IST 664 Natural Language Processing

Homework 3

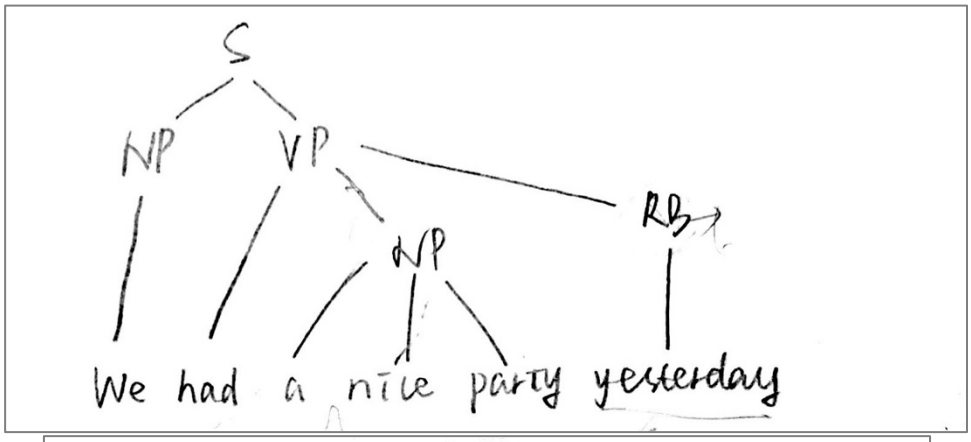
Harper He

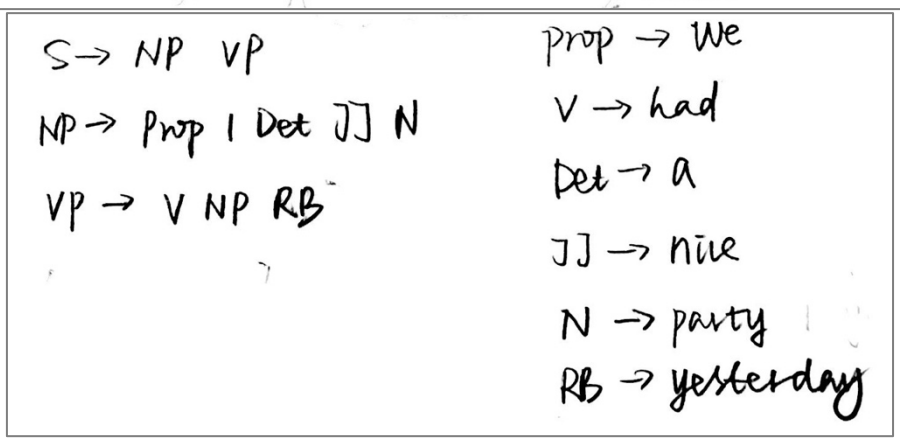
xhe128@syr.edu

Analysis of Sentences

To develop the context-free grammar that can parse all the sentences, I analyzed these sentence manually as well as using the online parse tools like Stanford Parser (http://nlp.stanford.edu:8080/parser/index.jsp). There were some differences between my parsing and the online tool's parsing, and I would explain the differences and use my understanding to parse these sentences.

(a). "We had a nice party yesterday"





Stanford Parser

Tagging

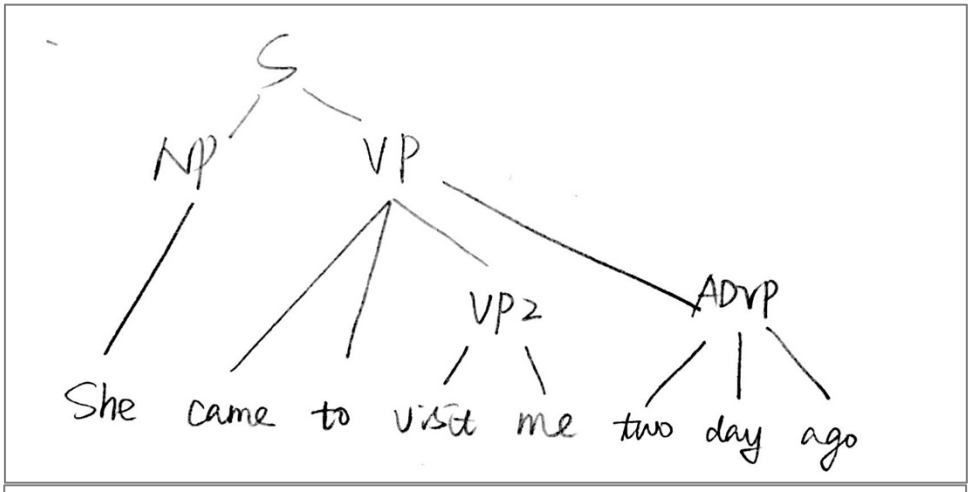
```
We/PRP  had/VBD  a/DT  nice/JJ  party/NN  yesterday/NN
```

Parse

```
(ROOT
  (S
    (NP (PRP We))
    (VP (VBD had)
      (NP (DT a)
        (ADJP (JJ nice))
        (NN party))
      (NP (NN yesterday)))))
```

As we can see, the Stanford Parser treats "yesterday" as a noun while I think it is an adverb. Because this word is used to modify the verb "had" to describe that the movement "had a party" happened on certain day, yesterday. Yesterday is regarded as a noun in sentences like "Yesterday was rainy and cold all day." or "Nobody's interested in yesterday's pop stars."

(b). "She came to visit me two days ago"



S → NP VP

NP → Prop

VP → InV To VP2

VP2 → V NP

ADVP → CD N RB

Prop → She I me

InV → came

To → to

V → visit

CD → two

N → days

RB → ago

Stanford Parser

Tagging

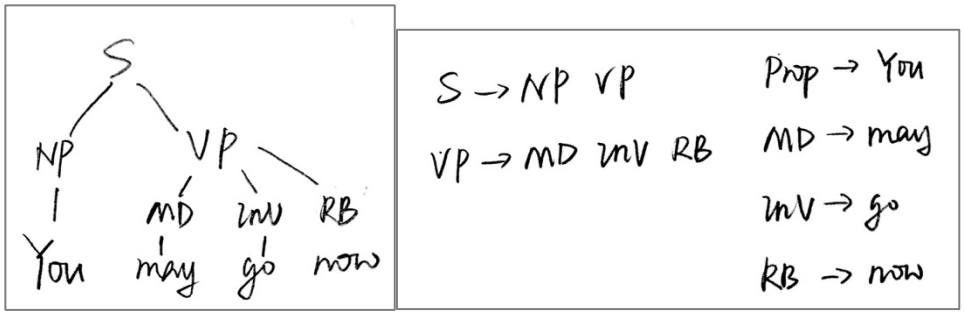
```
She/PRP  came/VBD  to/TO  visit/VB  me/PRP  two/CD  days/NNS  ago/RB
```

Parse

```
(ROOT
  (S
    (NP (PRP She))
    (VP (VBD came)
      (S
        (VP (TO to)
          (VP (VB visit)
            (NP (PRP me))
            (ADVP
              (NP (CD two) (NNS days))
              (RB ago)))))))))
```

As we can see, the Stanford Parser takes this sentence as two sentences while I think it is one sentence. Since word "came" is an intransitive verb, it is followed by "to". In my point of view, "come to do something" should be regarded as one verb phrase instead of two separate sentences "came" and "to do something". "Two days ago" is an adverb phrase to modify the verb "came".

(c). "You may go now"



Stanford Parser

Tagging

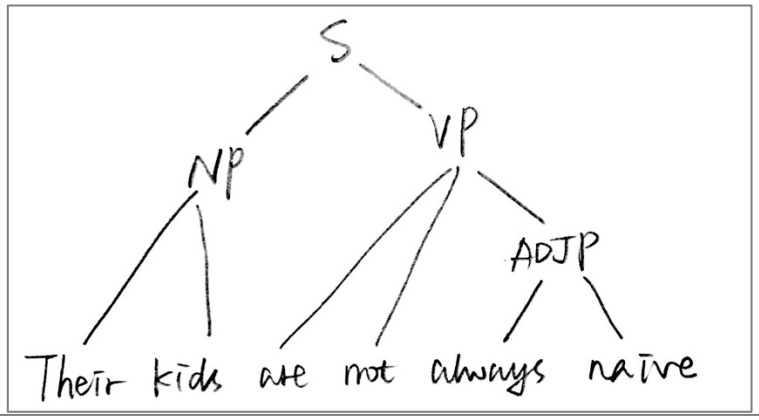
```
You/PRP  may/MD  go/VB  now/RB
```

Parse

```
(ROOT
  (S
    (NP (PRP You))
    (VP (MD may)
      (VP (VB go)
        (ADVP (RB now))))))
```

This sentence is relatively simple, its structure is "subject + modal verb + intransitive verb + adverb". I take the "modal verb + intransitive verb + adverb" as a verb phrase because in this case, "modal verb" and "intransitive verb" can be regarded as one verd phrase as the only predicate of this sentence.

(d). "Their kids are not always naive"



S → NP VP

NP → Det N

VP → V RB ADJP

ADJP → RB JJ

Det → Their

N → kids

V → are

RB → not | always

JJ → naive

Stanford Parser

Tagging

```
Their/PRP$  kids/NNS  are/VBP  not/RB  always/RB  naive/JJ
```

Parse

```
(ROOT
  (S
    (NP (PRP$ Their) (NNS kids))
    (VP (VBP are) (RB not)
      (ADVP (RB always))
      (ADJP (JJ naive)))))
```

In this sentence, "always" is an adverb to modify the adjective "naïve", so they comprise one adjective phrase "always naïve". The subject of this sentence is "their kids". And the predicate of this sentence is "are not".

Context-free Grammar

Based on my understanding and parsing of these sentences, I developed the following context-free grammar.

```
grammar = nltk.CFG.fromstring("""
  S -> NP VP
  VP -> V NP RB | InV TO VP2 ADVP| MD InV RB | V RB ADJP
  VP2 -> V NP
  NP -> Prop |Det JJ N | Det N
  ADJP -> RB JJ
  ADVP -> CD N RB
  V -> "had" | "visit" |  "are"
  InV ->  "came" |"go"
  Prop -> "We" |"You" | "She" | "me"
  Det -> "a" | "Their"
  N -> "party" | "days" | "kids"
  TO -> "to"
  JJ -> "nice" | "naive"
  CD -> "two"
  RB -> "ago" | "now" | "not" | "always" | "yesterday"
  MD -> "may"
""")
```

All the sentences can be parsed using the context-free grammar.

```
sent1list = "We had a nice party yesterday".split()
for tree in rd_parser.parse(sent1list):
    print (tree)
```

```
(S
  (NP (Prop We))
  (VP (V had) (NP (Det a) (JJ nice) (N party)) (RB yesterda
y)))
```

```
sent2list = "She came to visit me two days ago".split()
for tree in rd_parser.parse(sent2list):
    print (tree)
```

```
(S
  (NP (Prop She))
  (VP
    (InV came)
    (TO to)
    (VP2 (V visit) (NP (Prop me)))
    (ADVP (CD two) (N days) (RB ago))))
```

```
sent3list = "You may go now".split()
for tree in rd_parser.parse(sent3list):
    print (tree)
```

```
(S (NP (Prop You)) (VP (MD may) (InV go) (RB now)))
```

```
sent4list = "Their kids are not always naive".split()
for tree in rd_parser.parse(sent4list):
    print (tree)
```

```
(S
  (NP (Det Their) (N kids))
  (VP (V are) (RB not) (ADJP (RB always) (JJ naive))))
```

Below are the three sentences I generated that can be parsed by this grammar. As we can see, one is meaningful while the other two do not make sense.

```
sent5list = "You are not always nice".split()
for tree in rd_parser.parse(sent5list):
    print (tree)
```

```
(S
  (NP (Prop You))
  (VP (V are) (RB not) (ADJP (RB always) (JJ nice))))
```

```
sent6list = "Their party had me ago".split()
for tree in rd_parser.parse(sent6list):
    print (tree)
```

```
(S (NP (Det Their) (N party)) (VP (V had) (NP (Prop me)) (RB ago)))
```

```
sent7list = "a kids visit not now naive".split()
for tree in rd_parser.parse(sent7list):
    print (tree)
```

```
(S
  (NP (Det a) (N kids))
  (VP (V visit) (RB not) (ADJP (RB now) (JJ naive))))
```

I can think of two possible explanation for this situation. First, the context-free grammar I developed is not strict enough. Since I developed it based on the given sentences and my interpretation of these sentences, chances are that the grammar is not strict enough to make sure the new sentences are not only organized correctly but also make sense. Second, it is due to the limitation of grammar itself. As we know, the function of grammar is to organize sentences. But words can be combined and arranged in many different ways to form sentences. Grammar may restrict the property and sequence of the words, but this kind of restriction is not enough to make sure their combination and sequence are meaningful.

Probabilistic context-free Grammar

Assuming that the above given four sentences were the mini-mini training corpus, I developed a probabilistic context-free grammar as following.

```
# Probabilistic CFG with verb subcategories
prob_grammar = nltk.PCFG.fromstring("""
  S -> NP VP [1.0]
  VP -> TranV NP RB [0.25] | InV TO VP2 ADVP[0.25]  | MD InV RB[0.25]  | TranV RB ADJP[0.25]
  VP2 -> TranV NP [1.0]
  NP -> Prop [0.666] |Det JJ N[0.166] | Det N [0.166]
  ADJP -> RB JJ[1.0]
  ADVP -> CD N RB [1.0]
  InV -> "came" [0.5] | "go" [0.5]
  TranV -> "had"[0.333] | "visit" [0.333] | "are"[0.333]
  Prop -> "We"[0.25]  |"You"[0.25]  | "She"[0.25]  | "me" [0.25]
  Det -> "a"[0.5]  | "Their" [0.5]
  N -> "party"[0.333]  | "days" [0.333] | "kids"[0.333]
  TO -> "to" [1.0]
  JJ -> "nice"[0.5]  | "naive"[0.5]
  CD -> "two"[1.0]
  RB -> "ago"[0.2]  | "now"[0.2]  | "not" [0.2]  | "always"[0.2]  |"yesterday" [0.2]
  MD -> "may"[1.0]
""")
```

In the process of developing the probabilistic context-free grammar, I noticed that two constituents, "N" and "TranV" have 3 elements. Since productions for constituents have to sum to 1 but 1 is indivisible by 3. However, NLTK may have default setting regarding how to deal with decimal to round up to 1, so as long as I set their probability as 0.333, there would not be error report like "ValueError: Productions for N/TranV do not sum to 1". My previous method before Professor Zhang informed me this trick was to add 0.01 to the last word. I think this manipulation may cause my grammar a little biased since some words had a relatively higher probability.

In the category of "NP", as we can see, it contains Prop, Det JJ N and Det N. However, they are not of the same frequency, which means they are not of the same probability to appear. There are 6 NP in these 4 sentences and 4 of them are Prop while the remaining two are Det JJ N and Det N, so the probability of Prop is 4/6≈0.666 and the probability of Det JJ N and Det N is 1/6≈0.166 individually.

All the sentences can be parsed using the context-free grammar.

```
# use its parse function on a list of tokens
for tree in viterbi_parser.parse(sent1list):
    print (tree)
```

```
(S
  (NP (Prop We))
  (VP
    (TranV had)
    (NP (Det a) (JJ nice) (N party))
    (RB yesterday))) (p=3.83108e-05)
```

```
for tree in viterbi_parser.parse(sent2list):
    print (tree)
```

```
(S
  (NP (Prop She))
  (VP
    (InV came)
    (TO to)
    (VP2 (TranV visit) (NP (Prop me)))
    (ADVP (CD two) (N days) (RB ago)))) (p=7.68523e-05)
```

```
for tree in viterbi_parser.parse(sent3list):
    print (tree)
```

```
(S (NP (Prop You)) (VP (MD may) (InV go) (RB now))) (p=0.0041625)
```

```
for tree in viterbi_parser.parse(sent4list):
    print (tree)
```

```
(S
  (NP (Det Their) (N kids))
  (VP (TranV are) (RB not) (ADJP (RB always) (JJ naive)))) (p=4.60189e-05)
```

Appendix: Python code & output

In [1]:

```
# IST 664 Natural Language Processing
# Homework 3
# Harper He
# xhe128
```

In [1]:

```
import nltk
```

In [2]:

```
# develop the context-free grammar that can parse all the following sentences
```

In [9]:

```
grammar = nltk.CFG.fromstring("""
  S -> NP VP
  VP -> V NP RB | InV TO VP2 ADVP| MD InV RB | V RB ADJP
  VP2 -> V NP
  NP -> Prop |Det JJ N | Det N
  ADJP -> RB JJ
  ADVP -> CD N RB
  V -> "had" | "visit" |  "are"
  InV ->  "came" |"go"
  Prop -> "We" |"You" | "She" | "me"
  Det -> "a" | "Their"
  N -> "party" | "days" | "kids"
  TO -> "to"
  JJ -> "nice" | "naive"
  CD -> "two"
  RB -> "ago" | "now" | "not" | "always" | "yesterday"
  MD -> "may"
""")
```

In [10]:

```
rd_parser = nltk.RecursiveDescentParser(grammar)
```

In [11]:

```
sent1list = "We had a nice party yesterday".split()
for tree in rd_parser.parse(sent1list):
 print (tree)
```

```
(S
  (NP (Prop We))
  (VP (V had) (NP (Det a) (JJ nice) (N party)) (RB yesterday)))
```

In [12]:

```
sent2list = "She came to visit me two days ago".split()
for tree in rd_parser.parse(sent2list):
 print (tree)
```

```
(S
  (NP (Prop She))
  (VP
    (InV came)
    (TO to)
    (VP2 (V visit) (NP (Prop me)))
    (ADVP (CD two) (N days) (RB ago))))
```

In [13]:

```
sent3list = "You may go now".split()
for tree in rd_parser.parse(sent3list):
 print (tree)
```

```
(S (NP (Prop You)) (VP (MD may) (InV go) (RB now)))
```

In [14]:

```
sent4list = "Their kids are not always naive".split()
for tree in rd_parser.parse(sent4list):
 print (tree)
```

```
(S
  (NP (Det Their) (N kids))
  (VP (V are) (RB not) (ADJP (RB always) (JJ naive))))
```

In [10]:

```
# list three more different sentences that can be parsed by this grammar
# can you generate one that does not make sense?
# Why can't you have such a sentence based on your grammar?
```

In [21]:

```
sent5list = "You are not always nice".split()
for tree in rd_parser.parse(sent5list):
 print (tree)
```

```
(S
  (NP (Prop You))
  (VP (V are) (RB not) (ADJP (RB always) (JJ nice))))
```

In [22]:

```
sent6list = "Their party had me ago".split()
for tree in rd_parser.parse(sent6list):
 print (tree)
```

```
(S (NP (Det Their) (N party)) (VP (V had) (NP (Prop me)) (RB ago)))
```

In [26]:

```
sent7list = "a kids visit not now naive".split()
for tree in rd_parser.parse(sent7list):
 print (tree)
```

```
(S
  (NP (Det a) (N kids))
  (VP (V visit) (RB not) (ADJP (RB now) (JJ naive))))
```

In [14]:

```
# assuming that the above given four sentences are your mini-mini training corpus,
# you will write a probabilistic context-free grammar.
```

In [27]:

```
# Probabilistic CFG with verb subcategories
prob_grammar = nltk.PCFG.fromstring("""
  S -> NP VP [1.0]
  VP -> TranV NP RB [0.25] | InV TO VP2 ADVP[0.25]  | MD InV RB[0.25]  | TranV RB ADJP[0.25]
  VP2 -> TranV NP [1.0]
  NP -> Prop [0.666] |Det JJ N[0.166] | Det N [0.166]
  ADJP -> RB JJ[1.0]
  ADVP -> CD N RB [1.0]
  InV -> "came" [0.5] | "go" [0.5]
  TranV -> "had"[0.333] | "visit" [0.333] | "are"[0.333]
```

```
  Prop -> "We"[0.25]  |"You"[0.25]  | "She"[0.25]  | "me" [0.25]
  Det -> "a"[0.5]  | "Their" [0.5]
  N -> "party"[0.333]  | "days" [0.333] | "kids"[0.333]
  TO -> "to" [1.0]
  JJ -> "nice"[0.5]  | "naive"[0.5]
  CD -> "two"[1.0]
  RB -> "ago"[0.2]  | "now"[0.2]  | "not" [0.2]  | "always"[0.2]  |"yesterday" [0.2]
  MD -> "may"[1.0]
""")
```

In [28]:

```
# create a viterbi parser, a parser that expects a PCFG
viterbi_parser = nltk.ViterbiParser(prob_grammar)
```

In [29]:

```
# use its parse function on a list of tokens
for tree in viterbi_parser.parse(sent1list):
    print (tree)
```

```
(S
  (NP (Prop We))
  (VP
    (TranV had)
    (NP (Det a) (JJ nice) (N party))
    (RB yesterday))) (p=3.83108e-05)
```

In [30]:

```
for tree in viterbi_parser.parse(sent2list):
    print (tree)
```

```
(S
  (NP (Prop She))
  (VP
    (InV came)
    (TO to)
    (VP2 (TranV visit) (NP (Prop me)))
    (ADVP (CD two) (N days) (RB ago)))) (p=7.68523e-05)
```

In [31]:

```
for tree in viterbi_parser.parse(sent3list):
    print (tree)
```

```
(S (NP (Prop You)) (VP (MD may) (InV go) (RB now))) (p=0.0041625)
```

In [32]:

```
for tree in viterbi_parser.parse(sent4list):
    print (tree)
```

```
(S
  (NP (Det Their) (N kids))
  (VP (TranV are) (RB not) (ADJP (RB always) (JJ naive)))) (p=4.60189e-05)
```