LG 467 Computers in Linguistics

[1-2021] Topic 6: Parsing

Sakol Suethanapornkul

From HMMs to syntax

We have seen some simple ways of dealing with syntax:

- Markov models capture surface properties of syntax
  - N-grams (VMM): *In the end…*
  - HMM: *IN DT NN*
- But this isn't enough
  - Long distance dependencies in languages
  - wake [the old man] up

From HMMs to syntax

Take a look at this example:

Code 9.1

```
import nltk

nltk.pos_tag(nltk.word_tokenize("Let's wake up at
six"))
# [('Let', 'VB'), ("'s", 'POS'), ('wake', 'VB'),
# (‘up’, 'RP'), ('at', 'IN'), ('six', 'CD')]

nltk.pos_tag(nltk.word_tokenize("Let's wake that old
man up at six"))
# [('Let', 'VB'), ("'s", 'POS'), ('wake', 'VB'),
# (‘that', 'IN'), ('old', 'JJ'), ('man', 'NN'),
# (‘up’, 'RB'), ('at', 'IN'), ('six', 'CD')]
```

Describing structure of sentences

Constituent: word or group of words that function as a single unit

[That man] is my friend

[The guy in a blue shirt] loves cookies

*That is my friend

คุณครูแอบกิน[ยาถ่าย]

แม่ไป[ตลาดสดหน้าปากซอย]

*คุณครูแอบกินถ่าย

Describing structure of sentences

Constituent: word or group of words that function as a single unit

[on September tenth] I'll be moving

I'll be moving [on September tenth]

*On September I'll be moving tenth

[ตอนเช้าวันพรุ่งนี้]เราจะตื่นมาตักบาตร

เราจะตื่นมาตักบาตร[ตอนเช้าวันพรุ่งนี้]

*ตอนเช้าเราจะตื่นมาตักบาตรพรุ่งนี้

Context-Free Grammar (CFG)

CFG: formal system for modeling constituent structure

- A set of (de)composition rules over a set of symbols
- Sample rules:
  - NP → DT NN
  - NP → NNP
  - DT → the
  - NN → house | mouse

Rules in the form of $A \rightarrow \beta$

Context-Free Grammar (CFG)

CFG: formal system for modeling constituent structure

- A set of (de)composition rules over a set of symbols
- Sample rules:
  - NP → DT NN
  - NP → NNP
  - DT → the
  - NN → house | mouse

Terminals (often = tokens)

Context-Free Grammar (CFG)

CFG: formal system for modeling constituent structure

- A set of (de)composition rules over a set of symbols

- Sample rules:
  - NP → DT NN
  - NP → NNP
  - DT → the
  - NN → house | mouse

Non-terminals ("abstraction symbol")

Context-Free Grammar (CFG)

CFG: formal system for modeling constituent structure

- A set of (de)composition rules over a set of symbols

- Sample rules:
  - NP → DT NN
  - NP → NNP
  - DT → the
  - NN → house | mouse

Left side: Single non-terminal symbol

Right side: 1+ Non-terminal/terminal

Context-Free Grammar (CFG)

CFG: formal system for modeling constituent structure

- A set of (de)composition rules over a set of symbols

- Sample rules:
  - NP → DT NN
  - NP → NNP
  - DT → the
  - NN → house | mouse

Rules are hierarchically embedded

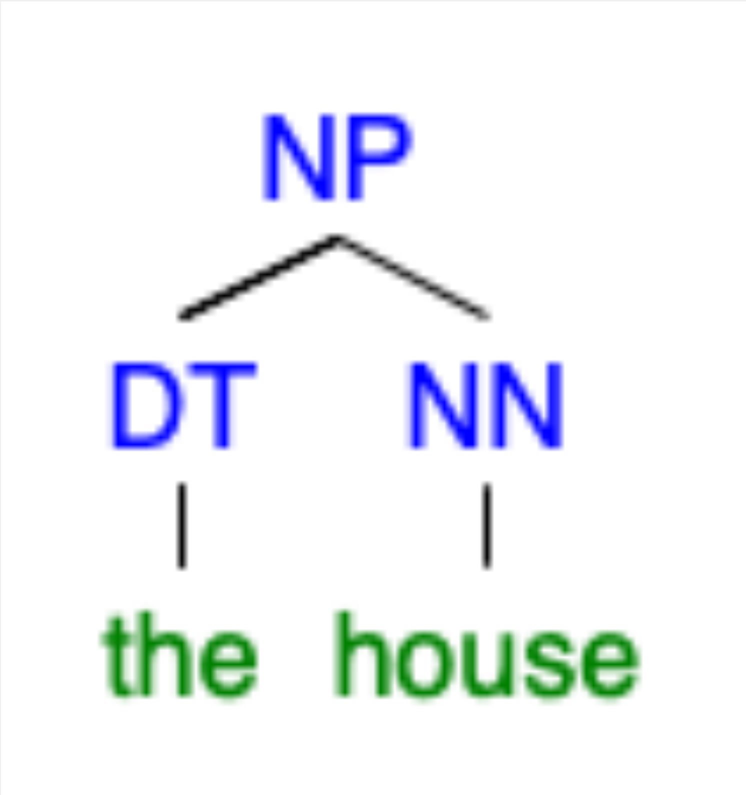
Context-Free Grammar (CFG)

CFG can be thought of in two ways:

- Device for **generating sentences**
- Device for **assigning a structure** to a given sentence

- NP → DT NN
- NP → NNP
- DT → the
- NN → house | mouse

"the house" derived from NP



Context-Free Grammar (CFG)

In CFG, a starting symbol must be selected

- Each grammar: one designated start symbol **S**
- Because CFG is used to define sentences, S = "sentence" node

Some parts of language can't be captured by context-free grammar rules

Context-Free Grammar (CFG)

Some examples:

- S → NP VP
- VP → V NP
- VP → V
- NP → DT NN
- V → eats
- NN → mouse | house
- DT → the

Now we can generate/parse:

Context-Free Grammar (CFG): Exercise

Let's try to extract context free rules from a sentence:

- Every sentence has S at the top
- S breaks down into phrases
- Phrases decompose into our POS tags/other phrases
- POS tags lead to tokens

Context-Free Grammar (CFG): Exercise

Sentence: They really go above and beyond!

Context-Free Grammar (CFG): Exercise

A possible analysis (from the English Web Treebank):

Question: What are the context-free grammar rules?

Context-Free Grammar (CFG): Exercise

Rules:

- S → NP ADVP VP
- NP → PRP
- VP → VBP ADVP
- ADVP → RB
- ADVP → RB CC RB

Context-Free Grammar (CFG)

If we use traditional V for verbs, for instance:

- S → NP VP
- VP → **V** NP
- NP → DT N
- **V** → bite
- N → dog | boy

Question: What sentences can we generate?

Context-Free Grammar (CFG)

This is why tags are necessary:

- VP → **VBZ** NP
- VP → **VBP** NP
- VBP → bite
- VBZ → bites
- ....

We can have subject-verb agreement as part of our rule!

Context-Free Grammar (CFG)

For a formal definition, a CFG "G" is defined by four parameters:
G ≡ N, Σ, R, S (this is a "4-tuple")

N Set of **non-terminal** symbols

Σ Set of **terminal** symbols (not in N)

R Set of rules, each in the form $A \rightarrow \beta$, where $A \in N$, $\beta \in (\Sigma \cup N)^*$

S Designated start symbol

Treebanks

Thus far, we have hand-crafted rules to describe one sentence

- Can we build a grammar of language, taking into account its usage?
- Yes! Grammar can be induced from **annotated data** (like what we just did in our exercise)
- With hundreds of sentences, we can also note the frequencies with which each rule is used
- We can save these probabilities along with rules, which turns a CFG into a **Probabilistic Context-Free Grammar (PCFG)**

Treebanks

Treebank: a syntactically annotated corpus (= corpus of trees)

- each sentence in a corpus paired with a **parse tree**
- all sentences in treebank → grammar of language*
- major roles:
  - syntactic parsing: assign a parse tree to any sentence
  - linguistic research: investigate syntactic phenomena

Treebanks

The **Penn Treebank** Project: A 4.5-m. words of AmE (see Marcus, 1993)

- POS tags we saw in the previous unit
- syntactic parses (parenthesized notation; see next slide)
- CFG rules: WSJ corpus (1 million words)
  - 1,000,000 non-lexical rule tokens
  - ~17,500 distinct rule types

Treebanks

Parsed sentences from the Penn Treebank

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

Source: Figure 12.7 in Jurafsky & Martin [chapter 12]

Parsing

CFG rules from a treebank allow us to **process** actual sentences generated by humans

- several algorithms available (J & M Chapter 13)
- NTLK offers a few implementations

Compiling the grammar

Let's begin with CFG rules:

Code 9.2

```
from nltk import CFG

grammar_string = """
S -> NP VP
NP -> PRP
VP -> VBD
PRP -> 'I'
VBD -> 'cried'
"""

exercise_grammar = CFG.fromstring(grammar_string)

exercise_grammar
```

Parsing

Now that we have CFG rules in place, let's go ahead and parse:

Code 9.3

```
test = word_tokenize("I cried")

# Make a parser object with our grammar
parser = nltk.ChartParser(exercise_grammar)

# Parse
trees = parser.parse(test)

for tree in trees:
    print(tree)
```

Parsing

As our grammar gets bigger, it will have more rules:

Code 9.4

```
grammar_string = """
S -> NP VP
NP -> PRP
VP -> VP | VP ADVP | VBD | VBZ VBG RB
ADVP -> RB
PRP -> 'I' | 'He'
VBD -> 'cried'
VBZ -> 'is'
VBG -> 'falling'
RB -> 'fast'
"""
```

Parsing

As our grammar gets bigger, it will have more rules:

```
exercise_grammar = CFG.fromstring(grammar_string)

test = word_tokenize("He is falling fast")
parser = nltk.ChartParser(exercise_grammar)
trees = parser.parse(test)

for tree in trees:
    print(tree)
```

Code 9.4
[Continue]

Question: How many parses will we get?

Parsing

Take note of bracketing (color added to improve readability):

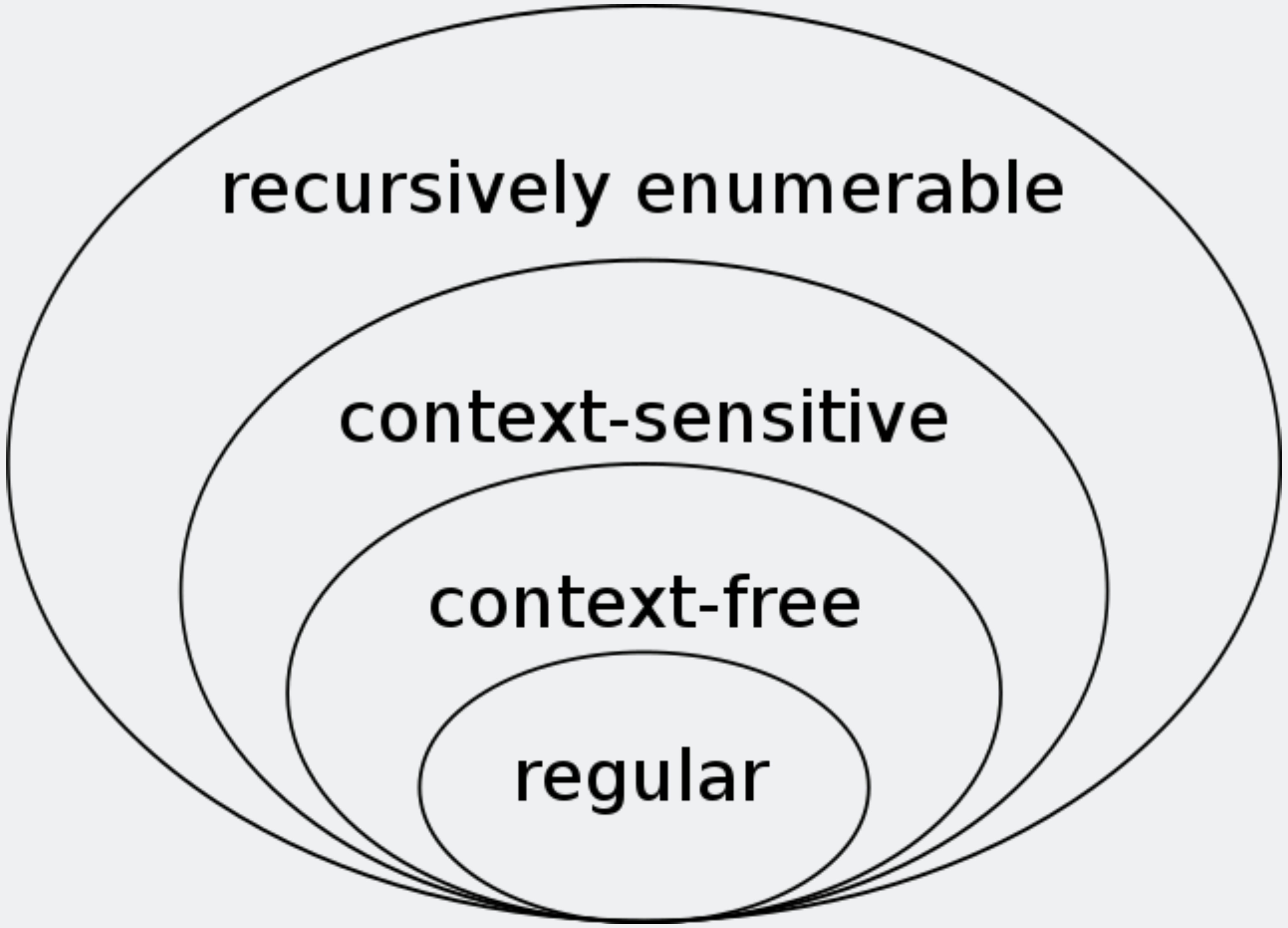
```
(S (NP (PRP He))
   (VP (VBZ is) (VBG falling) (RB fast)
   )
)

(S (NP (PRP He))
   (VP
      (VP (VBZ is) (VBG falling) (RB fast))
   )
)
```

Code 9.4
[Continue]

Language and complexity

Chomsky introduced a hierarchy of grammars in 1956:



Source: Wikipedia

Our plan next week...

- Parsing, Dependency Grammar
- Reading
  - J & M 3rd edition, Chapter 15