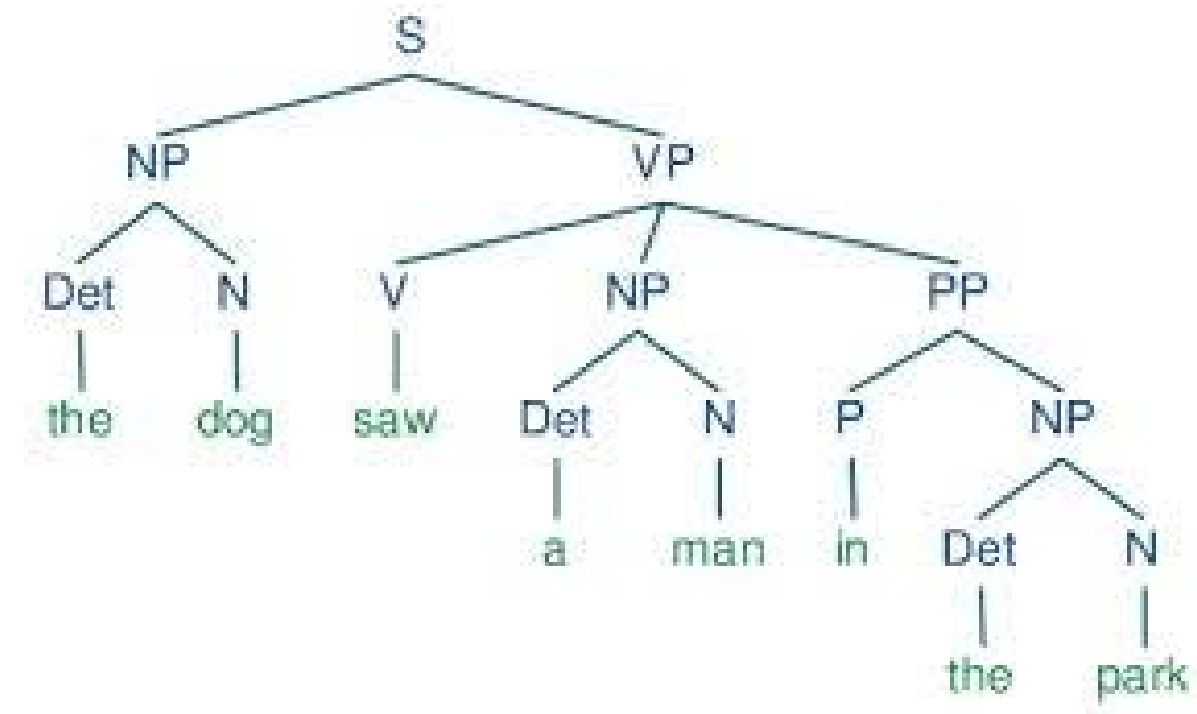


COMP90042 LECTURE 5

CONTEXT-FREE GRAMMARS

SYNTACTIC CONSTITUENTS

- Sequential models like HMMs assume entirely flat structure
- But language clearly isn't like that

[*A man*] [*saw* [*a dog*] [*in* [*the park*]]]

- Words group together to form syntactic constituents
  - Can be replaced, or moved around *as a unit*
- Grammars are allow us to formalize these intuitions
  - Symbols correspond to syntactic constituents

OUTLINE

- The context-free grammar formalism
- Parsing with CFGs
- Representing English with CFGs

BASICS OF CONTEXT-FREE GRAMMARS

- Symbols
  - Terminals: words such as *book*
  - Non-terminal: syntactic labels such as NP or NN
- Productions (rules)
  - Exactly one non-terminal on left-hand side (LHS)
  - An ordered list of symbols on right-hand side (RHS)

A SIMPLE GRAMMAR

Terminal symbols: *rat, the, ate, cheese*

Non-terminal symbols: S, NP, VP, DT, VBD, NN

Productions:

S $\rightarrow$ NP VP

NP $\rightarrow$ DT NN

VP $\rightarrow$ VBD NP

DT $\rightarrow$ *the*

NN $\rightarrow$ *rat*

NN $\rightarrow$ *cheese*

VBD $\rightarrow$ *ate*

GENERATING SENTENCES WITH CFGS

Always start with S (the sentence/start symbol)

S

Apply rule with S on LHS (S $\rightarrow$ NP VP), i.e substitute RHS

NP VP

Apply rule with NP on LHS (NP $\rightarrow$ DT NN)

DT NN VP

Apply rule with DT on LHS (DT $\rightarrow$ *the*)

***the* NN VP**

Apply rule with NN on LHS (NN $\rightarrow$ *rat*)

***the rat* VP**

GENERATING SENTENCES WITH CFGS

Apply rule with VP on LHS (VP $\rightarrow$ VBD NP)

***the rat* VBD NP**

Apply rule with VBD on LHS (VBD $\rightarrow$ *ate*)

***the rat ate* NP**

Apply rule with NP on LHS (NP $\rightarrow$ DT NN)

***the rat ate* DT NN**

Apply rule with DT on LHS (DT $\rightarrow$ *the*)

***the rat ate the* NN**

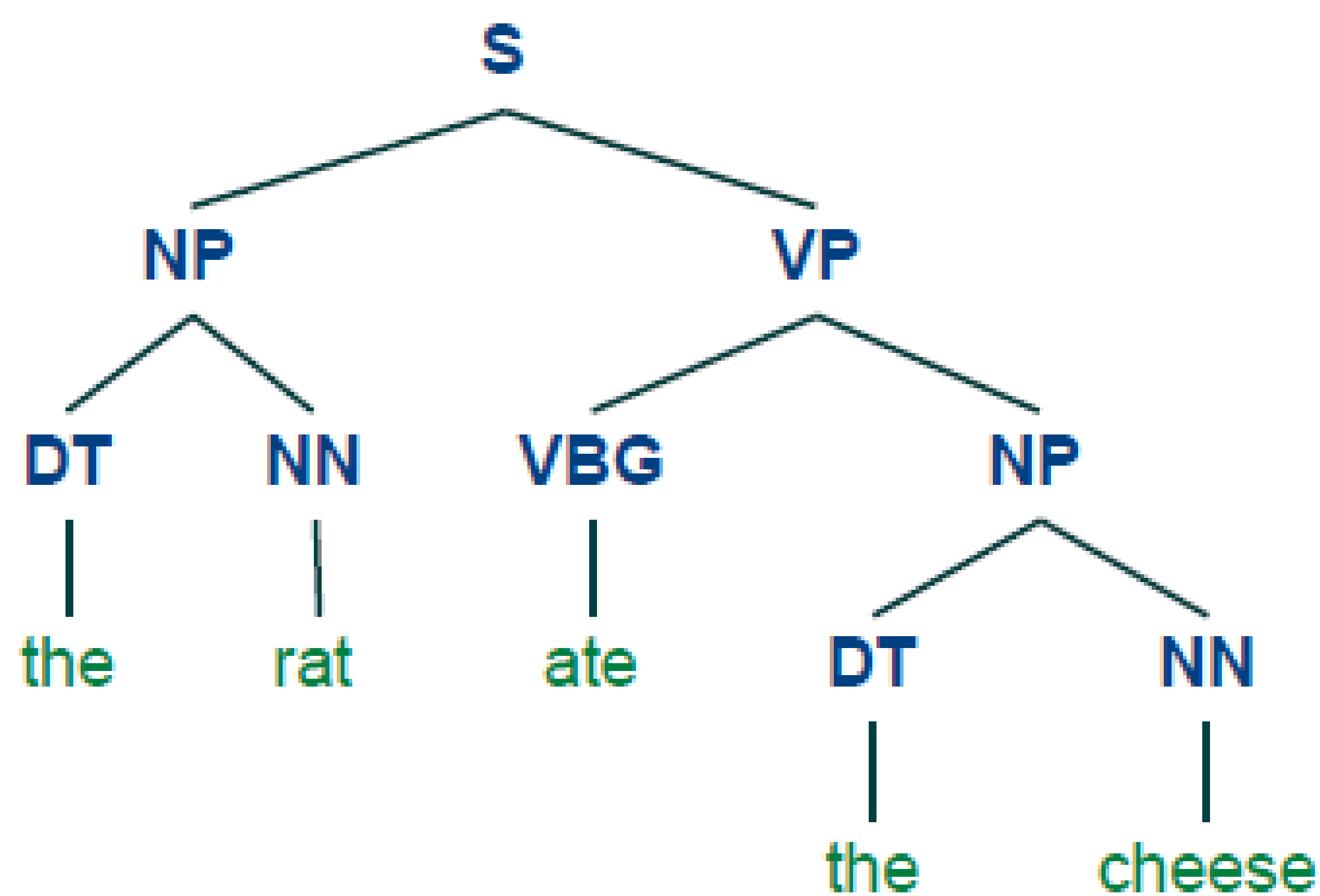
Apply rule with NN on LHS (NN $\rightarrow$ *cheese*)

the rat ate the cheese

CFG TREES

- Generation corresponds to a syntactic tree
- Non-terminals are internal nodes
- Terminals are leaves

(S (NP (DT the)
(NN rat))
(VP (VBG ate)
(NP (DT the)
(NN cheese))))



PARSING CFGS

- Parsing: given string, identify possible structures
- Brute force search is intractable for non-trivial grammars
  - Good solutions use dynamic programming
- Two general strategies
  - Bottom-up
    - Start with words, work up towards S
    - CYK parsing
  - Top-down
    - Start with S, work down towards words
    - Earley parsing

THE CYK PARSING ALGORITHM

- Convert grammar to Chomsky Normal Form (CNF)
- Fill in a parse table
- Use table to derive parse
- Covert result back to original grammar

CONVERT TO CNF

- Change grammar so all rules of form A $\rightarrow$ BC or A $\rightarrow$ a
- Step 1: Convert A $\rightarrow$ Bc A $\rightarrow$ BC, C $\rightarrow$ c
  - Not usually necessary in POS-based grammars
- Step 2: Convert A $\rightarrow$ BCD to A $\rightarrow$ BX, X $\rightarrow$ CD
  - Usually necessary, but not for our toy grammar

PARSE TABLE

the	*rat*	*ate*	*the*	*cheese*
DT [0,1]	NP [0,2]	[0,3]	[0,4]	S [0,5]
	NN [1,2]	[1,3]	[1,4]	[1,5]
		VBD [2,3]	[2,4]	VP [2,5]
			DT [3,4]	NP [3,5]
				NN [4,5]

S → NP VP

NP → DT NN

VP → VBD NP

DT → *the*

NN → *rat*

NN → *cheese*

VBD → *ate*

CYK: RETRIEVING THE PARSES

- S in the top-left corner of parse table indicates success
- To get parse(s), follow pointers back for each match
- Convert back from CNF by removing new non-terminals

EARLEY PARSING

- Create a chart of applied rules (edges)
  - Length of chart = length of sentence + 1
  - Edges are rules which are augmented with
    - A dot which indicates how much of the rule has been satisfied
    - A range over which it has been applied so far
    - E.g. $S \rightarrow NP \bullet VP\ [0,2]$
- Chart is filled from left to right with 3 operations
  - Predictor
  - Scanner
  - Completer

THE CHART

	the	*rat*	*ate*	*the*	*cheese*
Predictor	S → • NP VP [0,0] NP → • DT NN [0,0]	VP → • VB NP [2,2]	NP → • DT NN [3,3]		
Scanner	DT → the • [0,1]	NN → rat • [1,2]	VB → ate • [2,3]	DT → the • [3,4]	NN → cheese • [4,5]
Completer	NP→ DT • NN [0,1]	NP→ DT NN • [0,2] S → NP • VP [0,2]	VP → VB • NP [2,3]	NP → DT • NN [3,4]	NP→ DT NN • [3,5] VP → VB NP • [2,5] S→ NP VP • [0,5]

EARLEY: RETRIEVING THE PARSES

- Completed S rule covering sentence indicates success
- To get parse(s), follow pointers back for each completion

TOY GRAMMARS TO REAL GRAMMARS

- Toy grammars with handful of productions good for demonstration or extremely limited domains
- For real texts, we need real grammars
- Hundreds or thousands of production rules

KEY CONSTITUENTS IN PENN TREEBANK

- Sentence (S)
- Noun phrase (NP)
- Verb phrase (VP)
- Prepositional phrase (PP)
- Adjective phrase (AdjP)
- Adverbial phrase (AdvP)
- Subordinate clause (SBAR)

BASIC ENGLISH SENTENCE STRUCTURES

- Declarative sentences ($S \rightarrow NP\ VP$)
  - E.g. *The rat ate the cheese*
- Imperative sentences ($S \rightarrow VP$)
  - E.g. *Eat the cheese!*
- Yes/no questions ($S \rightarrow VB\ NP\ VP$)
  - E.g. *did the rat eat the cheese?*
- *Wh*-subject-questions ($S \rightarrow WH\ VP$)
  - *Who ate the cheese?*
- *Wh*-object-questions ($S \rightarrow WH\ VB\ NP\ VP$)
  - *What did the rat eat?*

ENGLISH NOUN PHRASES

- Pre-modifiers
  - DT, CD, ADJP, NNP, NN
  - E.g. *the two very best Philly cheese steaks*
- Post-modifiers
  - PP, VP, SBAR
  - A call from Mom coming today that I don't want to miss

$$NP \rightarrow (DT)\ (CD)\ (ADJP)\ NN \mid NNP+\ PP^*\ (VP)\ (SBAR)$$

$$NP \rightarrow PRP$$

VERB PHRASES

- Auxiliaries
  - MD, AdvP, VB, TO
  - E.g *should really have tried to wait*

VP → MD | VB | TO (AdvP) VP

- Arguments and adjuncts
  - NP, PP, SBAR, VP, AdvP
  - E.g *told him yesterday that I was ready*
  - E.g. *gave John a gift for his birthday to make amends*

VP → VB (NP) (NP) PP* AdvP* (VP) (SBAR)

OTHER CONSTITUENTS

- Prepositional phrase
  - PP → IN NP (*in the house*)
- Adjective phrase
  - AdjP → (AdvP) JJ (*really nice*)
- Adverb phrase
  - AdvP → (AdvP) RB (*not too well*)
- Subordinate clause
  - SBAR → (IN) S (*since I came here*)
- Coordination
  - NP → NP CC NP; VP → VP CC VP; etc. (*Jack and Jill*)
- Complex sentences
  - S → S SBAR; S → SBAR , S; etc. (*if he goes, I'll go*)

A FINAL WORD

- Context-free grammars can represent linguistic structure
- There are relatively fast dynamic programming algorithms to retrieve this structure
- But what about ambiguity?
  - Extreme ambiguity will slow down parsing
  - If multiple possible parses, which is best?

REQUIRED READING

- J&M2 Ch. 12.1-12.5, Ch. 13.1-13.4