




MARCH 31, 2019

NATURAL LANGUAGE PROCESSING

HOMEWORK ASSIGNMENT 2

MS INFORMATION MANAGEMENT
rnrathod@syr.edu



Developing context-free grammar that can parse all the following sentences.

- (a) “Today is a nice day”
- (b) “Bob and Mary went to France last month again”
- (c) “You can say that again”
- (d) “Birds are not necessarily able to fly”

I have use Python environment to test my grammar to make sure that it does the job. First, I have import nltk package, then I defined my first sentence into variable text 1. Than I have tokenized the sentence. For tokenization first I have use sent_tokenize to produce a list of text strings for individual sentence so that tokenization can be done better. After producing the list of sentence texts, I have applied the word tokenizer to the sentence. After tokenization I applied Stanford POS tagger to each sentence of tokens in the list to get tag of each word. Similar process was followed for all other sentences. I proceeded with top-down approach on all 4 sentences to come up with the following grammar:

The Grammar

```

S -> NP VP
NP -> Prop CONJ Prop | Prop | Det N | Det ADJ N
VP -> V PP NP | V NP | V VP | V Det ADV | V AP | Det V
AP -> ADV AP | ADJ VP
PP -> Det NP
ADV -> 'again' | 'necessarily' | 'not'
ADJ -> 'nice' | 'able'
Det -> 'to' | 'last' | 'a' | 'that'
CONJ -> 'and'
N -> 'month' | 'day'
V -> 'went' | 'is' | 'can' | 'say' | 'fly' | 'are'
Prop -> 'Bob' | 'France' | 'Today' | 'You' | 'Birds' | 'Mary'
""")

```

Part 1

a. “Today is a nice day”

(S (NP (Prop Today)) (VP (V is) (NP (Det a) (ADJ nice) (N day))))

The above output is obtained after parsing the sentence using above grammar

b. “Bob and Mary went to France last month again”

```

(S
  (NP (Prop Bob) (CONJ and) (Prop Mary))
  (VP (V went) (NP
    (PP (P to) (NP (Prop France) (DET last) (N month))) (ADV

```

again))))))

The above output is obtained after parsing the sentence using above grammar

c. "You can say that again"

(S (NP (Prop You)) (VP (V can) (VP (V say) (Det that) (ADV again))))))

The above output is obtained after parsing the sentence using above grammar

d. "Birds are not necessarily able to fly"

(S
 (NP (Prop Birds))
 (VP
 (V are)
 (AP
 (ADV not)
 (AP (ADV necessarily) (AP (ADJ able) (VP (Det to) (V fly)))))))))

The above output is obtained after parsing the sentence using above grammar

Part 2

For part 2 of the assignment besides these four sentences, we have to list up to three different sentences that can be parsed by this grammar. I have generated three different sentences which can be parsed by this grammar. The sentences are syntactically correct as it has been made using this grammar but semantically, they are not correct.

a. "You can fly to France Today"

The above sentence was made by me using the grammar. The sentence is semantically correct English sentence. The sentence was fully parsed by the grammar and tree generated is as follows.

(S
 (NP (Prop You))
 (VP
 (V can)
 (VP (V fly) (PP (Det to) (NP (Prop France))) (NP (Prop Today))))))

b. "Bob went to France Today"

The above sentence was made by me using the grammar. The sentence is semantically correct English sentence. The sentence was fully parsed by the grammar and tree generated is as follows.

(S
 (NP (Prop Bob))
 (VP (V went) (PP (Det to) (NP (Prop France))) (NP (Prop Today))))

c. **"Bob is not able to say"**

The above sentence was made by me using the grammar. The sentence is semantically correct English sentence. The sentence was fully parsed by the grammar and tree generated is as follows.

(S
 (NP (Prop Bob))
 (VP (V is) (AP (ADV not) (AP (ADJ able) (VP (Det to) (V say))))))

Part 3

Developing Probabilistic context free grammar

First, I evenly divided the probability equally among each element. Then I developed the following grammar to parse the sentences

S -> NP VP [1.0]
 NP -> Prop[0.4] | Det N[0.3] | Det ADJ N[0.3]
 VP -> V PP NP [0.16] | V NP [0.20] | V VP [0.16] | V Det ADV [0.16] | V AP [0.16] | Det V [0.16]
 AP -> ADV AP[0.5] | ADJ VP [0.5]
 PP -> Det NP [1.0]
 ADV -> 'again' [0.3] | 'necessarily' [0.3] | 'not' [0.4]
 ADJ -> 'nice' [0.5] | 'able' [0.5]
 Det -> 'to' [0.25] | 'last' [0.25] | 'a' [0.25] | 'that'[0.25]
 N -> 'month' [0.5] | 'day' [0.5]
 V -> 'went' [0.16] | 'is' [0.16] | 'can' [0.16] | 'say' [0.16] | 'fly' [0.20] | 'are' [0.16]
 Prop -> 'Bob'[0.4] | 'France' [0.18] | 'Today' [0.18] | 'You' [0.15] | 'Birds' [0.09] | 'Mary' [0.4]
 """)

a. **"Today is a nice day"**

With probability 4.32e-05 the below tree is obtained using the grammar

(S
 (NP (Prop Today))
 (VP (V is) (NP (Det a) (ADJ nice) (N day)))) (p=4.32e-05)

b. **“Bob and Mary went to France last month again”**

With probability 3.13632e-09 the below tree is obtained using the grammar

```
(S
  (NP (Prop Bob) (CC and) (Prop Mary))
  (VP
    (V went) (NP
      (PP
        (P to)
        (NP (Prop France) (JJ last) (N month))
        (RB again)))))) (p=3.13632e-09)
```

c. **“You can say that again”**

With probability 2.94912e-06 the below tree is obtained using the grammar

```
(S
  (NP (Prop You))
  (VP (V can) (VP (V say) (Det that) (ADV again)))) (p=2.94912e-06)
```

d. **“Birds are not necessarily able to fly”**

With probability 5.5296e-08 the below tree is obtained using the grammar

```
(S
  (NP (Prop Birds))
  (VP
    (V are)
    (AP
      (ADV not)
      (AP
        (ADV necessarily)
        (AP (ADJ able) (VP (Det to) (V fly))))))) (p=5.5296e-08)
```

The following are the generated sentence using the grammar in Part 1 of the assignment which is tested upon the Probabilistic Context Free Grammar.

a. **"You can fly to France Today"**

With probability 6.3701e-08 the below tree is obtained using the grammar

```
(S
  (NP (Prop You))
  (VP
```

(V can)
 (VP
 (V fly)
 (PP (Det to) (NP (Prop France)))
 (NP (Prop Today)))) (p=6.3701e-08)

b. "Bob went to France Today"

With probability 5.30842e-06 the below tree is obtained using the grammar

(S
 (NP (Prop Bob))
 (VP
 (V went)
 (PP (Det to) (NP (Prop France)))
 (NP (Prop Today)))) (p=5.30842e-06)

c. "Bob is not able to say"

With probability 1.31072e-06 the below tree is obtained using the grammar

(S
 (NP (Prop Bob))
 (VP
 (V is)
 (AP (ADV not) (AP (ADJ able) (VP (Det to) (V say)))))) (p=1.31072e-06)

Appendix

Outputs

Part 1

e. "Today is a nice day"

(S (NP (Prop Today)) (VP (V is) (NP (Det a) (ADJ nice) (N day))))

f. "Bob and Mary went to France last month again"

(S
 (NP (Prop Bob) (CONJ and) (Prop Mary))
 (VP (V went) (NP
 (PP (P to) (NP (Prop France) (DET last) (N month)) (ADV
 again))))))

g. "You can say that again"

(S (NP (Prop You)) (VP (V can) (VP (V say) (Det that) (ADV again))))

h. "Birds are not necessarily able to fly"

(S
 (NP (Prop Birds))
 (VP
 (V are)
 (AP
 (ADV not)
 (AP (ADV necessarily) (AP (ADJ able) (VP (Det to) (V fly))))))

Part 2

d. "You can fly to France Today"

(S
 (NP (Prop You))
 (VP
 (V can)
 (VP (V fly) (PP (Det to) (NP (Prop France))) (NP (Prop Today))))

e. "Bob went to France Today"

(S
 (NP (Prop Bob))
 (VP (V went) (PP (Det to) (NP (Prop France))) (NP (Prop Today))))

f. **"Bob is not able to say"**

(S
 (NP (Prop Bob))
 (VP (V is) (AP (ADV not) (AP (ADJ able) (VP (Det to) (V say))))))

Part 3

e. **"Today is a nice day"**

(S
 (NP (Prop Today))
 (VP (V is) (NP (Det a) (ADJ nice) (N day)))) (p=4.32e-05)

f. **"Bob and Mary went to France last month again"**

(S
 (NP (Prop Bob) (CC and) (Prop Mary))
 (VP
 (V went) (NP
 (PP
 (P to)
 (NP (Prop France) (JJ last) (N month))
 (RB again)))))) (p=3.13632e-09)

g. **"You can say that again"**

(S
 (NP (Prop You))
 (VP (V can) (VP (V say) (Det that) (ADV again)))) (p=2.94912e-06)

h. **"Birds are not necessarily able to fly"**

(S
 (NP (Prop Birds))
 (VP
 (V are)
 (AP
 (ADV not)
 (AP
 (ADV necessarily)

(AP (ADJ able) (VP (Det to) (V fly)))))) (p=5.5296e-08)

d. "You can fly to France Today"

(S
 (NP (Prop You))
 (VP
 (V can)
 (VP
 (V fly)
 (PP (Det to) (NP (Prop France)))
 (NP (Prop Today)))))) (p=6.3701e-08)

e. "Bob went to France Today"

(S
 (NP (Prop Bob))
 (VP
 (V went)
 (PP (Det to) (NP (Prop France)))
 (NP (Prop Today)))) (p=5.30842e-06)

f. "Bob is not able to say"

(S
 (NP (Prop Bob))
 (VP
 (V is)
 (AP (ADV not) (AP (ADJ able) (VP (Det to) (V say)))))) (p=1.31072e-06)

Screenshot

Part 1

```
In [62]: grammar = nltk.CFG.fromstring("""
S -> NP VP
NP -> Prop CONJ Prop | Prop | Det N | Det ADJ N
VP -> V PP NP | V NP | V VP | V Det ADV | V AP | Det V
AP -> ADV AP | ADJ VP
PP -> Det NP
ADV -> 'again' | 'necessarily' | 'not'
ADJ -> 'nice' | 'able'
Det -> 'to' | 'last' | 'a' | 'that'
CONJ -> 'and'
N -> 'month' | 'day'
V -> 'went' | 'is' | 'can' | 'say' | 'fly' | 'are'
Prop -> 'Bob' | 'France' | 'Today' | 'You' | 'Birds' | 'Mary'
""")
```

(a) "Today is a nice day"

```

In [3]: import nltk

In [4]: text1 = 'Today is a nice day'

In [5]: textsplit1 = nltk.sent_tokenize(text1)
textsplit1

Out[5]: ['Today is a nice day']

In [6]: tokentext1 = [nltk.word_tokenize(sent) for sent in textsplit1]
tokentext1

Out[6]: [['Today', 'is', 'a', 'nice', 'day']]

In [7]: taggedtext1 = [nltk.pos_tag(tokens) for tokens in tokentext1]
taggedtext1

Out[7]: [['Today', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('nice', 'JJ'), ('day', 'NN')]]

In [63]: rd_parser = nltk.RecursiveDescentParser(grammar)

In [54]: sentlist = "Today is a nice day".split()
for tree in rd_parser.parse(sentlist):
    print (tree)

(S (NP (Prop Today)) (VP (V is) (NP (Det a) (ADJ nice) (N day))))

```

(b) “Bob and Mary went to France last month again”

```

In [8]: text2 = 'Bob and Mary went to France last month again'

In [9]: textsplit2 = nltk.sent_tokenize(text2)
textsplit2

Out[9]: ['Bob and Mary went to France last month again']

In [10]: tokentext2 = [nltk.word_tokenize(sent) for sent in textsplit2]
tokentext2

Out[10]: [['Bob', 'and', 'Mary', 'went', 'to', 'France', 'last', 'month', 'again']]

In [11]: taggedtext2 = [nltk.pos_tag(tokens) for tokens in tokentext2]
taggedtext2

Out[11]: [['Bob', 'NNP'),
           ('and', 'CC'),
           ('Mary', 'NNP'),
           ('went', 'VBD'),
           ('to', 'TO'),
           ('France', 'NNP'),
           ('last', 'JJ'),
           ('month', 'NN'),
           ('again', 'RB')]]

In [463]: sent6list = 'Bob and Mary went to France last month again'.split()
for tree in rd_parser.parse(sent6list):
    print (tree)

(S
  (NP (Prop Bob) (CC and) (Prop Mary))
  (VP
    (V went)
    (NP
      (PP (P to) (NP (Prop France) (JJ last) (N month)) (RB again))))))

```

(b) “You can say that again”

```

In [12]: text3 = 'You can say that again'

In [13]: textsplit3 = nltk.sent_tokenize(text3)
textsplit3

Out[13]: ['You can say that again']

In [14]: tokentext3 = [nltk.word_tokenize(sent) for sent in textsplit3]
tokentext3

Out[14]: [['You', 'can', 'say', 'that', 'again']]

In [15]: taggedtext3 = [nltk.pos_tag(tokens) for tokens in tokentext3]
taggedtext3

Out[15]: [[('You', 'PRP'),
            ('can', 'MD'),
            ('say', 'VB'),
            ('that', 'DT'),
            ('again', 'RB')]]

In [68]: sentlist = "You can say that again".split()
for tree in rd_parser.parse(sentlist) :
    print (tree)

(S (NP (Prop You)) (VP (V can) (VP (V say) (Det that) (ADV again))))

```

(d) “Birds are not necessarily able to fly”

```

In [16]: text4 = 'Birds are not necessarily able to fly'

In [17]: textsplit4 = nltk.sent_tokenize(text4)
textsplit4

Out[17]: ['Birds are not necessarily able to fly']

In [18]: tokentext4 = [nltk.word_tokenize(sent) for sent in textsplit4]
tokentext4

Out[18]: [['Birds', 'are', 'not', 'necessarily', 'able', 'to', 'fly']]

In [19]: taggedtext4 = [nltk.pos_tag(tokens) for tokens in tokentext4]
taggedtext4

Out[19]: [[('Birds', 'NNS'),
            ('are', 'VBP'),
            ('not', 'RB'),
            ('necessarily', 'RB'),
            ('able', 'JJ'),
            ('to', 'TO'),
            ('fly', 'VB')]]

In [69]: sentlist = "Birds are not necessarily able to fly".split()
for tree in rd_parser.parse(sentlist) :
    print (tree)

(S
  (NP (Prop Birds))
  (VP
    (V are)
    (AP
      (ADV not)
      (AP (ADV necessarily) (AP (ADJ able) (VP (Det to) (V fly)))))))

```

Part 2

a. "You can fly to France Today"

```
In [75]: sentlist = "You can fly to France Today".split()
for tree in rd_parser.parse(sentlist) :
    print (tree)

(S
 (NP (Prop You))
 (VP
  (V can)
  (VP (V fly) (PP (Det to) (NP (Prop France))) (NP (Prop Today))))))
```

b. "Bob went to France Today"

```
In [77]: sentlist = "Bob went to France Today".split()
for tree in rd_parser.parse(sentlist) :
    print (tree)

(S
 (NP (Prop Bob))
 (VP (V went) (PP (Det to) (NP (Prop France))) (NP (Prop Today))))
```

c. "Bob is not able to say"

```
In [78]: sentlist = "Bob is not able to say".split()
for tree in rd_parser.parse(sentlist) :
    print (tree)

(S
 (NP (Prop Bob))
 (VP (V is) (AP (ADV not) (AP (ADJ able) (VP (Det to) (V say))))))
```

Part 3

```
In [81]: grammar2 = nltk.PCFG.fromstring("""
S --> NP VP [1.0]
NP --> Prop[0.4] | Det N[0.3] | Det ADJ N[0.3]
VP --> V PP NP [0.16] | V NP [0.20] | V VP [0.16] | V Det ADV [0.16] | V AP [0.16] | Det V [0.16]
AP --> ADV AP[0.5] | ADJ VP [0.5]
PP --> Det NP [1.0]
ADV --> 'again' [0.3] | 'necessarily' [0.3] | 'not' [0.4]
ADJ --> 'nice' [0.5] | 'able' [0.5]
Det --> 'to' [0.25] | 'last' [0.25] | 'a' [0.25] | 'that' [0.25]
N --> 'month' [0.5] | 'day' [0.5]
V --> 'went' [0.16] | 'is' [0.16] | 'can' [0.16] | 'say' [0.16] | 'fly' [0.20] | 'are' [0.16]
Prop --> 'Bob' [0.4] | 'France' [0.18] | 'Today' [0.18] | 'You' [0.15] | 'Birds' [0.09]
""")

In [90]: vb = nltk.ViterbiParser(grammar2)
```

(a) "Today is a nice day"

```
In [91]: sentlist = "Today is a nice day".split()
for tree in vb.parse(sentlist) :
    print (tree)

(S
 (NP (Prop Today))
 (VP (V is) (NP (Det a) (ADJ nice) (N day)))) (p=4.32e-05)
```

(b) "Bob and Mary went to France last month again"

```
In [52]: sentlistp = 'Bob and Mary went to France last month again'.split()
for tree in vb.parse(sentlistp):
    print (tree)

(S
 (NP (Prop Bob) (CC and) (Prop Mary))
 (VP
  (V went)
  (NP
   (PP
    (P to)
    (NP (Prop France) (JJ last) (N month))
    (RB again)))) (p=3.13632e-09)
```

“You can say that again”

```
In [93]: sentlist = "You can say that again".split()
for tree in vb.parse(sentlist) :
    print (tree)

(S
 (NP (Prop You))
 (VP (V can) (VP (V say) (Det that) (ADV again)))) (p=2.94912e-06)
```

(d) “Birds are not necessarily able to fly”

```
In [94]: sentlist = "Birds are not necessarily able to fly".split()
for tree in vb.parse(sentlist) :
    print (tree)

(S
 (NP (Prop Birds))
 (VP
  (V are)
  (AP
   (ADV not)
   (AP
    (ADV necessarily)
    (AP (ADJ able) (VP (Det to) (V fly)))))) (p=5.5296e-08)
```

"You can fly to France Today"

```
In [95]: sentlist = "You can fly to France Today".split()
for tree in vb.parse(sentlist) :
    print (tree)

(S
 (NP (Prop You))
 (VP
  (V can)
  (VP
   (V fly)
   (PP (Det to) (NP (Prop France)))
   (NP (Prop Today)))) (p=6.3701e-08)
```

"Bob went to France Today"

```
In [96]: sentlist = "Bob went to France Today".split()
for tree in vb.parse(sentlist) :
    print (tree)

(S
 (NP (Prop Bob))
 (VP
  (V went)
  (PP (Det to) (NP (Prop France)))
  (NP (Prop Today)))) (p=5.30842e-06)
```

"Bob is not able to say"

```
In [97]: sentlist = "Bob is not able to say".split()
         for tree in vb.parse(sentlist) :
             print (tree)

(S
 (NP (Prop Bob))
 (VP
  (V is)
  (AP (ADV not) (AP (ADJ able) (VP (Det to) (V say)))))) (p=1.31072e-06)
```