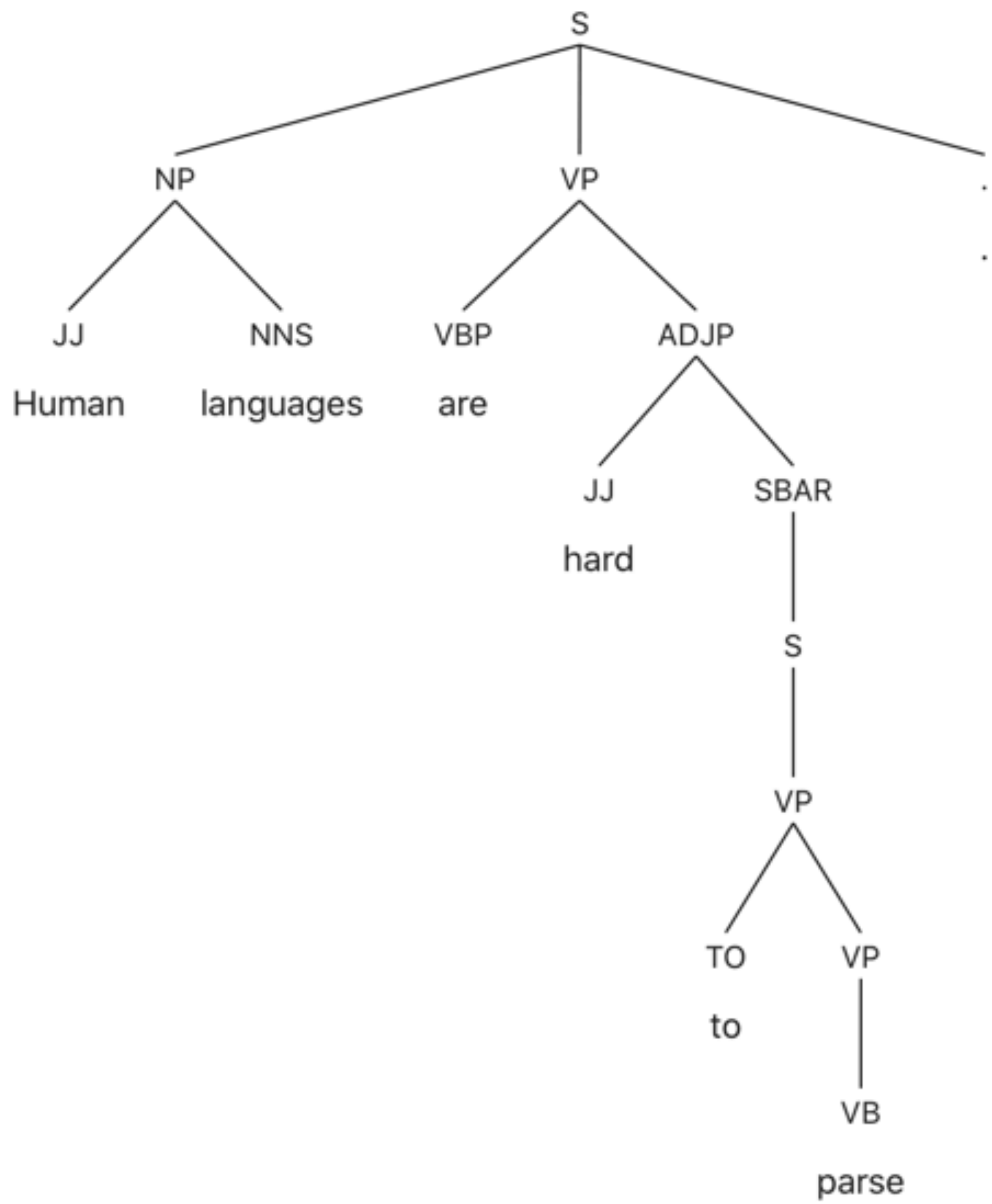
Human judgments

- How do we know what's in and out? We simply ask humans
- But how do humans know?
  - Bad idea: big lists
  - Better idea: grammars

A hierarchical view

- A grammar is a *finite set of* ***rules*** licensing a (possibly infinite) *number of* ***strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]
  - [noun phrase] → [determiner]? [adjective]* [noun]
  - [predicate] → [verb phrase] [adjunct]
- Rules are *phrasal* or *terminal*
  - Phrasal rules form **constituents** in a tree
  - Terminal rules are **parts of speech** and produce words

Example



S
NP
VP
.
JJ
NNS
VBP
ADJP
.
Human
languages
are
JJ
SBAR
hard
S
VP
TO
VP
to
VB
parse

POS Examples

- No general agreement about the exact set of parts of speech
- Penn Treebank tagset examples

POS Examples

- No general agreement about the exact set of parts of speech
- Penn Treebank tagset examples
  - nouns: NN, NNS, NNP, NNPS

POS Examples

- No general agreement about the exact set of parts of speech
- Penn Treebank tagset examples
  - nouns: NN, NNS, NNP, NNPS
  - adverbs: RB, RBR, RBS, RP

POS Examples

- No general agreement about the exact set of parts of speech
- Penn Treebank tagset examples
  - nouns: NN, NNS, NNP, NNPS
  - adverbs: RB, RBR, RBS, RP
  - verbs: VB, VBD, VBG, VBN, VBP, VBZ

POS Examples

- No general agreement about the exact set of parts of speech
- Penn Treebank tagset examples
  - nouns: NN, NNS, NNP, NNPS
  - adverbs: RB, RBR, RBS, RP
  - verbs: VB, VBD, VBG, VBN, VBP, VBZ
  - (Here, different tags are used to capture the small bit of morphology present in English)

Parts of Speech (POS)

- Three definitions of **noun**

Grammar school
("metaphysical")
a person, place, thing, or idea

Parts of Speech (POS)

- Three definitions of **noun**

Grammar school ("metaphysical")	Distributional
a person, place, thing, or idea	*the set of words that have the same distribution as other nouns*
	{I,you,he} saw the {bird,cat,dog}.

Parts of Speech (POS)

- Three definitions of **noun**

<u>Grammar school</u>
("metaphysical")
a person, place, thing, or idea

<u>Distributional</u>

the set of words that have the same distribution as other nouns

{I,you,he} saw the {bird,cat,dog}.

<u>Functional</u>

the set of words that serve as arguments to verbs

verb
noun *adverb*
adjective

Phrases and Constituents

- Longer sequences of words can perform the same function as individual parts of speech:
  - I saw [a$_{DT}$ kid$_{N}$]$_{NP}$
  - I saw [a kid playing basketball]$_{NP}$
  - I saw [a kid playing basketball alone on the court]$_{NP}$
- This gives rise to the idea of a *phrasal constituent*, which functions as a unit in relation to the rest of the sentence

Constituent tests

- How do you know if a phrase functions as a constituent?
- A few tests
  - *Coordination*
    - Kim [read a book], [gave it to Sandy], and [left].
  - *Substitution with a word*
    - Kim read [a very interesting book about grammar].
    - Kim read [it].
  - See Bender #51

Heads, arguments, & adjuncts

- Syntax is about the relationships among words and phrases in a sentence

Heads, arguments, & adjuncts

- Syntax is about the relationships among words and phrases in a sentence
- Each constituent has its own internal structure, as well as relationship with words and constituents outside it

Heads, arguments, & adjuncts

- Syntax is about the relationships among words and phrases in a sentence
- Each constituent has its own internal structure, as well as relationship with words and constituents outside it
- Hierarchical structure among constituents
  - Top down, each constituent has a head
  - Heads have (phrasal) dependents
  - Dependents can be required (*arguments*) or optional (*adjuncts)*
  - A head word often controls the structure of its modifiers

Heads

- *Head*: “the sub-constituent which determines the internal structure and external distribution of the constituent as a whole” (Bender #52)
- Examples
  - sentence: (usually) the main verb
  - noun phrase: (usually) the main noun
  - verb phrase: (usually) the active verb

Dependents: Arguments & adjuncts

- *Dependents* of a head:
  - *Arguments:* selected/licensed by the head and **complete** the meaning
  - *Adjuncts:* not selected and **refine** the meaning

Constituent structure

- The head often constrains the internal structure of a constituent
- Examples
  - verb
    - [Kim]$^{\text{ARGUMENT}}$ **is** [ready]$^{\text{ADJUNCT}}$.
  - adjective
    - Kim is [**ready**$_{\text{ADJ}}$ [to make a pizza]$_{\text{V}}$].
    - * Kim is [**tired**$_{\text{ADJ}}$ [to make a pizza]$_{\text{V}}$].
  - noun
    - [The [red]$_{\text{ADJ}}$ **ball**]
    - * [The [red]$_{\text{ADJ}}$ **ball** [the stick]$_{\text{N}}$]
    - [The [red]$_{\text{ADJ}}$ **ball** [on top of the stick]$_{\text{PP}}$]

More examples

- Kim **planned** [to **give** Sandy books].
- * Kim **planned** [to **give** Sandy].
- Kim **planned** [to give books].
- * Kim **planned** [to **see** Sandy books].
- Kim [**would** [**give** Sandy books]].
- Pat [**helped** [Kim **give** Sandy books]].
- * [[**Give** Sandy books] [**surprised** Kim]].

Summary

what is syntax?

A finite set of rules licensing an infinite number of strings

The rules specify how words and phrases relate to one another in a hierarchical manner

No one knows what the actual rules are, but there is consensus that the rules must exist!

Outline

what is syntax?

where do grammars come from?

how can a computer find a sentence's structure?

Treebanks

- Collections of natural text that are annotated according to a particular syntactic theory
  - Usually created by linguistic experts
  - Ideally as large as possible
  - Theories are usually coarsely divided into *constituent/ phrase* or *dependency* structure

Formalisms

- **Phrase-structure** and **dependency** grammars
  - Phrase-structure: encodes the phrasal components of language
  - Dependency grammars encode the relationships between words

Penn Treebank (1993)

https://catalog.ldc.upen

- ABOUT
- MEMBERS
- COMMUNICATIONS
- LANGUAGE RESOURCES
- Data
  - Obtaining Data
  - Catalog
  - By Year
  - Top Ten Corpora
  - Projects
  - Search
  - Memberships
  - Data Scholarships
- Tools
- Papers
- LR Wiki
- DATA MANAGEMENT
- COLLABORATIONS

Home › Language Resources › Data

Treebank-3

Item Name:	Treebank-3
Author(s):	Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, Ann Taylor
LDC Catalog No.:	LDC99T42
ISBN:	1-58563-163-9
ISLRN:	141-282-691-413-2
Member Year(s):	1999
DCMI Type(s):	Text
Data Source(s):	telephone speech, newswire, microphone speech, transcribed speech, varied
Project(s):	TIDES, GALE
Application(s):	parsing, natural language processing, tagging
Language(s):	English
Language ID(s):	eng
License(s):	LDC User Agreement for Non-Members
Online Documentation:	LDC99T42 Documents
Licensing Instructions:	Subscription & Standard Members, and Non-Members
Citation:	Marcus, Mitchell, et al. Treebank-3 LDC99T42. Web Download. Philadelphia: Linguistic Data Consortium, 1999.
Related Works:	View

Introduction

This release contains the following Treebank-2 Material:

- One million words of 1989 Wall Street Journal material annotated in Treebank II style.
- A small sample of ATIS-3 material annotated in Treebank II style.
- A fully tagged version of the Brown Corpus.

and the following new material:

- Switchboard tagged, dysfluency-annotated, and parsed text
- Brown parsed text

The Treebank bracketing style is designed to allow the extraction of simple predicate/argument structure. Over one million words of text are provided with this bracketing applied.

Data

The Penn Treebank

- Syntactic annotation of a million words of the 1989 Wall Street Journal, plus other corpora (released in 1993)
  - (Trivia: People often discuss “The Penn Treebank” when the mean the WSJ portion of it)

The Penn Treebank

- Syntactic annotation of a million words of the 1989 Wall Street Journal, plus other corpora (released in 1993)
  - (Trivia: People often discuss “The Penn Treebank” when the mean the WSJ portion of it)
- Contains 74 total tags: 36 parts of speech, 7 punctuation tags, and 31 phrasal constituent tags, plus some relation markings

The Penn Treebank

- Syntactic annotation of a million words of the 1989 Wall Street Journal, plus other corpora (released in 1993)
  - (Trivia: People often discuss “The Penn Treebank” when the mean the WSJ portion of it)
- Contains 74 total tags: 36 parts of speech, 7 punctuation tags, and 31 phrasal constituent tags, plus some relation markings
- Was the foundation for an entire field of research and applications for over twenty years

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

https://commons.wikimedia.org/wiki/File:PierreVinken.jpg

Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD        S years) )
        (JJ old)
      (, ,) )
    (VP (MD wil )
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

x 49,208

https://commons.wikimedia.org/wiki/File:PierreVinken.jpg

Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone

Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:

Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP

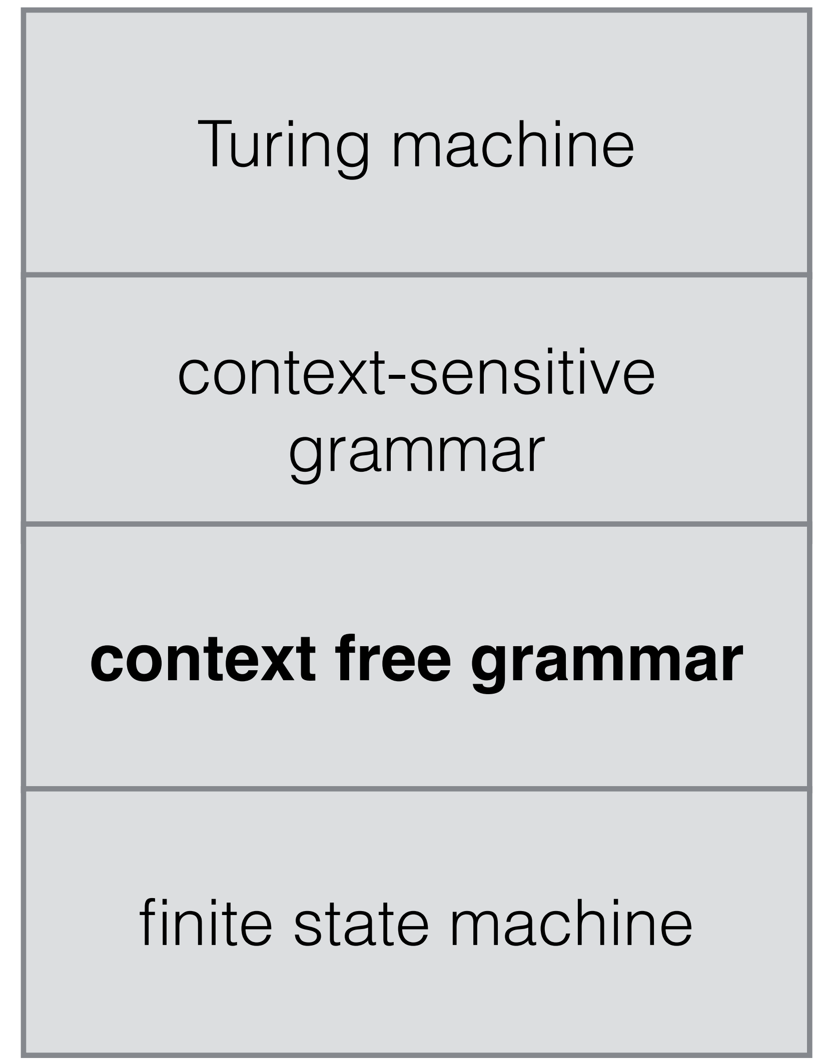
Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:

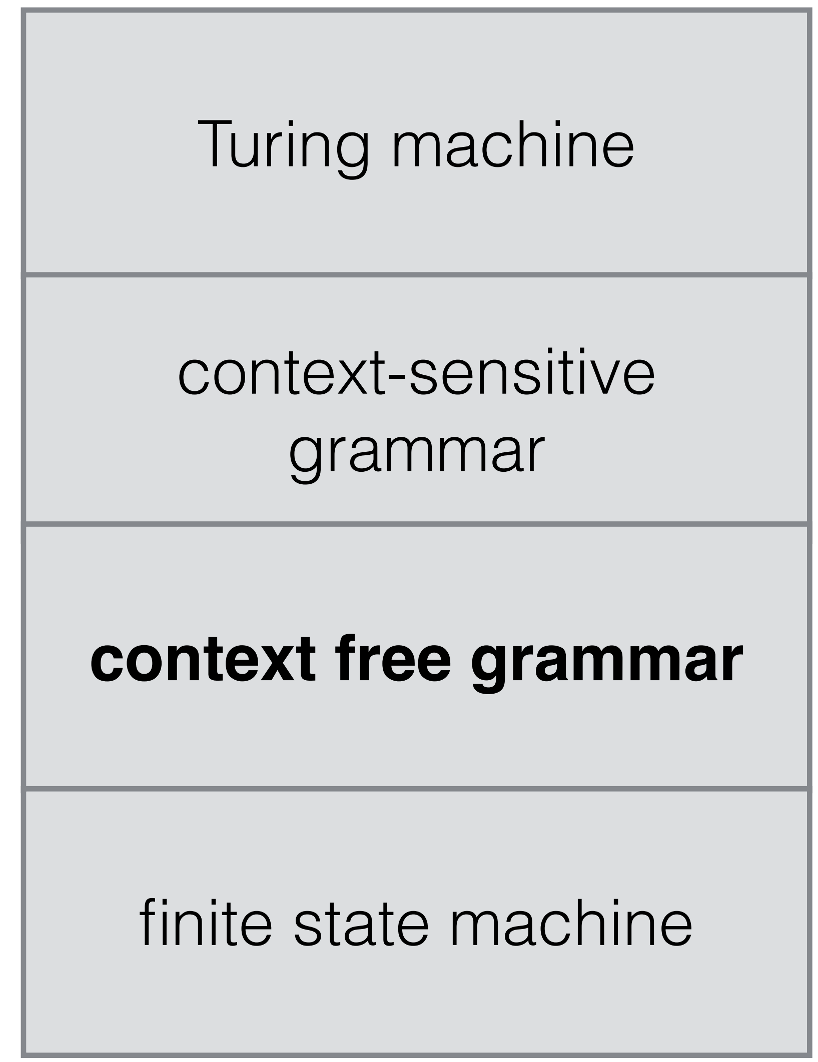
Chomsky formal language hierarchy



Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal

Chomsky formal language hierarchy



Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal
    - Replace it with

Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal
    - Replace it with
    - Recurse

Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal
    - Replace it with
    - Recurse
- Terminates when there are no more nonterminals

Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

TOP

TOP → S

TOP

S

TOP → S

S → VP

TOP

S

VP

TOP → S

S → VP

VP → (VB→halt) NP PP

TOP

S

VP

halt NP PP

TOP → S

S → VP

VP → (VB→halt) NP PP

NP → (DT The)
(JJ→market-jarring)
(CD→25)

TOP

S

VP

halt NP PP

halt **The market-jarring 25** PP

TOP → S

S → VP

VP → (VB→halt) NP PP

NP → (DT The)
(JJ→market-jarring)
(CD→25)

PP → (IN→at) NP

TOP	TOP → S
S	S → VP
VP	VP → (VB→halt) NP PP
halt NP PP	NP → (DT The) (JJ→market-jarring) (CD→25)
halt **The market-jarring 25** PP	PP → (IN→at) NP
halt The market-jarring 25 **at NP**	NP → (DT→the) (NN→bond)

TOP	TOP → S
S	S → VP
VP	VP → (VB→halt) NP PP
halt NP PP	NP → (DT The) (JJ→market-jarring) (CD→25)
halt **The market-jarring 25** PP	PP → (IN→at) NP
halt The market-jarring 25 **at NP**	NP → (DT→the) (NN→bond)
halt The market-jarring 25 at **the bond**	

TOP	TOP → S
S	S → VP
VP	VP → (VB→halt) NP PP
halt NP PP	NP → (DT The) (JJ→market-jarring) (CD→25)
halt **The market-jarring 25** PP	PP → (IN→at) NP
halt The market-jarring 25 **at NP** (NN→bond)	NP → (DT→the)
halt The market-jarring 25 at **the bond**	

(TOP (S (VP (VB halt) (NP (DT The) (JJ market-jarring) (CD 25)) (PP (IN at) (NP (DT the) (NN bond))))))

A problem with the Penn Treebank

- One language, English
  - Represents a very narrow typology (e.g., little morphology)
  - Consider the tags we looked at before
    - nouns: NN, NNS, NNP, NNPS
    - adverbs: RB, RBR, RBS, RP
    - verbs: VB, VBD, VBG, VBN, VBP, VBZ
  - How well will these generalize to other languages?
-

Dependency Treebanks (2012)

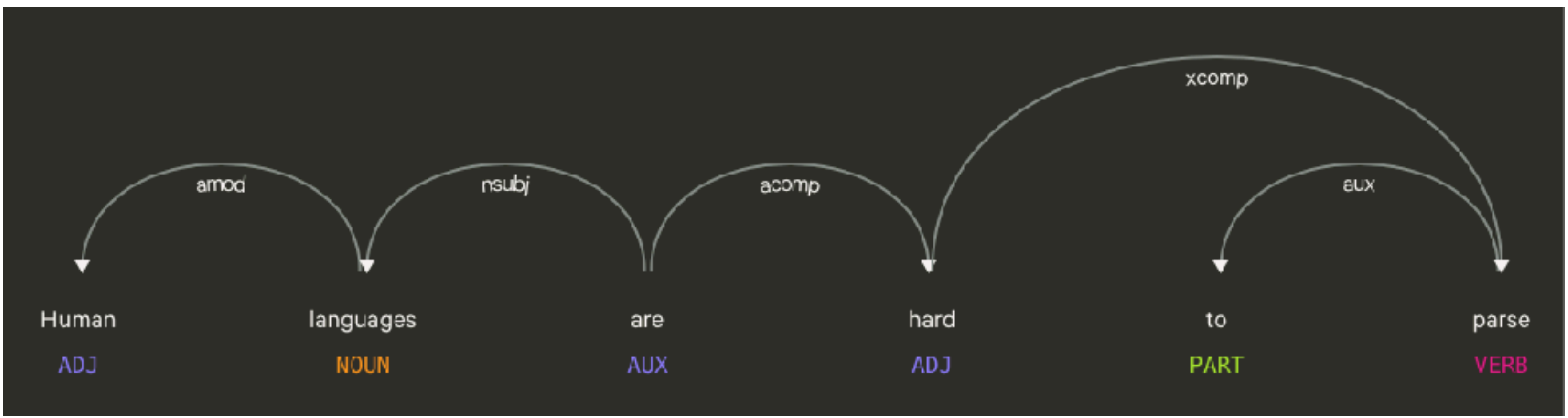
Universal Dependencies

- Dependency trees annotated across languages in a consistent manner

https://universaldependencies.org

Example

- Instead of encoding phrase structure, it encodes dependencies between words
- Often more directly encodes information we care about (i.e., *who* did *what* to *whom*)



Guiding principles

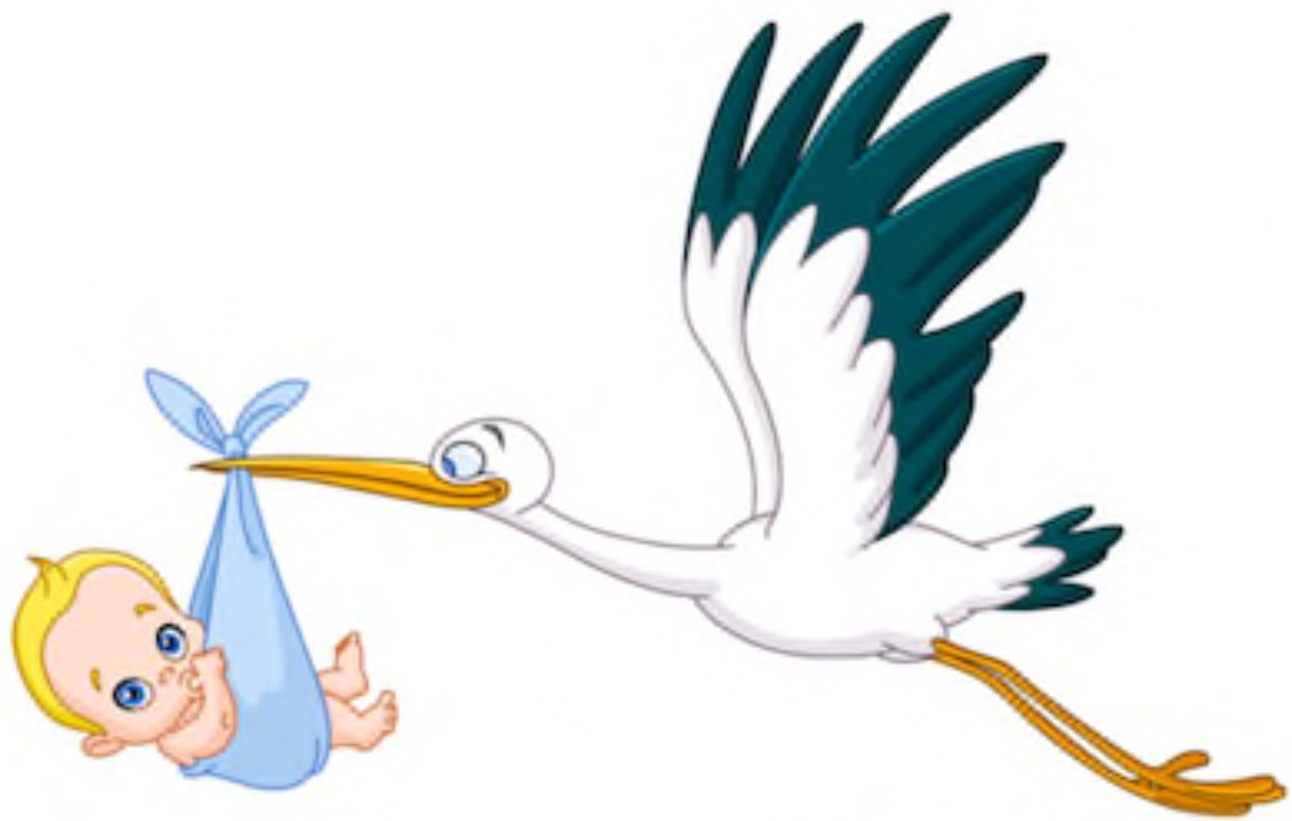
- Works for individual languages
- Suitable across languages
- Easy to use when annotating
- Easy to parse quickly
- Understandable to laypeople
- Usable by downstream tasks

https://universaldependencies.org/introduction.html

Universal Dependencies

- Parts of speech
  - open class
    - ADJ, ADV, INTJ, NOUN, PROPN, VERB
  - closed class
    - ADP, AUX, CCONJ, DET, NUM, PART, PRON, SCONJ
  - other
    - PUNCT, SYM, X

Where do grammars come from?



https://www.shutterstock.com/image-vector/stork-carrying-baby-boy-133823486

Where do grammars come from?

- Treebanks!
- Given a treebank, and a formalism, we can learn statistics by counting over the annotated instances

Probabilities

- For example, a context-free grammar
  - S → NP , NP VP . [0.002]
  - NP → NNP NNP [0.037]
  - , → , [0.999]
  - NP → * [X]
  - VP → VB NP [0.057]
  - NP → PRP$ NN [0.008]
  - . → . [0.987]
- Probabilities given as $P(X) = \sum_{X' \in N} \frac{P(X)}{P(X')}$

Summary

Grammars are learned from Treebanks

where do grammars come from?

Treebanks are annotated according to a particular theory or formalism

Outline

what is syntax?

where do grammars come from?

how can a computer find a sentence's structure?

Formal Language Theory

- Consider the claims underlying our grammar-based view of language
  1. Sentences are either in or out of a language
  2. Sentences have an invisible hidden structure

Formal Language Theory

- Consider the claims underlying our grammar-based view of language
  1. Sentences are either in or out of a language
  2. Sentences have an invisible hidden structure
- We can generalize this discussion to make a connection between natural and other kinds of languages

Formal Language Theory

- Consider the claims underlying our grammar-based view of language
  1. Sentences are either in or out of a language
  2. Sentences have an invisible hidden structure
- We can generalize this discussion to make a connection between natural and other kinds of languages
- Consider, for example, *computer programs*
  - They either compile or don't compile
  - Their structure determines their interpretation

Formal Language Theory

- Generalization: define a **language** to be a set of strings under some alphabet, Σ
  - e.g., the set of valid English sentences (where the "alphabet" is English words), or the set of valid Python programs

Formal Language Theory

- Generalization: define a **language** to be a set of strings under some alphabet, Σ
  - e.g., the set of valid English sentences (where the "alphabet" is English words), or the set of valid Python programs
- Formal Language Theory provides a common framework for studying properties of these languages, e.g.,
  - Is this file a valid C++ program? A valid Czech sentence?
  - What is the structure?
  - How hard / time-consuming is it to answer these questions?

The Chomsky Hierarchy

- Definitions: given
  - an alphabet (Σ),
  - terminal symbols, e.g., $a \in \Sigma$
  - nonterminal symbols, e.g., {S, N, A, B}
  - α, β, γ, strings of terminals and/or nonterminals

Type	Rules	Name	Recognized by
3	$A \rightarrow aB$	Regular	Regular expressions
2	$A \rightarrow \alpha$	Context-free	Pushdown automata
1	$\alpha A \beta \rightarrow \alpha \gamma \beta$	Context-sensitive	Linear-bounded Turing machine
0	$\alpha A \beta \rightarrow \gamma$	Recursively enumerable	Turing Machines

Problems

- What is the value?

  (5 + 7) * 11

- Who did what to whom?

  Him the Almighty hurled

  Dipanjan taught Johnmark

*If we have a grammar, we can answer these with **parsing***

Parsing

- If the grammar has certain properties (Type 2 or 3), we can efficiently answer two questions with a **parser**
  - Is the sentence in the language of the parser?
  - What is the structure above that sentence?

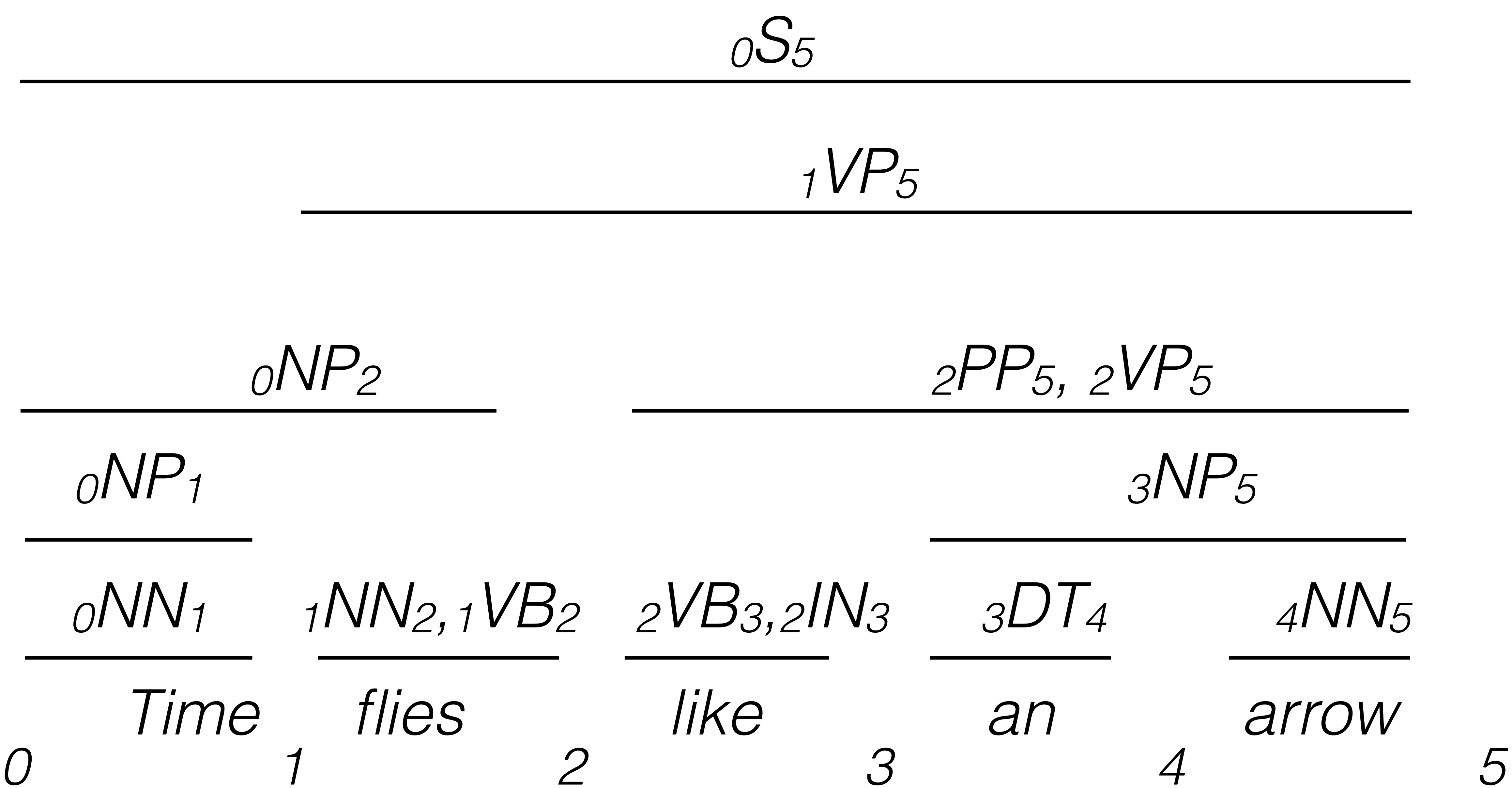
Algorithms

- The **CKY algorithm** for parsing with constituency grammars
- **Transition-based** parsing with dependency grammars

Chart parsing for constituency grammars

- Maintains a chart of nonterminals spanning words, e.g.,
  - NP over words 1..4 and 2..5
  - VP over words 4..6 and 4..8
  - etc

Chart parsing for constituency grammars



CKY algorithm

- How do we produce this chart? Cocke-Younger-Kasami (CYK/CKY)
- Basic idea is to apply rules in a bottom-up fashion, applying all rules, and (recursively) building larger constituents from smaller ones
- Input: sentence of length N

```
for width in 2..N
    for begin i in 1..{N - width}
    j = i + width
        for split k in {i + 1}..{j - 1}
            for all rules A → B C
                create iAj if iBk and kCj
```

CKY algorithm

	Time		*flies*		*like*		*an*		*arrow*	
0		*1*		*2*		*3*		*4*		*5*

CKY algorithm

NN	NN,VB	VB,IN	DT	NN
Time	flies	like	an	arrow

0 1 2 3 4 5

CKY algorithm

$NP \rightarrow NN$ | | | $NP \rightarrow DT\ NN$ (spans 3–5)

$NP \rightarrow NN$			$NP \rightarrow DT\ NN$	
NN	*NN,VB*	*VB,IN*	*DT*	*NN*
Time	*flies*	*like*	*an*	*arrow*

0 *1* *2* *3* *4* *5*

CKY algorithm

$NP \rightarrow NN\ NN$ $\qquad$ $PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN$ $\qquad$ $NP \rightarrow DT\ NN$

NN	*NN,VB*	*VB,IN*	*DT*	*NN*
Time	*flies*	*like*	*an*	*arrow*

0 *1* *2* *3* *4* *5*

CKY algorithm

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$ $\quad$ $PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN	NN,VB	VB,IN	DT	NN
Time	*flies*	*like*	*an*	*arrow*

0 1 2 3 4 5

CKY algorithm

$VP \rightarrow VB\ PP$

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN	NN,VB	VB,IN	DT	NN
Time	*flies*	*like*	*an*	*arrow*

0 1 2 3 4 5

CKY algorithm

$S \rightarrow {}_0NP_1\ {}_1VP_5$

$VP \rightarrow VB\ PP$

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$

$PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN	*NN,VB*	*VB,IN*	*DT*	*NN*
Time	*flies*	*like*	*an*	*arrow*

0 1 2 3 4 5

CKY algorithm

$S \rightarrow {}_0NP_2\ {}_2VP_5$

$S \rightarrow {}_0NP_1\ {}_1VP_5$

$VP \rightarrow VB\ PP$

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$

$PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN	NN,VB	VB,IN	DT	NN
Time	*flies*	*like*	*an*	*arrow*

0 1 2 3 4 5

CKY algorithm

- Termination: is there a chart entry at ${}_0S_N$?
  - ✓ string is in the language
  - Obtain the structure by following backpointers
  - Not covered: adding probabilities to rules to resolve amgibuities

Dependency parsing

- The situation is different in many ways
  - We’re no longer building labeled constituents
  - Instead, we’re searching for word dependencies

Dependency parsing

- The situation is different in many ways
  - We're no longer building labeled constituents
  - Instead, we're searching for word dependencies
- This is accomplished by a stack-based **transition parser**
  - Repeatedly (a) shift a word onto the stack or (b) create a LEFT or RIGHT dependency from the top two words

ROOT human languages are hard to parse

step	stack	words	action	relation

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	
6	[are,hard]	[to,parse]	SHIFT	

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	
6	[are,hard]	[to,parse]	SHIFT	
7	[are,hard,to]	[parse]	SHIFT	

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	
6	[are,hard]	[to,parse]	SHIFT	
7	[are,hard,to]	[parse]	SHIFT	
8	[are,hard,to,parse]	[]	LEFTARC	to←parse

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	
6	[are,hard]	[to,parse]	SHIFT	
7	[are,hard,to]	[parse]	SHIFT	
8	[are,hard,to,parse]	[]	LEFTARC	to←parse
9	[are,hard,parse]	[]	RIGHTARC	hard→parse

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	
6	[are,hard]	[to,parse]	SHIFT	
7	[are,hard,to]	[parse]	SHIFT	
8	[are,hard,to,parse]	[]	LEFTARC	to←parse
9	[are,hard,parse]	[]	RIGHTARC	hard→parse
10	[are,hard]	[]	RIGHTARC	are→hard

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	
6	[are,hard]	[to,parse]	SHIFT	
7	[are,hard,to]	[parse]	SHIFT	
8	[are,hard,to,parse]	[]	LEFTARC	to←parse
9	[are,hard,parse]	[]	RIGHTARC	hard→parse
10	[are,hard]	[]	RIGHTARC	are→hard
11	[are]	[]	RIGHTARC	ROOT→are

ROOT human languages are hard to parse

step	stack	words	action	relation
0	[]	[human,langs,are,hard,to,parse]	SHIFT	
1	[human]	[langs,are,hard,to,parse]	SHIFT	
2	[human,langs]	[are,hard,to,parse]	LEFTARC	human←langs
3	[langs]	[are,hard,to,parse]	SHIFT	
4	[langs,are]	[hard,to,parse]	LEFTARC	langs←are
5	[are]	[hard,to,parse]	SHIFT	
6	[are,hard]	[to,parse]	SHIFT	
7	[are,hard,to]	[parse]	SHIFT	
8	[are,hard,to,parse]	[]	LEFTARC	to←parse
9	[are,hard,parse]	[]	RIGHTARC	hard→parse
10	[are,hard]	[]	RIGHTARC	are→hard
11	[are]	[]	RIGHTARC	ROOT→are
12	[]	[]	DONE	

Unanswered questions

- How do we score rules (for constituency parsing) and actions and relations (for dependency parsing)?
  - Probabilities can be read from Treebanks
  - Actions can be informed by feature selection
- How do we know the right path to take?
  - We can try multiple paths using beam search
  - We get lots of savings via dynamic programming

Summary

For context-free grammars, the (weighted) CKY algorithm can be used to find the most probable (maximum a posteriori) tree given a certain grammar

how can a computer find a sentence's structure?

For dependency grammars, the most popular approach is a variation of transition-based parsers

Resources

- Demos:
  - AllenNLP: https://demo.allennlp.org
  - Berkeley Neural Parser: https://parser.kitaev.io
  - Spacy dependency parser: https://explosion.ai/demos/displacy

Outline

what is syntax?

where do grammars come from?

how can a computer find a sentence's structure?

Outline

what is syntax?

where do grammars come from?

how can a computer find a sentence's structure?

the study of the internal structure of sentences (in natural and synthetic languages)

Outline

what is syntax?

where do grammars come from?

how can a computer find a sentence's structure?

the study of the internal structure of sentences (in natural and synthetic languages)

they are created by linguists, usually under particular grammatical theories

Outline

what is syntax?	where do grammars come from?	how can a computer find a sentence's structure?
the study of the internal structure of sentences (in natural and synthetic languages)	they are created by linguists, usually under particular grammatical theories	train a grammar from a treebank and then apply that grammar to new sentences using parsing algorithms