NLP

Introduction to NLP

241.

Syntax

Syntax

- Is language more than just a “bag of words”?
  - Grammatical rules apply to categories and groups of words, not individual words.
- Example
  - a sentence includes a subject and a predicate. The subject is a noun phrase and the predicate is a verb phrase.
  - Noun phrases:
    - The cat, Samantha, She
  - Verb phrase:
    - arrived, went away, had dinner
- When people learn a new word, they learn its syntactic usage.
  - Examples: wug (n), cluvious (adj) – use them in sentences
  - Hard to come up with made up words: forkle, vleer, etc. all taken.


TABLES ARE
FOR EATING
CUSTOMERS
ONLY
NO LOITERING

Defining Parts of Speech

- What do nouns typically have in common?
  - E.g., *can* be preceded by “the”.
- What about verbs?
  - Verbs can be preceded by “can’t”.
- Adjectives can come between “the” and a noun.
  - How is this different from grade school definitions?
- Determiners
  - a, the, many, no, five
- Prepositions
  - for, to, in, without, before

Constituents

- Constituents are continuous
- Constituents are non-crossing
  - if two constituents share one word, then one of them must completely contain the other.
- Each word is a constituent

Constituent Tests

- "coordination" test
  - She bought a bagel and three chocolate croissants
- "pronoun" test
  - A small dog is barking in the park.
  - It is barking in the park
- "question by repetition" test:
  - I have seen blue elephants
  - Blue elephants?
  - * Seen blue?
  - Seen blue elephants?
- "topicalization" test:
  - Blue elephants, I have seen.
- "question" test:
  - *What* have I seen?
- "deletion" test
  - Last year I saw <u>a blue elephant in the zoo</u>.
- "semantic" test
- "intuitition" test

How to generate sentences

- One way: tree structure
  - Generate the tree structure first
  - Then fill the leaf nodes with terminals

A Simple Syntactic Rule

- The simplest rule for a sentence, e.g. “Birds fly”

  $$\mathtt{S} \rightarrow \mathtt{N}\ \mathtt{V}$$

Simplest Grammar

S → N V

N → Samantha | Min | Jorge

V → left | sang | walked

Sample sentences:

Samantha sang

Jorge left

Syntax

- The verbs so far were intransitive (no direct object)
- What rules are needed next?
  - Transitive verbs and direct objects (“Jorge saw Samantha”)
  - Determiners (“the cats”)
- Combinatorial explosion (even for the simplest form of sentences)
  - Need for noun phrases
  - Ditto for verb phrases


DUDE, WHY DO YOU TALK BACKWARDS ALL THE TIME?
BACKWARDS-TALKING I AM NOT. ENGLISH-CENTRIC VIEW OF GRAMMATICAL STRUCTURE, YOU HAVE
©2012 BY DOUG SAVAGE

Latest Grammar

S → NP VP

NP → DT N

VP → V NP

DT → the | a

N → child | cat | dog

V → took | saw | liked | scared | chased

Sample sentences:

a dog chased the cat

the child saw a dog

Alternatives

- Different expansions of a category are delineated with ″|″
  - `NP → PN | DT CN`
- One rule for proper nouns and another for common nouns

Latest Grammar

S → NP VP

NP → DT CN

NP → PN

VP → V NP

DT → the | a

CN → child | cat | dog

PN → Samantha | Jorge | Min

V → took | saw | liked | scared | chased

Sample sentences:

a child scared Jorge

Min took the child

Optional categories

- Wherever `N` is allowed in a sentence,
  - `DT N`
  - `JJ N`
  - `DT JJ N`

  are also allowed
- We can use the notation for alternatives
  - `NP → N | DT N | JJ N | DT JJ N`
- Optional categories can be also marked using parentheses:
  - `NP → (DT) (JJ) N`

Verb Phrases

- Samantha ran.
- Samantha ran to the park.
- Samantha ran away.
- Samantha bought a cookie.
- Samantha bought a cookie for John.
- Overall structure
  - `VP` $\rightarrow$ `V(NP)(P)(NP)`

Latest Grammar

S → NP VP

NP → DT CN

NP → PN

VP → V (NP) (P) (NP)

DT → the | a

CN → child | cat | dog

PN → Samantha | Jorge | Min

P → to | for | from | in

V → took | saw | liked | scared | chased | gave

Sample sentences:

Samantha saw the cat

Jorge gave the cat to Min

Prepositional Phrases

- Examples:
  - Mary bought a book for John **in a bookstore**.
  - The bookstore sells magazines.
  - The bookstore **on Main St**. sells magazines.
  - Mary ran away.
  - Mary ran **down the hill**.
- Changes are needed to both NP and VP to accommodate prepositional phrases
  - Wherever a preposition is allowed, it can be followed by a noun phrase.
  - Run up
  - NP can contain any number of PPs but only up to two NPs.
- How do we revise the grammar accordingly?

The Rules So Far

- **S → NP VP**
- **NP → (DT) (JJ) N (PP)**
- **VP → V (NP) (PP)**
- **PP → P (NP)**

PP Ambiguity

- The boy saw the woman with the telescope.

```
PP → PREP NP
VP → V NP PP
VP → V NP
NP → DT N
NP → DT N PP
```

Repetition (*)

- (JJ*) = a sequence of zero or more JJ
- Are all sequences of adjectives allowed?
  - a big red house
  - * a red big house
- Adjective ordering in English depends on semantics!

Exercise

- The Little Red Riding Hood
- Three Little Pigs
- The Three Musketeers
- The Steadfast Tin Soldier
- The French Connection
- Old Macdonald
- Five Golden Rings
- The Ancient Mariner

Adjective ordering

- **Det**
- Number
- Strength
- Size
- Age
- Shape
- Color
- Origin
- Material
- Purpose
- **Noun**

- det < number < size < color < purpose < noun
- strength < material < noun
- origin < noun

Nested Sentences

- Examples:
  - I don’t recall whether I took the dog out.
  - Do you know if the mall is still open?
- `VP → V (NP) (NP) (C S) (PP*)`
- Can (C S) appear inside an NP?
  - Whether he will win the elections remains to be seen.

Recursion

- S can generate VP, VP can generate S
- NP can generate PP, PP can generate NP
- What does recursion allow?
- Is there a longest sentence in English?
- Conjunction of NPs:

  `NP → NP and NP`

- Conjunction of PPs:

  `PP → PP and PP`

- Conjunction of VPs:

  `VP → VP and VP`

Meta-patterns

- S $\rightarrow$ NP VP
  - NP $\rightarrow$ (DT) (JJ) N (PP)
  - VP $\rightarrow$ V (NP) (PP)
  - PP $\rightarrow$ P (NP)
- Is there a meta-pattern here?
  - XP $\rightarrow$ (specifier) X’
  - X’ $\rightarrow$ X (complement)
- Example: NP $\rightarrow$ DT N’
- X-bar Theory
  - http://www.unlweb.net/wiki/X-bar_theory

Meta-rules for Conjunctions

- Conjunction
  - `X → X and X`
- This kind of rule even covers entire sentences
  - `S → S and S`

Auxiliaries

- Is “Aux V” a constituent?
  - I <u>have seen</u> blue elephants and <u>will remember</u> them forever.
- Recursion:
  - VP -> Aux VP
  - Raj may have been sleeping.
- Is such recursion unlimited?

Exercise

- Grammar:
  - `S → NP VP | CP VP`
  - `NP → (DT) (JJ*) N (CP) (PP*)`
  - `VP → V (NP) (NP) (PP*) | V (NP) (CP) (PP*)`
  - `PP → P NP`
  - `CP → C S`
- What rules are needed to generate these three sentences:
  - 1. The small dog of the neighbors brought me an old tennis ball.
  - 2. That wugs have three eyes is unproven by scientists.
  - 3. I saw the gift that the old man gave me at the meeting.

Notes

- Syntax helps with sentences like
    * The milk drank the cat
    The milk is drunk by the cat
- Overgeneration
    The girl saw
- Undergeneration
- Grammar – between the two

Arguments vs. Adjuncts

- Arguments
  - Mandatory (e.g., “* Romeo likes”, “*likes Juliet”)
  - Cannot be repeated (e.g., “* Juliet likes Romeo John”)
  - Verbs can have more than one subcategorization frame
- Adjuncts
  - Optional
  - Typically prepositional phrases or adverbs
  - Can be repeated (e.g., “Apparently Candace ate pizza yesterday at the restaurant with pleasure”)

NLP

Introduction to NLP

242.

Introduction to Parsing

Parsing programming languages

```
#include <stdio.h>

int main()
{
  int n, reverse = 0;

  printf("Enter a number to reverse\n");
  scanf("%d",&n);

  while (n != 0)
  {
    reverse = reverse * 10;
    reverse = reverse + n%10;
    n = n/10;
  }
  printf("Reverse of entered number is = %d\n", reverse);

  return 0;
}
```

Parsing human languages

- Rather different than computer languages
  - Can you think in which ways?

Parsing human languages

- Rather different than computer languages
  - No types for words
  - No brackets around phrases
  - Ambiguity
    - Words
    - Parses
  - Implied information

The parsing problem

- Parsing means associating tree structures to a sentence, given a grammar (e.g., CFG)
  - There may be exactly one such tree structure
  - There may be many such structures
  - There may be none
- Grammars (e.g., CFG) are declarative
  - They don't specify how the parse tree will be constructed

Syntactic Issues

- PP attachment
  - I saw the man with the telescope
- Gaps
  - Mary likes Physics but hates Chemistry
- Coordination scope
  - Small boys and girls are playing
- Particles vs. prepositions
  - She ran up a large bill
- Gerund vs. adjective
  - Frightening kids can cause trouble

Applications of parsing

- Grammar checking
  - I want to return this shoes.
- Question answering
  - How many people in sales make $40K or more per year?
- Machine translation
  - E.g., word order – SVO vs. SOV
- Information extraction
  - *Breaking Bad* takes place in New Mexico.
- Speech generation
- Speech understanding

NLP

Introduction to NLP

Context-free grammars

Context-free grammars

- A context-free grammar is a 4-tuple (N,Σ,R,S)
  - N: non-terminal symbols
  - Σ: terminal symbols (disjoint from N)
  - R: rules ($A \rightarrow \beta$), where β is a string from $(\Sigma \cup N)^*$
  - S: start symbol from N

Example

```
["the", "child", "ate", "the", "cake", "with", "the", "fork"]
```

```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

Example

```
["the", "child", "ate", "the", "cake", "with", "the", "fork"]
```

S -> NP VP
NP -> DT **N** | NP PP
PP -> **PRP** NP
VP -> **V** NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'

Heads marked in bold face

Phrase-structure grammars (1/2)

- Sentences are not just bags of words
  - Alice bought Bob flowers
  - Bob bought Alice flowers
- Context-free view of language
  - A prepositional phrase looks the same whether it is part of the subject NP or part of the VP
- Constituent order
  - SVO (subject verb object)
  - SOV (subject object verb)

Phrase-structure grammars (2/2)

- Auxiliary verbs
  - The dog may have eaten my homework
- Imperative sentences
  - Leave the book on the table
- Interrogative sentences
  - Did the customer have a complaint?
  - Who had a complaint?
- Negative sentences
  - The customer didn’t have a complaint

A longer example

```
S -> NP VP | Aux NP VP | VP
NP -> PRON | Det Nom
Nom -> N | Nom N | Nom PP
PP -> PRP NP
VP -> V | V NP | VP PP
Det -> 'the' | 'a' | 'this'
PRON -> 'he' | 'she'
N -> 'book' | 'boys' | 'girl'
PRP -> 'with' | 'in'
V -> 'takes' | 'take'
```

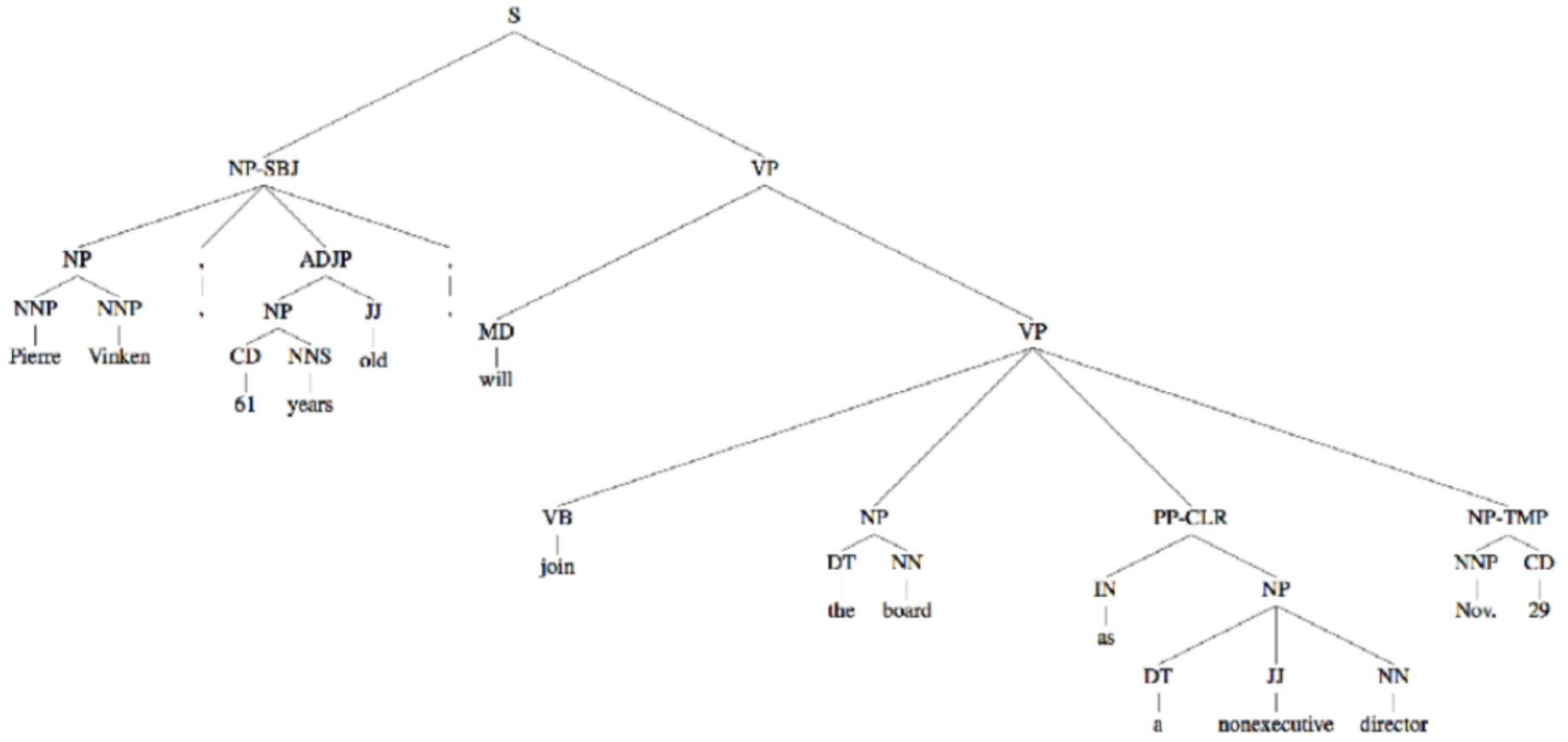
What changes were made to the grammar?

A longer example

```
S -> NP VP | Aux NP VP | VP
NP -> PRON | Det Nom
Nom -> N | Nom N | Nom PP
PP -> PRP NP
VP -> V | V NP | VP PP
Det -> 'the' | 'a' | 'this'
PRON -> 'he' | 'she'
N -> 'book' | 'boys' | 'girl'
PRP -> 'with' | 'in'
V -> 'takes' | 'take'
```

A longer example

```
S -> NP VP | Aux NP VP | VP
NP -> PRON | Det Nom
Nom -> N | Nom N | Nom PP
PP -> PRP NP
VP -> V | V NP | VP PP
Det -> 'the' | 'a' | 'this'
PRON -> 'he' | 'she'
N -> 'book' | 'boys' | 'girl'
PRP -> 'with' | 'in'
V -> 'takes' | 'take'
```


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

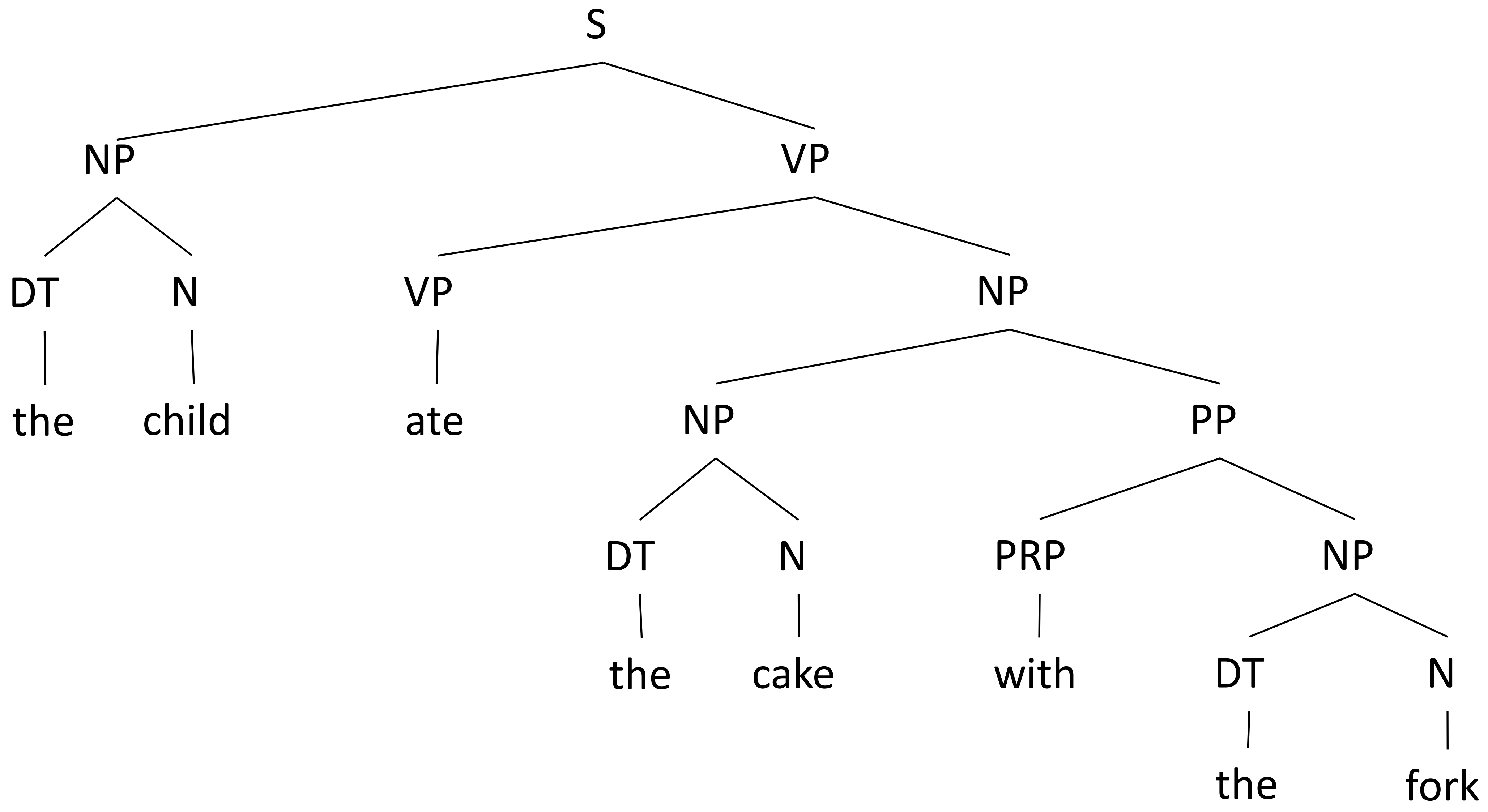
Penn Treebank Example

```
( (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) )))
    (. .) ))
( (S
    (NP-SBJ (NNP Mr.) (NNP Vinken) )
    (VP (VBZ is)
      (NP-PRD
        (NP (NN chairman) )
        (PP (IN of)
          (NP
            (NP (NNP Elsevier) (NNP N.V.) )
            (, ,)
            (NP (DT the) (NNP Dutch) (VBG publishing) (NN group) )))))
    (. .) ))
```

CFGs are equivalent to PDAs

- PDA = Pushdown Automata
- Example: consider the language $L=\{x^n y^n\}$
  - stack is empty, input=xxxyyy
  - push * onto stack, input=xxyyy
  - push * onto stack, input=xyyy
  - push * onto stack, input=yyy
  - pop * from stack, input=yy
  - pop * from stack, input=y
  - pop * from stack, input=“”

Leftmost derivation

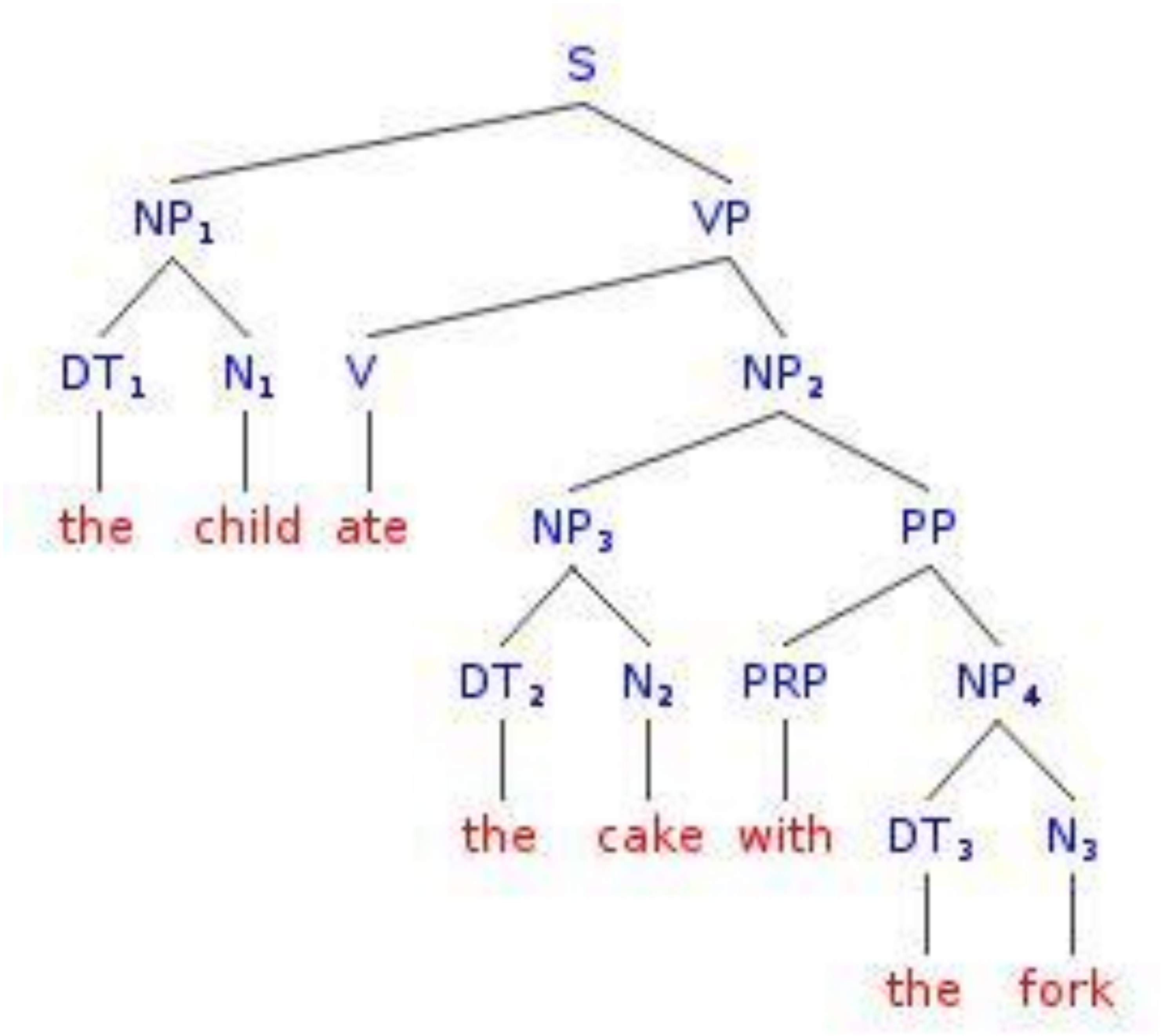


NLP

Introduction to NLP

243.

Classic parsing methods




```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

Parsing as search

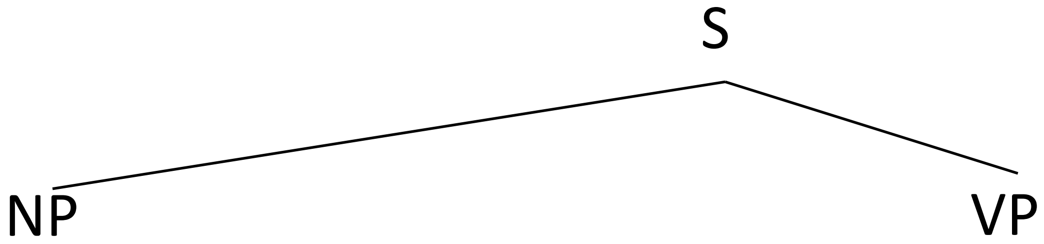
- There are two types of constraints on the parses
  - From the input sentence
  - From the grammar
- Therefore, two general approaches to parsing
  - Top-down
  - Bottom-up

Top down parsing

S

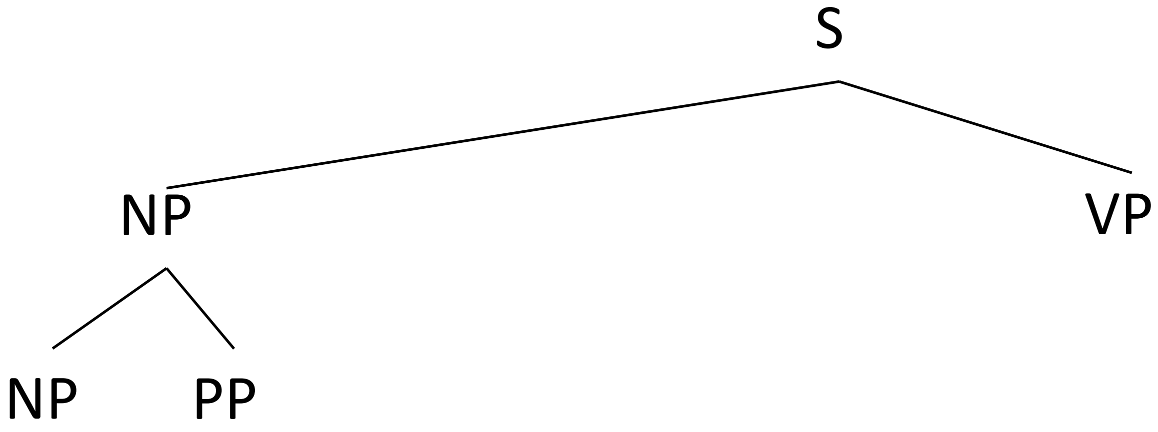
```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

Top down parsing



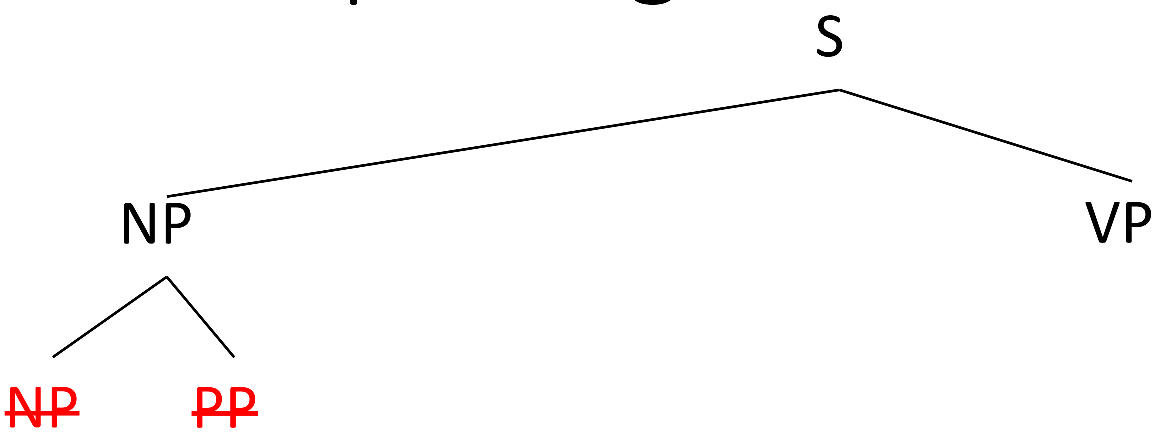

```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

Top down parsing



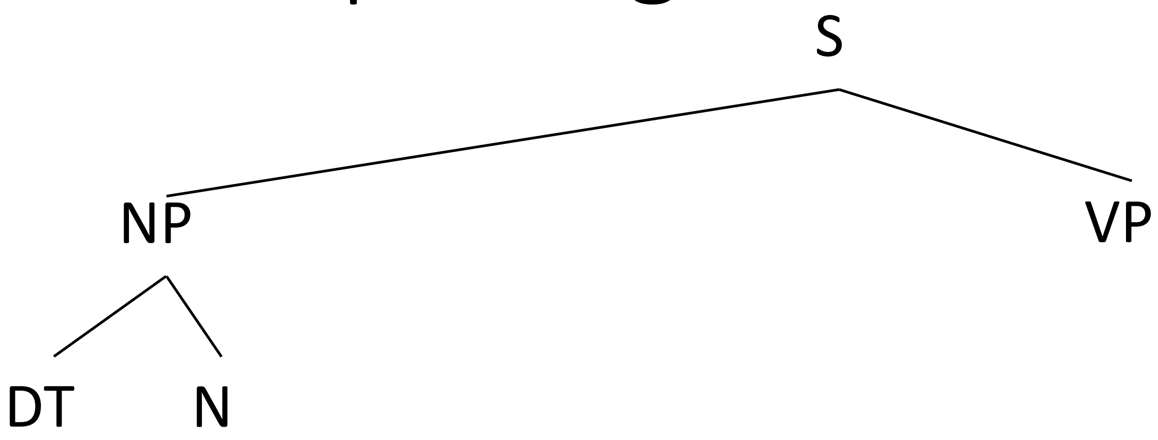

```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

Top down parsing



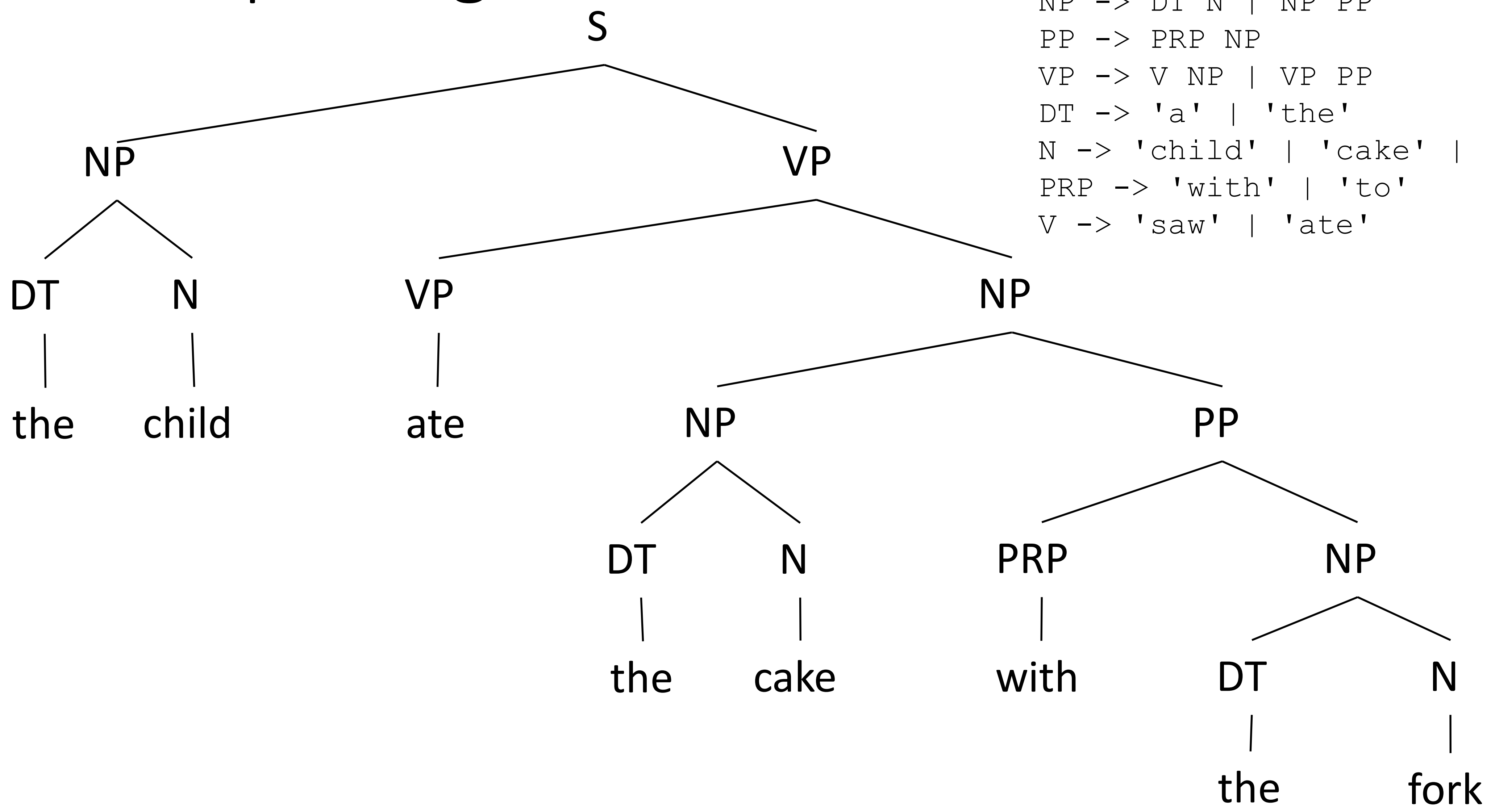

```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

Top down parsing




```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

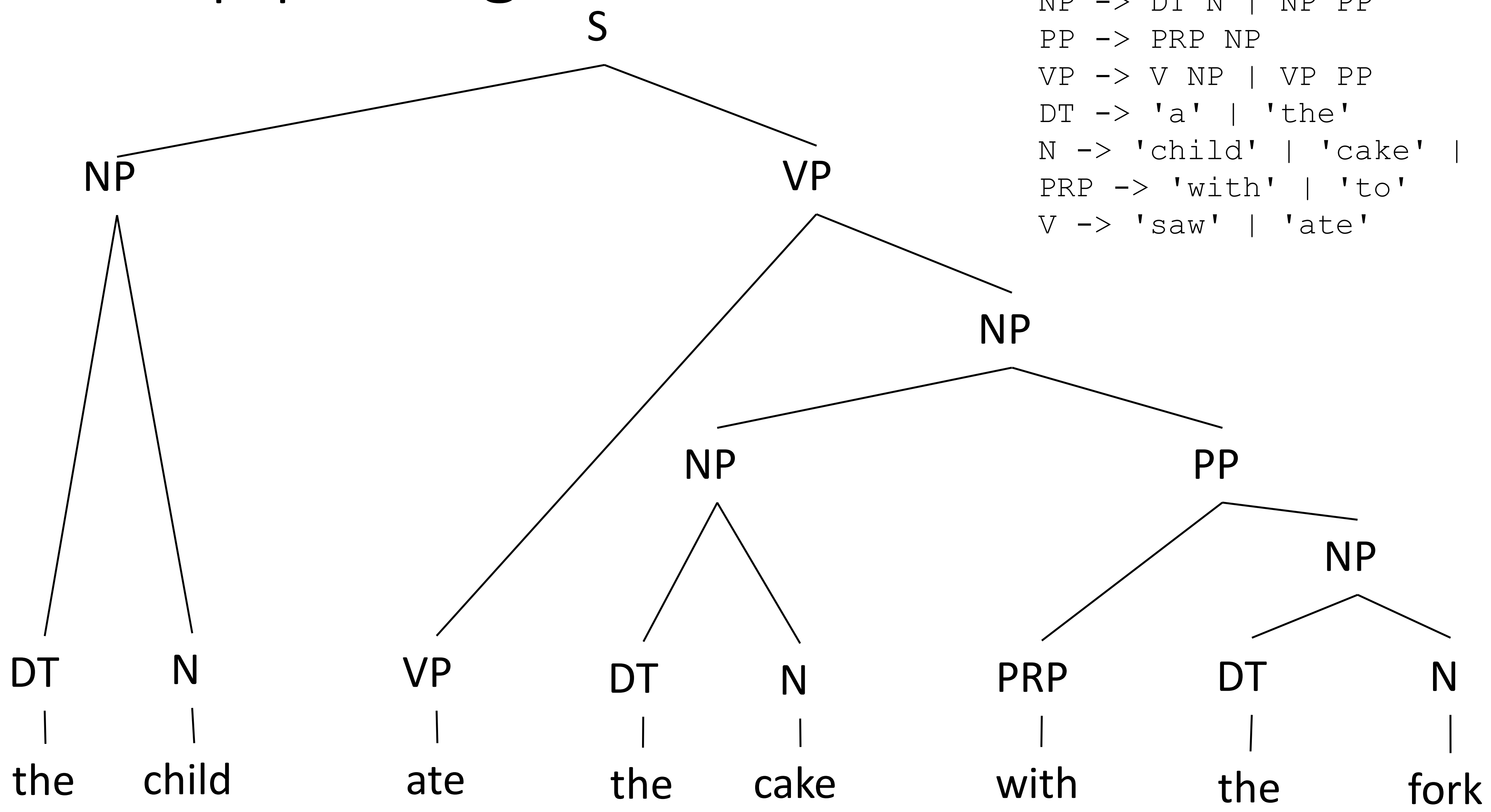
Top down parsing




```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

Bottom up parsing

```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```



Bottom up vs. top down methods

- Bottom up
  - explores options that won’t lead to a full parse
  - Example: shift-reduce (srparser in nltk)
  - Example: CKY (Cocke-Kasami-Younger)
- Top down
  - explores options that don’t match the full sentence
  - Example: recursive descent (rdparser in nltk)
  - Example: Earley parser
- Dynamic programming
  - caches of intermediate results (memoization)

Introduction to NLP

Shift-Reduce Parsing

Shift-Reduce Parsing

- A bottom-up parser
  - Tries to match the RHS of a production until it can build an S
- *Shift* operation
  - Each word in the input sentence is pushed onto a stack
- *Reduce-n* operation
  - If the top *n* words on the top of the stack match the RHS of a production, then they are popped and replaced by the LHS of the production
- Breadth-first search
- Stopping condition
  - The process stops when the input sentence has been processed and S has been popped from the stack

Shift-Reduce Parsing Example

```
  [ * the child ate the cake]
S [ 'the' * child ate the cake]
R [ DT * child ate the cake]
S [ DT 'child' * ate the cake]
R [ DT N * ate the cake]
R [ NP * ate the cake]
S [ NP 'ate' * the cake]
R [ NP V * the cake]
S [ NP V 'the' * cake]
R [ NP V DT * cake]
S [ NP V DT 'cake' * ]
R [ NP V DT N * ]
R [ NP V NP * ]
R [ NP VP * ]
R [ S * ]
```

(S (NP (DT the) (N child)) (VP (V ate) (NP (DT the) (N cake))))

Shift-Reduce Parsing

- In nltk

```
>>> from nltk.app import srparser;
>>> srparser())
```

NLP

Introduction to NLP

244.

Cocke-Kasami-Younger (CKY) Parsing

Notes on Left Recursion

- Problematic for many parsing methods
  - Infinite loops when expanding
- But appropriate linguistically
  - NP -> DT N
  - NP -> PN
  - DT -> NP ‘s
  - Mary’s mother’s sister’s friend

Chart Parsing

- Top-down parsers have problems with expanding the same non-terminal
  - In particular, pre-terminals such as POS
  - Bad idea to use top-down (recursive descent) parsing as is
- Bottom-up parsers have problems with generating locally feasible subtrees that are not viable globally
- Chart parsing will address these issues

Dynamic Programming

- Motivation
  - A lot of the work is repeated
  - Caching intermediate results improves the complexity
- Dynamic programming
  - Building a parse for a substring [i,j] based on all parses [i,k] and [k, j] that are included in it.
- Complexity
  - O(n^3) for recognizing an input string of length *n*

Dynamic Programming

- CKY (Cocke-Kasami-Younger)
  - bottom-up
  - requires a normalized (binarized) grammar
- Earley parser
  - top-down
  - more complicated
  - (separate lecture)

CKY Algorithm

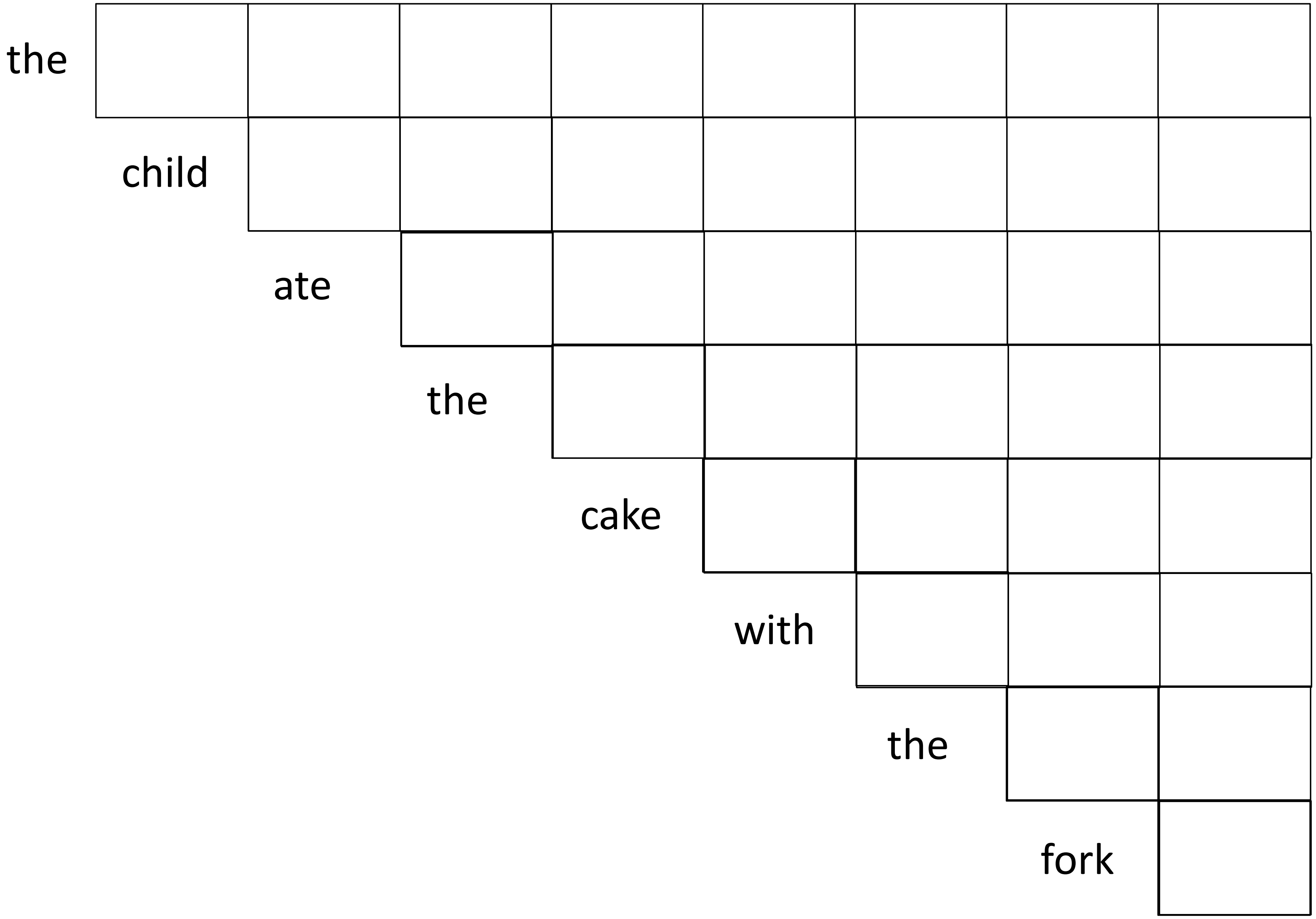
```
function cky (sentence W, grammar G) returns table
  for i in 1..length(W) do
    table[i-1,i] = {A|A->Wi in G}
  for j in 2..length(W) do
    for i in j-2 down to 0 do
      for k in (i+1) to (j-1) do
        table[i,j] = table[i,j] union {A|A->BC in G, B in
table [I,k], C in table [k,j]}

If the start symbol S is in table [0,n] then W is in L(G)
```

Example

```
["the", "child", "ate", "the", "cake", "with", "the", "fork"]
```

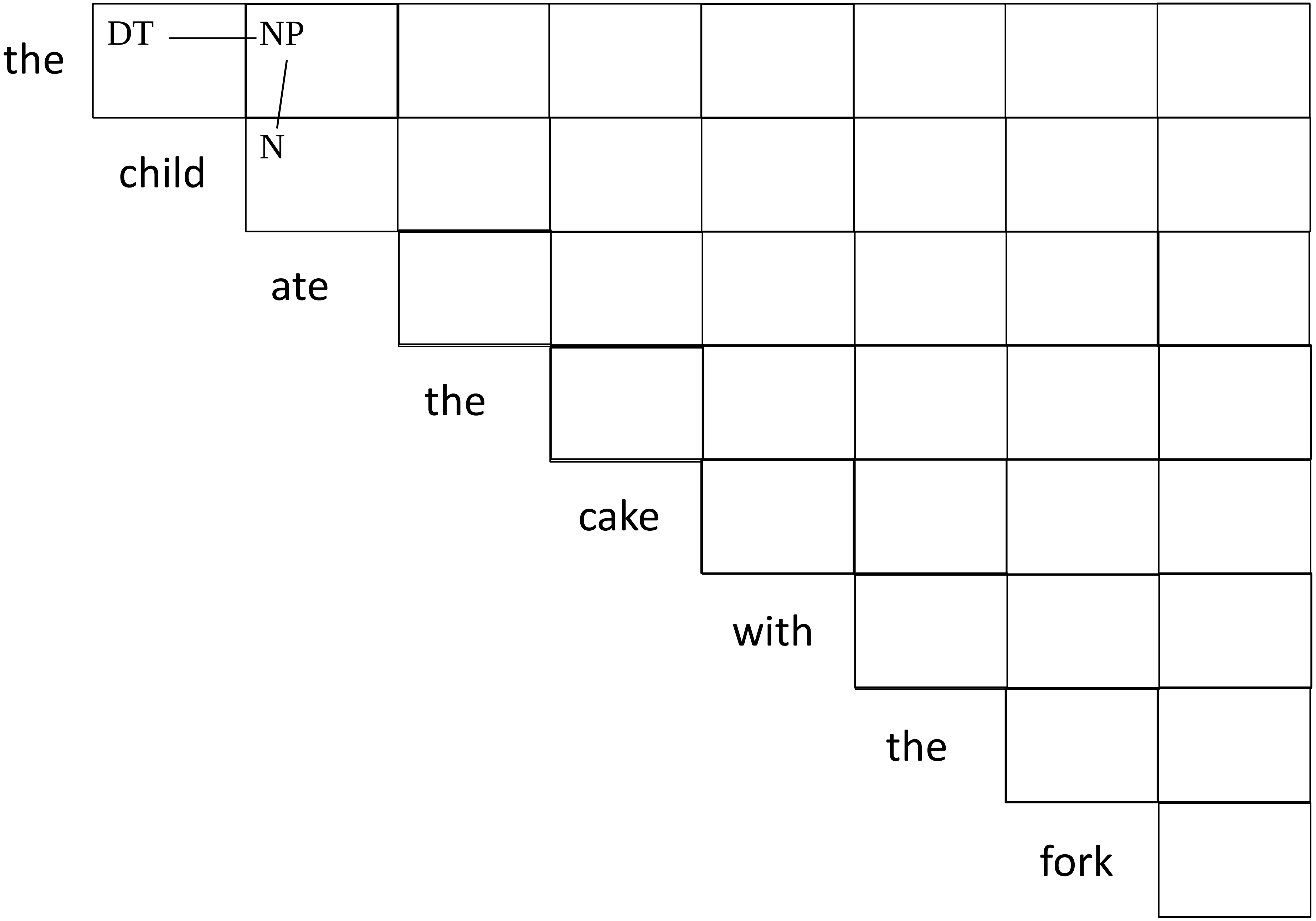
```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

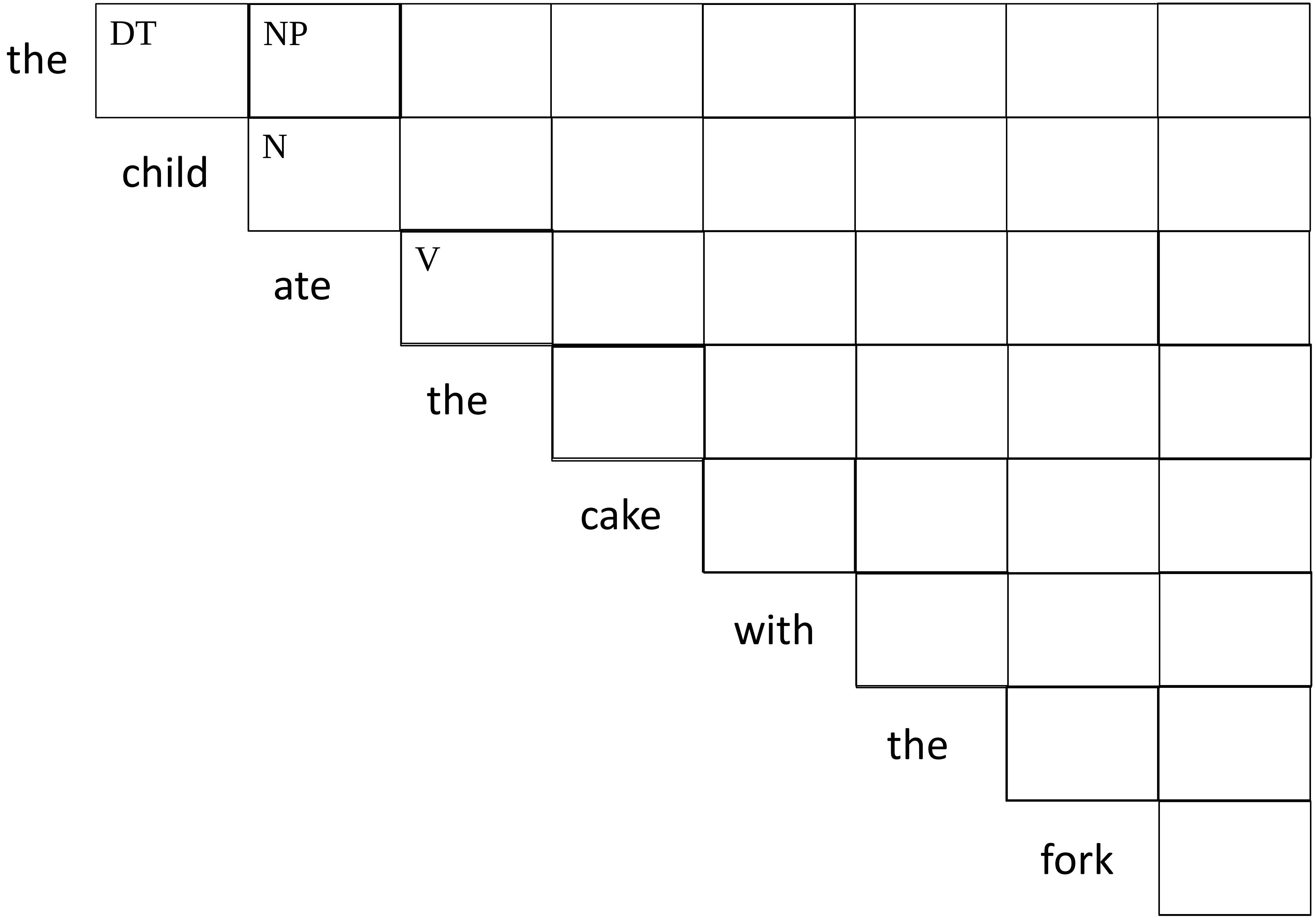

the
child
ate
the
cake
with
the
fork

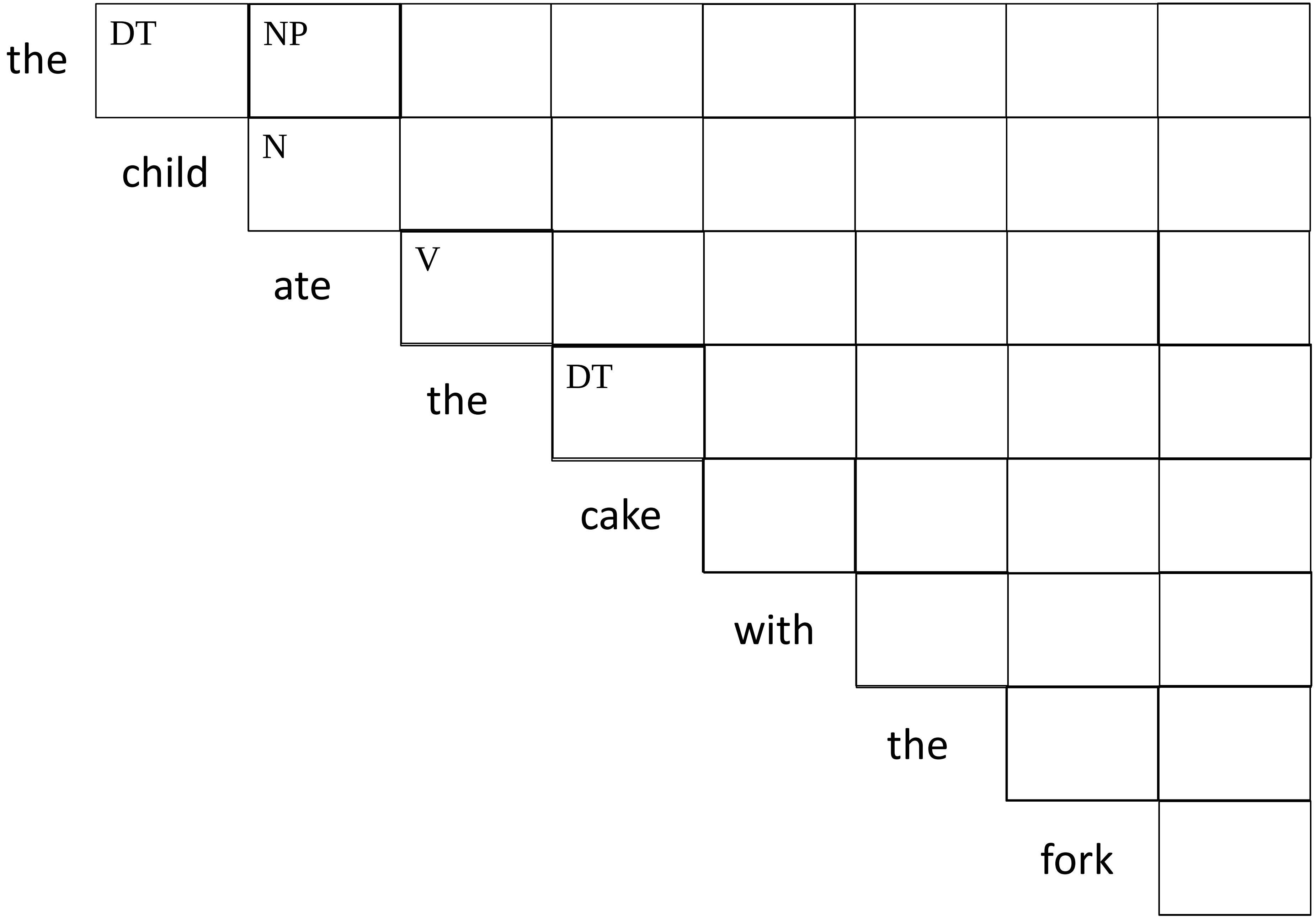
the	DT							
child								
ate								
the								
cake								
with								
the								
fork								

the	DT							
child		N						
ate								
the								
cake								
with								
the								
fork								

the	DT	NP						
	child	N						
		ate						
			the					
				cake				
					with			
						the		
							fork	


the
child
ate
the
cake
with
the
fork
DT
NP
N

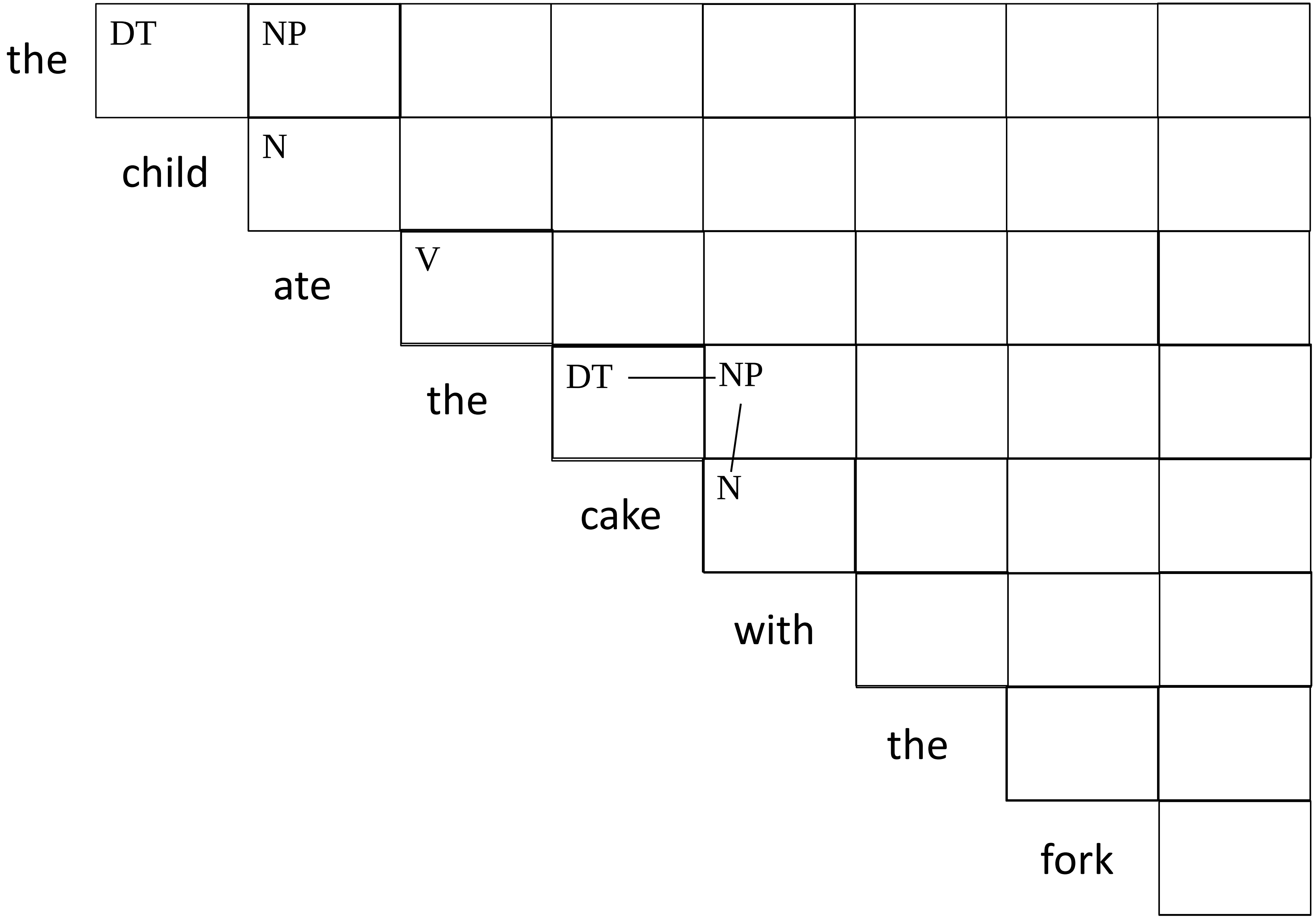

the
DT
NP
child
N
ate
V
the
cake
with
the
fork



the
DT
NP
child
N
ate
V
the
DT
cake
with
the
fork

the	DT	NP						
child		N						
ate			V					
the				DT				
cake					N			
with								
the								
fork								

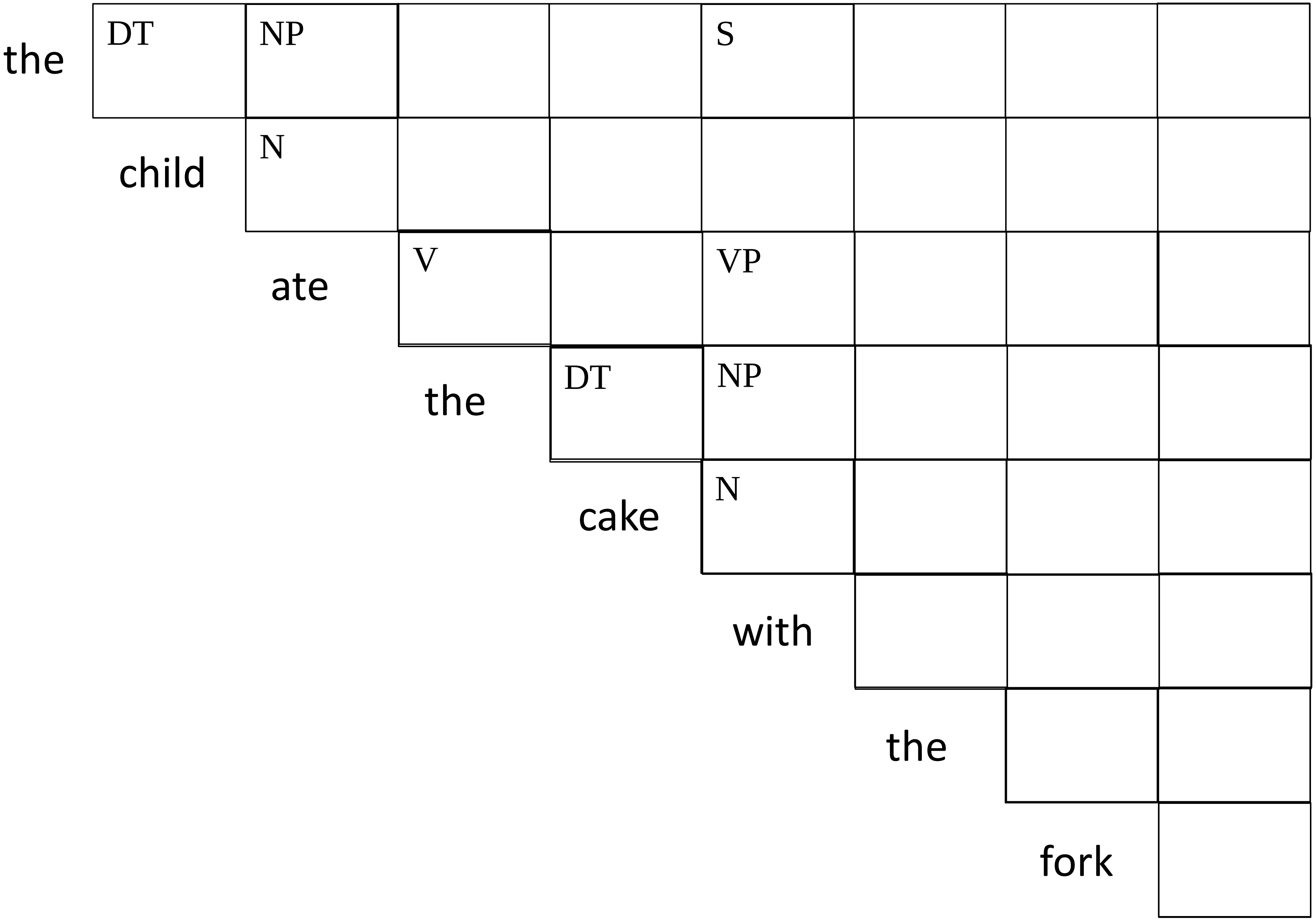
the	DT	NP						
	child	N						
		ate	V					
			the	DT	NP			
				cake	N			
					with			
						the		
							fork	



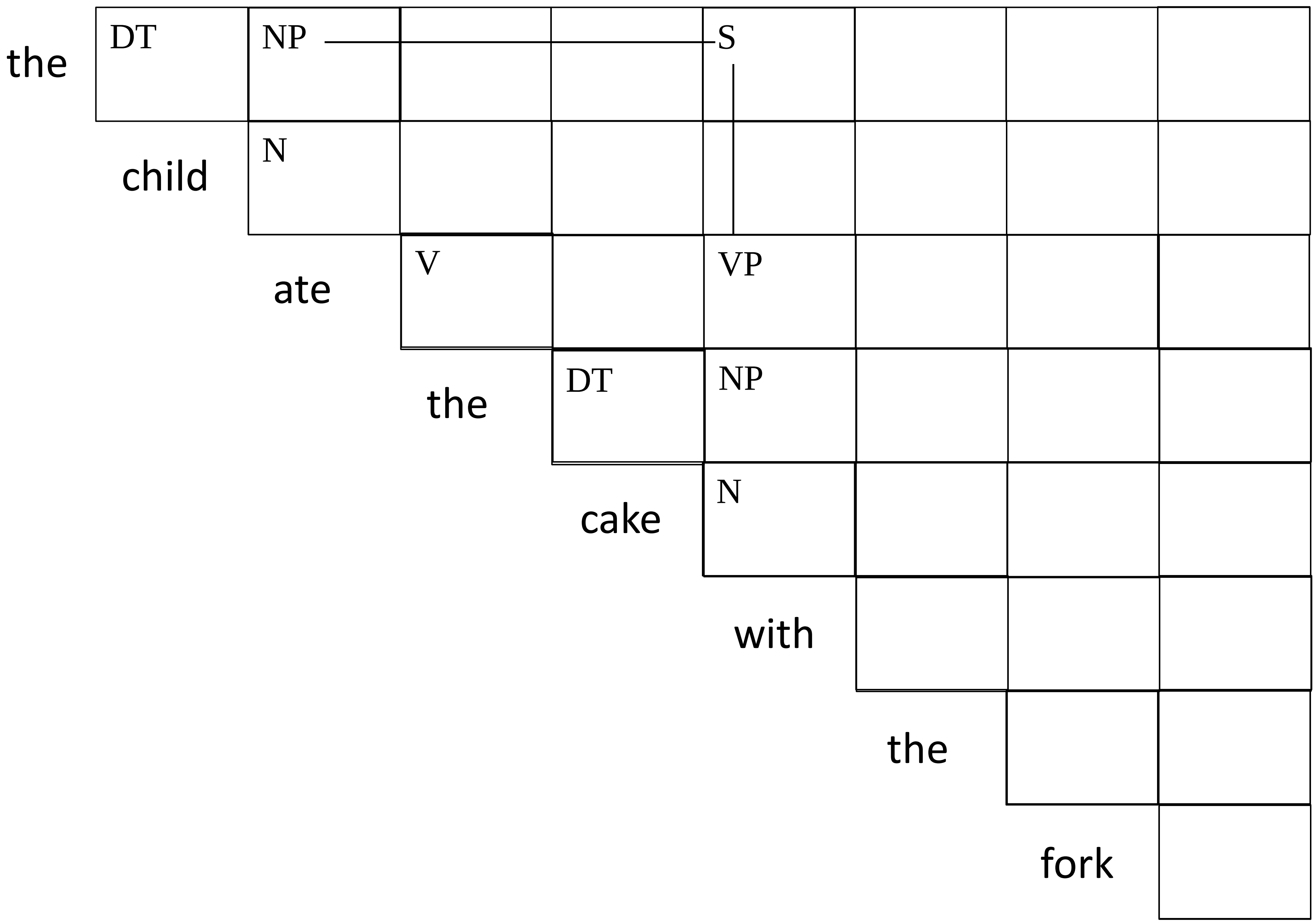
the
DT
NP
child
N
ate
V
the
DT
NP
cake
N
with
the
fork

the	DT	NP						
	child	N						
		ate	V		VP			
			the	DT	NP			
				cake	N			
					with			
						the		
							fork	

the	DT	NP						
	child	N						
		ate	V		VP			
			the	DT	NP			
				cake	N			
					with			
						the		
							fork	



the
DT
NP
S
child
N
ate
V
VP
the
DT
NP
cake
N
with
the
fork


the
child
ate
the
cake
with
the
fork
DT
NP
S
N
V
VP
DT
NP
N

the	DT	NP			S			
child		N						
ate			V		VP			
the				DT	NP			
cake					N			
with						PRP		
the								
fork								

the	DT	NP			S			
child		N						
ate			V		VP			
the				DT	NP			NP
cake					N			
with						PRP		PP
the							DT	NP
fork								N

the	DT	NP			S			
child		N						
ate			V		VP			VP
the				DT	NP			NP
cake					N			
with						PRP		PP
the							DT	NP
fork								N

the	DT	NP			S			
child		N						
ate			V		VP			VP
the				DT	NP			NP
cake					N			
with						PRP		PP
the							DT	NP
fork								N

the	DT	NP			S			S
child		N						
ate			V		VP			VP
the				DT	NP			NP
cake					N			
with						PRP		PP
the							DT	NP
fork								N

the	DT	NP			S			S
child		N						
ate			V		VP			VP
the				DT	NP			NP
cake					N			
with						PRP		PP
the							DT	NP
fork								N

[0] DT [1] N [2] ==> [0] NP [2]

[3] DT [4] N [5] ==> [3] NP [5]

[6] DT [7] N [8] ==> [6] NP [8]

[2] V [3] NP [5] ==> [2] VP [5]

[5] PRP [6] NP [8] ==> [5] PP [8]

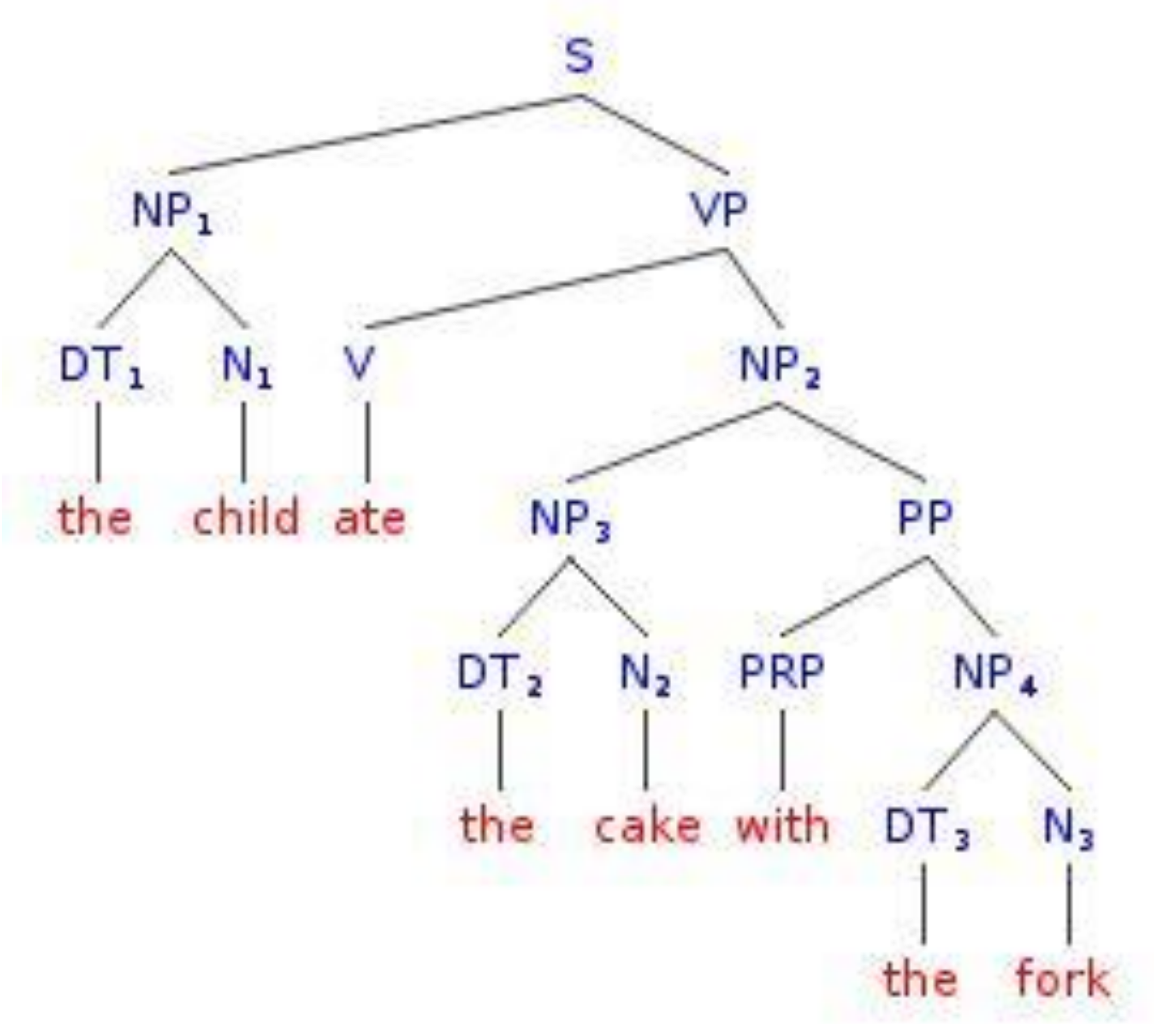
[0] NP [2] VP [5] ==> [0] S [5]

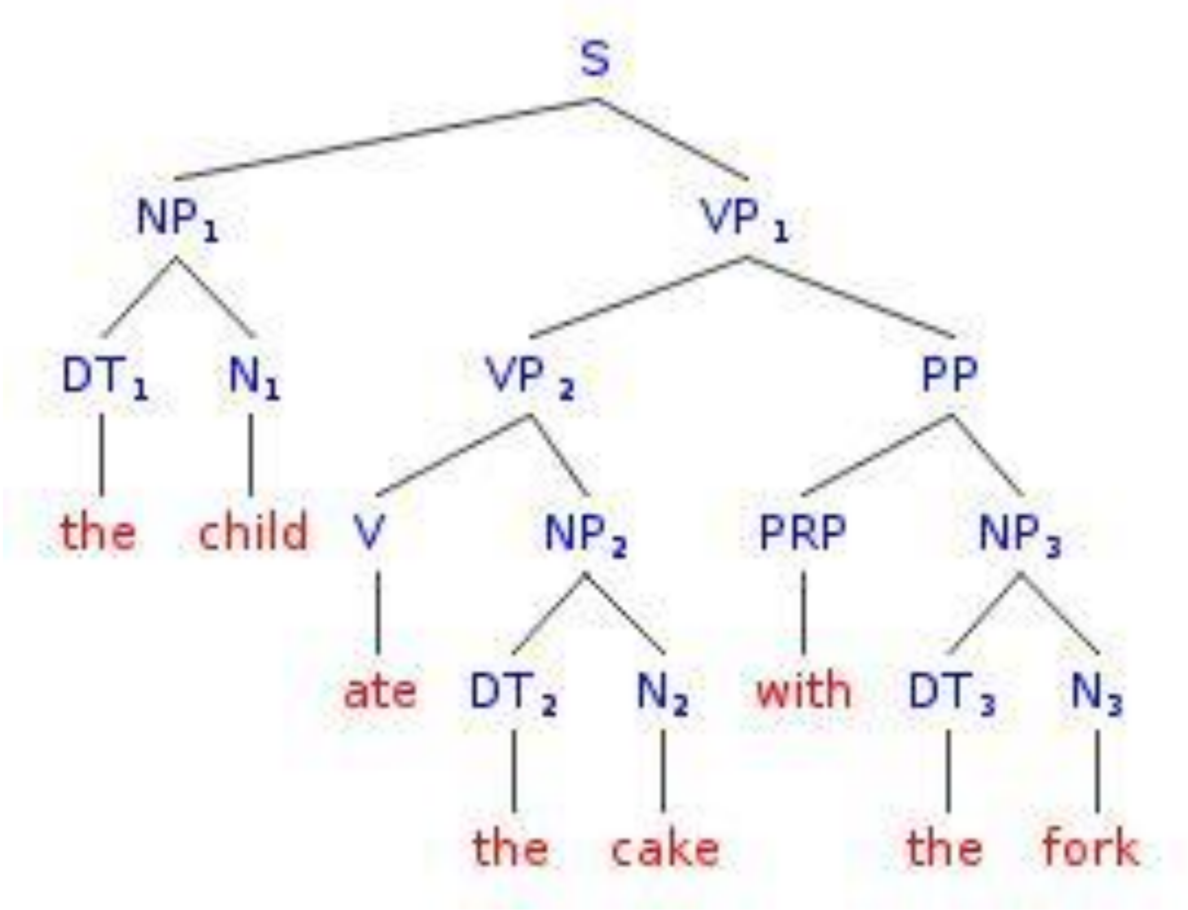
[3] NP [5] PP [8] ==> [3] NP [8]

[2] V [3] NP [8] ==> [2] VP [8]

[2] VP [5] PP [8] ==> [2] VP [8]

[0] NP [2] VP [8] ==> [0] S [8]





What is the *meaning* of each of these sentences?

```
(S
  (NP (DT the) (N child))
  (VP
    (VP (V ate) (NP (DT the) (N cake)))
    (PP (PRP with) (NP (DT the) (N fork)))))
```

```
(S
  (NP (DT the) (N child))
  (VP
    (VP (V ate) (NP (DT the) (N cake)))
    (PP (PRP with) (NP (DT the) (N fork)))))
```

```
(S
  (NP (DT the) (N child))
  (VP
    (V ate)
    (NP
      (NP (DT the) (N cake))
      (PP (PRP with) (NP (DT the) (N fork))))))
```

Complexity of CKY

- Space complexity
  - There are O(n^2) cells in the table
- Single parse
  - Each cell requires a linear lookup.
  - Total time complexity is O(n^3)
- All parses
  - Total time complexity is exponential

A longer example

```
["take", "this", "book"]

    S -> NP VP | Aux NP VP | VP
    NP -> PRON | Det Nom
    Nom -> N | Nom N | Nom PP
    PP -> PRP NP
    VP -> V | V NP | VP PP
    Det -> 'the' | 'a' | 'this'
    PRON -> 'he' | 'she'
    N -> 'book' | 'boys' | 'girl'
    PRP -> 'with' | 'in'
    V -> 'takes' | 'take'
```

Non-binary productions

```
["take", "this", "book"]

    S -> NP VP | Aux NP VP | VP
    NP -> PRON | Det Nom
    Nom -> N | Nom N | Nom PP
    PP -> PRP NP
    VP -> V | V NP | VP PP
    Det -> 'the' | 'a' | 'this'
    PRON -> 'he' | 'she'
    N -> 'book' | 'boys' | 'girl'
    PRP -> 'with' | 'in'
    V -> 'takes' | 'take'
```

Chomsky Normal Form (CNF)

- All rules have to be in binary form:
  - $X \rightarrow Y\ Z$ or $X \rightarrow w$
- This introduces new non-terminals for
  - hybrid rules
  - n-ary rules
  - unary rules
  - epsilon rules (e.g., $NP \rightarrow \varepsilon$)
- Any CFG can be converted to CNF
  - See Aho & Ullman p. 152

ATIS grammar

Original version

S → NP VP
S → Aux NP VP

S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

From Jurafsky and Martin

ATIS grammar in CNF

Original version

S → NP VP
S → Aux NP VP

S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

CNF version

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
PP → Prep NP

ATIS grammar in CNF

Original version

S → NP VP
S → Aux NP VP

S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

CNF version

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
PP → Prep NP

Chomsky Normal Form

- All rules have to be in binary form:
  - $X \rightarrow Y\ Z$ or $X \rightarrow w$
- New non-terminals for hybrid rules, n-ary and unary rules:
  - INF-VP $\rightarrow$ to VP *becomes*
    - INF-VP $\rightarrow$ TO VP
    - TO $\rightarrow$ to
  - S $\rightarrow$ Aux NP VP *becomes*
    - S $\rightarrow$ R1 VP
    - R1 $\rightarrow$ Aux NP
  - S $\rightarrow$ VP VP $\rightarrow$ Verb VP $\rightarrow$ Verb NP VP $\rightarrow$ Verb PP *becomes*
    - S $\rightarrow$ book
    - S $\rightarrow$ buy
    - S $\rightarrow$ R2 PP
    - S $\rightarrow$ Verb PP
  - etc.

Issues with CKY

- Weak equivalence only
  - Same language, different structure
  - If the grammar had to be converted to CNF, then the final parse tree doesn't match the original grammar
  - However, it can be converted back using a specific procedure
- Syntactic ambiguity
  - (Deterministic) CKY has no way to perform syntactic disambiguation

Notes

- Demo:
  - http://lxmls.it.pt/2015/cky.html
- Recognizing vs. parsing
  - Recognizing just means determining if the string is part of the language defined by the CFG
  - Parsing is more complicated – it involves producing a parse tree

NLP

Introduction to NLP

Issues with Context-free Grammars

Agreement

- Number
  - Chen is/people are
- Person
  - I am/Chen is
- Tense
  - Chen was reading/Chen is reading/Chen will be reading
- Case
  - not in English but in many other languages such as German, Russian, Greek
- Gender
  - not in English but in many other languages such as German, French, Spanish

Combinatorial explosion

- Many combinations of rules are needed to express agreement
  - S $\rightarrow$ NP VP
  - S $\rightarrow$ 1sgNP 1sgVP
  - S $\rightarrow$ 2sgNP 2sgVP
  - S $\rightarrow$ 3sgNP 3sgVP
  - …
  - 1sgNP $\rightarrow$ 1sgN
  - ...

Subcategorization frames

- Direct object
  - The dog ate a sausage
- Prepositional phrase
  - Mary left the car in the garage
- Predicative adjective
  - The receptionist looked worried
- Bare infinitive
  - She helped me buy this place
- To-infinitive
  - The girl wanted to be alone
- Participial phrase
  - He stayed crying after the movie ended
- That-clause
  - Ravi doesn’t believe that it will rain tomorrow
- Question-form clauses
  - She wondered where to go
- Empty (ϕ)
  - She slept

CFG independence assumptions

- Non-independence
  - All NPs
    - 11% NP PP, 9% DT NN, 6% PRP
  - NPs under S
    - 9% NP PP, 9% DT NN, 21% PRP
  - NPs under VP
    - 23% NP PP, 7% DT NN, 4% PRP
  - example from Dan Klein

NLP

Introduction to NLP

247.

Parsing Evaluation

Parsing Evaluation

- Parseval: precision and recall
  - get the proper constituents
- Labeled precision and recall
  - also get the correct non-terminal labels
- F1
  - harmonic mean of precision and recall
- Crossing brackets
  - (A (B C)) vs ((A B) C)
- PTB corpus
  - training 02-21, development 22, test 23

Evaluation Example

```
GOLD = (S (NP (DT The) (JJ Japanese) (JJ industrial) (NNS companies))
          (VP (MD should) (VP (VB know) (ADVP (JJR better)))) (. .)

CHAR = (S (NP (DT The) (JJ Japanese) (JJ industrial) (NNS companies))
           (VP (MD should) (VP (VB know)) ((ADVP (RBR better)))) (. .))
```

Evaluation Example

```
GOLD = (S (NP (DT The) (JJ Japanese) (JJ industrial) (NNS companies))
          (VP (MD should) (VP (VB know) (ADVP (JJR better)))) (. .)

CHAR = (S (NP (DT The) (JJ Japanese) (JJ industrial) (NNS companies))
           (VP (MD should) (VP (VB know)) ((ADVP (RBR better)))) (. .))
```

```
Bracketing Recall          =  80.00
Bracketing Precision       =  66.67
Bracketing FMeasure        =  72.73
Complete match             =   0.00
No crossing                = 100.00
Tagging accuracy           =  87.50
```

NLP