Xử lý ngôn ngữ tự nhiên

Chương 6: Parsing và Grammar

Khoa CNTT, Đại học Kỹ thuật - Công nghệ Cần Thơ

Lưu hành nội bộ

Nội dung

Syntax

- Cú pháp (syntax) quy định sự sắp xếp các từ
  - Nhóm từ đi cùng nhau (constituency)
  - Quan hệ văn phạm (grammatical relations)
  - Các quy tắt lệ thuộc: thứ tự các từ loại

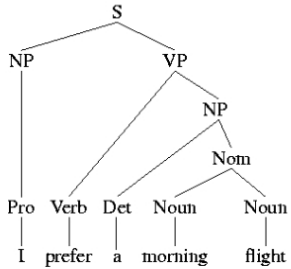
Context-free grammar and trees

- Mô hình toán học mô tả cấu trúc ngôn ngữ
- Terminals và non-terminals
- Nguồn gốc (derivation)
- Parse tree
- Start symbol

Noun → *flight* | *breeze* | *trip* | *morning* | ...
Verb → *is* | *prefer* | *like* | *need* | *want* | *fly* ...
Adjective → *cheapest* | *non-stop* | *first* | *latest* | *other* | *direct* | ...
Pronoun → *me* | *I* | *you* | *it* | ...
Proper-Noun → *Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American* | ...
Determiner → *the* | *a* | *an* | *this* | *these* | *that* | ...
Preposition → *from* | *to* | *on* | *near* | ...
Conjunction → *and* | *or* | *but* | ...

The lexicon for L_0

S → *NP VP*	I + want a morning flight
NP → *Pronoun*	I
| *Proper-Noun*	Los Angeles
| *Det Nominal*	a + flight
Nominal → *Noun Nominal*	morning + flight
| *Noun*	flights
VP → *Verb*	do
| *Verb NP*	want + a flight
| *Verb NP PP*	leave + Boston + in the morning
| *Verb PP*	leaving + on Thursday
PP → *Preposition NP*	from + Los Angeles



The grammar for L_0

Parsing

- Parsing: sự phân tích ngữ pháp (từ, câu)
- Còn được gọi là syntactic analysis
- Là quá trình xác định xem một chuỗi các words liên tục nhau có đúng ngữ pháp hay không
- Chia nhỏ câu thành từ, ngữ và xác định nhóm thành phần: noun - danh từ, verb - động từ
- Tổng hợp hóa tác vụ POS

Grammatically vs. semantically

- Ví dụ câu: `John bought a book`
  - Câu đúng ngữ pháp (grammatically correct)
- Ví dụ câu: `book bought a John`
  - Câu đúng ngữ pháp (grammatically correct)
  - Câu sai ngữ nghĩa (semantically incorrect)

Nhóm Parsing

- Parsing được chia thành 2 nhóm:
  - Top-down parsing: bắt đầu từ ký tự bắt đầu (start symbol) và tiếp tục cho đến khi gặp các thành phần đơn (individual components)
  - Bottom-up parsing: bắt đầu từ các thành phần đơn và tiếp tục cho đến ký tự bắt đầu

Treebank corpus

- Treebank là corpus gồm 199 documents được cung cấp kèm theo thư viện NLTK
- https://en.wikipedia.org/wiki/Treebank

```
1 import nltk
2 import nltk.corpus
3 from nltk.corpus import treebank
4
5 print(nltk.corpus.treebank.fileids())
6 print(len(nltk.corpus.treebank.fileids()))
7
8 #199
```

['wsj_0001.mrg', 'wsj_0002.mrg', 'wsj_0003.mrg', 'wsj_0004.mrg', 'wsj_0005.mrg', 'wsj_0006.mrg', 'wsj_0007.mrg', 'wsj_0008.
mrg', 'wsj_0009.mrg', 'wsj_0010.mrg', 'wsj_0011.mrg', 'wsj_0012.mrg', 'wsj_0013.mrg', 'wsj_0014.mrg', 'wsj_0015.mrg', 'wsj_
0016.mrg', 'wsj_0017.mrg', 'wsj_0018.mrg', 'wsj_0019.mrg', 'wsj_0020.mrg', 'wsj_0021.mrg', 'wsj_0022.mrg', 'wsj_0023.mrg',
'wsj_0024.mrg', 'wsj_0025.mrg', 'wsj_0026.mrg', 'wsj_0027.mrg', 'wsj_0028.mrg', 'wsj_0029.mrg', 'wsj_0030.mrg', 'wsj_0031.m
rg', 'wsj_0032.mrg', 'wsj_0033.mrg', 'wsj_0034.mrg', 'wsj_0035.mrg', 'wsj_0036.mrg', 'wsj_0037.mrg', 'wsj_0038.mrg', 'wsj_0
039.mrg', 'wsj_0040.mrg', 'wsj_0041.mrg', 'wsj_0042.mrg', 'wsj_0043.mrg', 'wsj_0044.mrg', 'wsj_0045.mrg', 'wsj_0046.mrg', '
wsj_0047.mrg', 'wsj_0048.mrg', 'wsj_0049.mrg', 'wsj_0050.mrg', 'wsj_0051.mrg', 'wsj_0052.mrg', 'wsj_0053.mrg', 'wsj_0054.mr
g', 'wsj_0055.mrg', 'wsj_0056.mrg', 'wsj_0057.mrg', 'wsj_0058.mrg', 'wsj_0059.mrg', 'wsj_0060.mrg', 'wsj_0061.mrg', 'wsj_00
62.mrg', 'wsj_0063.mrg', 'wsj_0064.mrg', 'wsj_0065.mrg', 'wsj_0066.mrg', 'wsj_0067.mrg', 'wsj_0068.mrg', 'wsj_0069.mrg', 'w
sj_0070.mrg', 'wsj_0071.mrg', 'wsj_0072.mrg', 'wsj_0073.mrg', 'wsj_0074.mrg', 'wsj_0075.mrg', 'wsj_0076.mrg', 'wsj_0077.mrg
', 'wsj_0078.mrg', 'wsj_0079.mrg', 'wsj_0080.mrg', 'wsj_0081.mrg', 'wsj_0082.mrg', 'wsj_0083.mrg', 'wsj_0084.mrg', 'wsj_008
5.mrg', 'wsj_0086.mrg', 'wsj_0087.mrg', 'wsj_0088.mrg', 'wsj_0089.mrg', 'wsj_0090.mrg', 'wsj_0091.mrg', 'wsj_0092.mrg', 'ws
j_0093.mrg', 'wsj_0094.mrg', 'wsj_0095.mrg', 'wsj_0096.mrg', 'wsj_0097.mrg', 'wsj_0098.mrg', 'wsj_0099.mrg', 'wsj_0100.mrg
', 'wsj_0101.mrg', 'wsj_0102.mrg', 'wsj_0103.mrg', 'wsj_0104.mrg', 'wsj_0105.mrg', 'wsj_0106.mrg', 'wsj_0107.mrg', 'wsj_0108
.mrg', 'wsj_0109.mrg', 'wsj_0110.mrg', 'wsj_0111.mrg', 'wsj_0112.mrg', 'wsj_0113.mrg', 'wsj_0114.mrg', 'wsj_0115.mrg', 'wsj
_0116.mrg', 'wsj_0117.mrg', 'wsj_0118.mrg', 'wsj_0119.mrg', 'wsj_0120.mrg', 'wsj_0121.mrg', 'wsj_0122.mrg', 'wsj_0123.mrg',
'wsj_0124.mrg', 'wsj_0125.mrg', 'wsj_0126.mrg', 'wsj_0127.mrg', 'wsj_0128.mrg', 'wsj_0129.mrg', 'wsj_0130.mrg', 'wsj_0131.m
rg', 'wsj_0132.mrg', 'wsj_0133.mrg', 'wsj_0134.mrg', 'wsj_0135.mrg', 'wsj_0136.mrg', 'wsj_0137.mrg', 'wsj_0138.mrg', 'wsj_0
139.mrg', 'wsj_0140.mrg', 'wsj_0141.mrg', 'wsj_0142.mrg', 'wsj_0143.mrg', 'wsj_0144.mrg', 'wsj_0145.mrg', 'wsj_0146.mrg', '
wsj_0147.mrg', 'wsj_0148.mrg', 'wsj_0149.mrg', 'wsj_0150.mrg', 'wsj_0151.mrg', 'wsj_0152.mrg', 'wsj_0153.mrg', 'wsj_0154.mr
g', 'wsj_0155.mrg', 'wsj_0156.mrg', 'wsj_0157.mrg', 'wsj_0158.mrg', 'wsj_0159.mrg', 'wsj_0160.mrg', 'wsj_0161.mrg', 'wsj_01
62.mrg', 'wsj_0163.mrg', 'wsj_0164.mrg', 'wsj_0165.mrg', 'wsj_0166.mrg', 'wsj_0167.mrg', 'wsj_0168.mrg', 'wsj_0169.mrg', 'w
sj_0170.mrg', 'wsj_0171.mrg', 'wsj_0172.mrg', 'wsj_0173.mrg', 'wsj_0174.mrg', 'wsj_0175.mrg', 'wsj_0176.mrg', 'wsj_0177.mrg
', 'wsj_0178.mrg', 'wsj_0179.mrg', 'wsj_0180.mrg', 'wsj_0181.mrg', 'wsj_0182.mrg', 'wsj_0183.mrg', 'wsj_0184.mrg', 'wsj_018
5.mrg', 'wsj_0186.mrg', 'wsj_0187.mrg', 'wsj_0188.mrg', 'wsj_0189.mrg', 'wsj_0190.mrg', 'wsj_0191.mrg', 'wsj_0192.mrg', 'ws
j_0193.mrg', 'wsj_0194.mrg', 'wsj_0195.mrg', 'wsj_0196.mrg', 'wsj_0197.mrg', 'wsj_0198.mrg', 'wsj_0199.mrg']

```
9
10 print(treebank.words('wsj_0007.mrg'))
11 print(len(treebank.words('wsj_0007.mrg')))
12 print(treebank.tagged_words('wsj_0007.mrg'))
13
14 #['McDermott', 'International', 'Inc.', 'said', '0', ...]
15 #75
16 #[('McDermott', 'NNP'), ('International', 'NNP'), ...]
```

Penn Treebank corpus

```
1 import nltk
2 from nltk.corpus import treebank
3 print(treebank.sents('wsj_0007.mrg')[2])
4 print(treebank.parsed_sents('wsj_0007.mrg')[2])
5
6 #['Bailey', 'Controls', ',', 'based', '*', 'in', 'Wickliffe', ',', '
    Ohio', ',', 'makes', 'computerized', 'industrial', 'controls', '
    systems', '.']
```

```
(S
  (NP-SBJ
    (NP (NNP Bailey) (NNP Controls))
    (, ,)
    (VP
      (VBN based)
      (NP (-NONE- *))
      (PP-LOC-CLR
        (IN in)
        (NP (NP (NNP Wickliffe)) (, ,) (NP (NNP Ohio)))))
    (, ,))
  (VP
    (VBZ makes)
    (NP
      (JJ computerized)
      (JJ industrial)
      (NNS controls)
      (NNS systems)))
  (. .))
```

POS tags in Penn Treebank

1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential there
5	FW	Foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker

POS tags in Penn Treebank

11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb

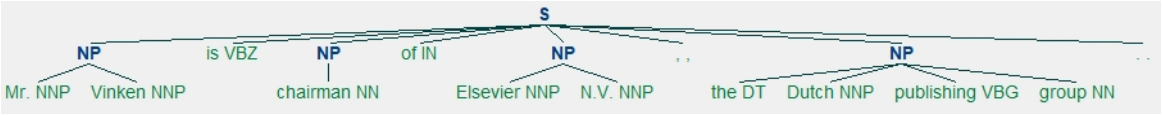
POS tags in Penn Treebank

21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	*to*
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle

POS tags in Penn Treebank

31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb

```
import nltk
from nltk.corpus import treebank_chunk
print(treebank_chunk.chunked_sents()[1])
treebank_chunk.chunked_sents()[1].draw()
```



```
import nltk
from nltk.corpus import treebank_chunk

print(treebank_chunk.chunked_sents()[1])

print(treebank_chunk.chunked_sents()[1].leaves())
#[('Mr.', 'NNP'), ('Vinken', 'NNP'), ('is', 'VBZ'), ('chairman', 'NN'),
    ('of', 'IN'), ('Elsevier', 'NNP'), #('N.V.', 'NNP'), (',', ','),
   ('the', 'DT'), ('Dutch', 'NNP'), ('publishing', 'VBG'), ('group', '
   NN'), ('.', '.')]

print(treebank_chunk.chunked_sents()[1].pos())
#[(('Mr.', 'NNP'), 'NP'), (('Vinken', 'NNP'), 'NP'), (('is', 'VBZ'), 'S
    '), (('chairman', 'NN'), 'NP'), (('of', #'IN'), 'S'), (('Elsevier',
    'NNP'), 'NP'), (('N.V.', 'NNP'), 'NP'), ((',', ','), 'S'), (('the
    ', 'DT'), 'NP'), #(('Dutch', 'NNP'), 'NP'), (('publishing', 'VBG'),
    'NP'), (('group', 'NN'), 'NP'), (('.', '.'), 'S')]
```

```
11
12 print(treebank_chunk.chunked_sents()[1].productions())
13
14 #[S -> NP ('is', 'VBZ') NP ('of', 'IN') NP (',', ',') NP ('.', '.'), NP
     -> ('Mr.', 'NNP') ('Vinken', 'NNP'), NP -> #('chairman', 'NN'), NP
     -> ('Elsevier', 'NNP') ('N.V.', 'NNP'), NP -> ('the', 'DT') ('
    Dutch', 'NNP') #('publishing', 'VBG') ('group', 'NN')]
```

Context Free Grammar (CFG)

- Văn phạm phi ngữ cảnh (CFG) bao gồm các thành phần:
  - Một tập các nút không kết thúc (non terminal nodes) (N)
  - Một tập các nút kết thúc (terminal nodes) (T)
  - Ký hiệu bắt đầu (S)
  - Tập các luật dẫn xuất (production rules) (P) có dạng $A \rightarrow a$

Các luật CFG

- Luật cấu trúc ngữ (phrase structure rules) xác định thông qua các luật dẫn xuất $A \rightarrow a$
- Luật cấu trúc câu (sentence structure rules)
  - Cấu trúc tường thuật (declarative structure): chủ ngữ - vị ngữ
  - Cấu trúc mệnh lệnh (imperative structure): câu ra lệnh, bắt đầu bằng động từ
  - Cấu trúc Yes-No
  - Cấu trúc câu hỏi Wh (Wh-question)

Các luật CFG

- S $\rightarrow$ NP VP
- S $\rightarrow$ VP
- S $\rightarrow$ Aux NP VP
- S $\rightarrow$ Wh-NP VP
- S $\rightarrow$ Wh-NP Aux NP VP
- NP $\rightarrow$ (Det) (AP) Nom (PP)
- VP $\rightarrow$ Verb (NP) (NP) (PP)

Ví dụ CFG 1

```
1  import nltk
2  from nltk import Nonterminal, nonterminals, Production, CFG
3  nonterminal1 = Nonterminal('NP')
4  nonterminal2 = Nonterminal('VP')
5  nonterminal3 = Nonterminal('PP')
6
7  print(nonterminal1.symbol())
8  print(nonterminal2.symbol())
9  print(nonterminal3.symbol())
10 #NP
11 #VP
12 #PP
13
14 print(nonterminal1==nonterminal2)
15 print(nonterminal2==nonterminal3)
16 print(nonterminal1==nonterminal3)
```

```
#False
#False
#False

S, NP, VP, PP = nonterminals('S, NP, VP, PP')
N, V, P, DT = nonterminals('N, V, P, DT')
production1 = Production(S, [NP, VP])
production2 = Production(NP, [DT, NP])
production3 = Production(VP, [V, NP,NP,PP])

print(production1.lhs())
print(production1.rhs())
print(production3.lhs())
print(production3.rhs())
#S
#(NP, VP)
#VP
#(V, NP, NP, PP)
```

```
35
36 print(production3 == Production(VP, [V,NP,NP,PP]))
37 print(production2 == production3)
38 #True
39 #False
```

Ví dụ CFG 2

```
from nltk import Nonterminal, nonterminals, Production, CFG
nt1 = Nonterminal('NP')
nt2 = Nonterminal('VP')
print(nt1.symbol())
print(nt1 == Nonterminal('NP'))
#NP
#True

S, NP, VP, PP = nonterminals('S, NP, VP, PP')
N, V, P, DT = nonterminals('N, V, P, DT')
prod1 = Production(S, [NP, VP])
prod2 = Production(NP, [DT, NP])
print(prod1.lhs())
print(prod1.rhs())
print (prod1 == Production(S, [NP, VP]))
```

```
#S
#(NP, VP)
#True

grammar = CFG.fromstring("""
    S -> NP VP
    PP -> P NP
    NP -> 'the' N | N PP | 'the' N PP
    VP -> V NP | V PP | V NP PP
    N -> 'cat'
    N -> 'dog'
    N -> 'rug'
    V -> 'chased'
    V -> 'sat'
    P -> 'in'
    P -> 'on'
""")
print(grammar)
```

```
34 #Grammar with 15 productions (start state = S)
35 #   S -> NP VP
36 #   PP -> P NP
37 #   NP -> 'the' N
38 #   NP -> N PP
39 #   NP -> 'the' N PP
40 #   VP -> V NP
41 #   VP -> V PP
42 #   VP -> V NP PP
43 #   N -> 'cat'
44 #   N -> 'dog'
45 #   N -> 'rug'
46 #   V -> 'chased'
47 #   V -> 'sat'
48 #   P -> 'in'
49 #   P -> 'on'
```

ATIS grammar

- Các luật dẫn xuất được cung cấp bởi các bộ văn phạm

```
import nltk
gram1 = nltk.data.load('grammars/large_grammars/atis.cfg')
print(gram1)

#Grammar with 5517 productions (start state = SIGMA)
#ABBCL_NP -> QUANP_DTI QUANP_DTI QUANP_CD AJP_JJ NOUN_NP PRPRTCL_VBG
#ADJ_ABL -> only
#ADJ_ABL -> such
#ADJ_AP -> pt_adj_ap
#ADJ_AP -> other
#ADJ_AP -> last
#ADJ_AP -> less
#ADJ_AT -> the
#...
```

Trích xuất câu trong ATIS grammar

```
1  import nltk
2  sent = nltk.data.load('grammars/large_grammars/atis_sentences.txt')
3  sent = nltk.parse.util.extract_test_sentences(sent)
4  print(len(sent))
5  testingsent=sent[25]
6  print(testingsent)
7  print(testingsent[0])
8  print(testingsent[1])
9
10 #98
11 #(['list', 'those', 'flights', 'that', 'stop', 'over', 'in', 'salt', '
       lake', 'city', '.'], 11)
12 #['list', 'those', 'flights', 'that', 'stop', 'over', 'in', 'salt', '
       lake', 'city', '.']
13 #11
```

Recursive Descent Parser

```
from nltk import CFG
from nltk.parse import RecursiveDescentParser

grammar = CFG.fromstring("""
    S -> NP VP
    PP -> P NP
    NP -> 'the' N | N PP | 'the' N PP
    VP -> V NP | V PP | V NP PP
    N -> 'cat'
    N -> 'dog'
    N -> 'rug'
    V -> 'chased'
    V -> 'sat'
    P -> 'in'
    P -> 'on'
""")
```

```
17
18 rd = RecursiveDescentParser(grammar)
19
20 sentence1 = 'the cat chased the dog'.split()
21 sentence2 = 'the cat chased the dog on the rug'.split()
22
23 for t in rd.parse(sentence1):
24     print(t)
25 for t in rd.parse(sentence2):
26     print(t)
27
28 #(S (NP the (N cat)) (VP (V chased) (NP the (N dog))))
29 #(S
30 #(NP the (N cat))
31 #(VP (V chased) (NP the (N dog) (PP (P on) (NP the (N rug))))))
32 #(S
33 #(NP the (N cat))
34 #(VP (V chased) (NP the (N dog)) (PP (P on) (NP the (N rug)))))
```

Chart Parser

```
1  import nltk
2
3  sent = 'I saw a dog'
4  nltk.parse.chart.demo(2, print_times=False, trace=1, sent=sent,
       numparses=1)
5
6  #* Sentence:
7  #I saw a dog
8  #['I', 'saw', 'a', 'dog']
9  #
10 #* Strategy: Bottom-up
```

```
|.    I    .   saw   .    a    .   dog   .|
|[---------]         .         .         .| [0:1] 'I'
|.         [---------]         .         .| [1:2] 'saw'
|.         .         [---------]         .| [2:3] 'a'
|.         .         .         [---------]| [3:4] 'dog'
|>         .         .         .         .| [0:0] NP -> * 'I'
|[---------]         .         .         .| [0:1] NP -> 'I' *
|>         .         .         .         .| [0:0] S  -> * NP VP
|>         .         .         .         .| [0:0] NP -> * NP PP
|[--------->         .         .         .| [0:1] S  -> NP * VP
|[--------->         .         .         .| [0:1] NP -> NP * PP
|.         >         .         .         .| [1:1] Verb -> * 'saw'
|.         [---------]         .         .| [1:2] Verb -> 'saw' *
|.         >         .         .         .| [1:1] VP -> * Verb NP
|.         >         .         .         .| [1:1] VP -> * Verb
|.         [---------->         .         .| [1:2] VP -> Verb * NP
|.         [---------]         .         .| [1:2] VP -> Verb *
|.         >         .         .         .| [1:1] VP -> * VP PP
|[--------------------]         .         .| [0:2] S  -> NP VP *
|.         [---------->         .         .| [1:2] VP -> VP * PP
```

```
import nltk
sent = 'I saw John with a dog'
nltk.parse.chart.demo(1, print_times=False, trace=0, sent=sent,
    numparses=2)

#* Sentence:
#I saw John with a dog
#['I', 'saw', 'John', 'with', 'a', 'dog']
#
#* Strategy: Top-down
#
#Nr edges in chart: 48
#(S
#(NP I)
#(VP (Verb saw) (NP (NP John) (PP with (NP (Det a) (Noun dog))))))
#(S
#(NP I)
#(VP (VP (Verb saw) (NP John)) (PP with (NP (Det a) (Noun dog)))))
```

```
1 import nltk
2 sent = 'I saw John with a dog'
3 nltk.parse.chart.demo(5, print_times=False, trace=1, sent=sent,
      numparses=2)
4
5 #* Sentence:
6 #I saw John with a dog
7 #['I', 'saw', 'John', 'with', 'a', 'dog']
8 #
9 #* Strategy: Stepping (top-down vs bottom-up)
```

```
*** SWITCH TO TOP DOWN
|[------]      .      .      .      .      .| [0:1] 'I'
|.      [------]      .      .      .      .| [1:2] 'saw'
|.      .      [------]      .      .      .| [2:3] 'John'
|.      .      .      [------]      .      .| [3:4] 'with'
|.      .      .      .      [------]      .| [4:5] 'a'
|.      .      .      .      .      [------]| [5:6] 'dog'
|>      .      .      .      .      .      .| [0:0] S  -> * NP VP
|>      .      .      .      .      .      .| [0:0] NP -> * NP PP
|>      .      .      .      .      .      .| [0:0] NP -> * Det Noun
|>      .      .      .      .      .      .| [0:0] NP -> * 'I'
|[------]      .      .      .      .      .| [0:1] NP -> 'I' *
|[------>      .      .      .      .      .| [0:1] S  -> NP * VP
|[------>      .      .      .      .      .| [0:1] NP -> NP * PP
|.      >      .      .      .      .      .| [1:1] VP -> * VP PP
|.      >      .      .      .      .      .| [1:1] VP -> * Verb NP
|.      >      .      .      .      .      .| [1:1] VP -> * Verb
|.      >      .      .      .      .      .| [1:1] Verb -> * 'saw'
|.      [------]      .      .      .      .| [1:2] Verb -> 'saw' *
|.      [------>      .      .      .      .| [1:2] VP -> Verb * NP
```

Probabilistic CFG

- Là dạng cú pháp CFG có thêm thông tin về xác suất xảy ra luật

```
import nltk
from nltk.corpus import treebank
from itertools import islice
from nltk.grammar import PCFG, induce_pcfg, toy_pcfg1, toy_pcfg2

gram2 = PCFG.fromstring("""
    A -> B B [.3] | C B C [.7]
    B -> B D [.5] | C [.5]
    C -> 'a' [.1] | 'b' [0.9]
    D -> 'b' [1.0]
""")
```

```
prod1 = gram2.productions()[0]
print(prod1)
prod2 = gram2.productions()[1]
print(prod2)
#A -> B B [0.3]
#A -> C B C [0.7]

print(prod2.lhs())
print(prod2.rhs())
print((prod2.prob()))
print(gram2.start())
print(gram2.productions())

#A
#(C, B, C)
#0.7
#A
#[A -> B B [0.3], A -> C B C [0.7], B -> B D [0.5], B -> C [0.5], C ->
    'a' [0.1], C -> 'b' [0.9], D -> 'b' [1.0]]
```

Kết hợp grammar và parser

```
import nltk
from nltk.corpus import treebank
from itertools import islice
from nltk.grammar import PCFG, induce_pcfg, toy_pcfg1, toy_pcfg2
grammar = toy_pcfg2
print(grammar)

#Grammar with 23 productions (start state = S)
#S -> NP VP [1.0]
#VP -> V NP [0.59]
#VP -> V [0.4]
#VP -> VP PP [0.01]
#NP -> Det N [0.41]
#NP -> Name [0.28]
#NP -> NP PP [0.31]
#PP -> P NP [1.0]
```

```
from nltk.parse import pchart

tokens = "Jack saw Bob with my cookie".split()
parser = pchart.InsideChartParser(grammar)
for t in parser.parse(tokens):
    print(t)
#(S
#(NP (Name Jack))
#(VP
#(V saw)
#(NP
#(NP (Name Bob))
#(PP (P with) (NP (Det my) (N cookie)))))) (p=6.31607e-06)
#(S
#(NP (Name Jack))
#(VP
#(VP (V saw) (NP (Name Bob)))
#(PP (P with) (NP (Det my) (N cookie))))) (p=2.03744e-07)
```

Visualization

- https://demos.explosion.ai/displacy/