Lab 10. Named Entity Recognition

Joshua E (225229117) ¶

Exercise-1

In [1]:
```
sentence1="Rajkumar said on Monday that WASHINGTON -- In the wake of a string
```

In [2]:
```
sentence1
```

Out[2]:
```
'Rajkumar said on Monday that WASHINGTON -- In the wake of a string of abuses
by New York police officers in the 1990s, Loretta E. Lynch, the top federal p
rosecutor in Brooklyn, spoke forcefully about the pain of abroken trust that
Africa-Amercans felt and said responsibility for repairing generations miscom
munication and mistrust fell to law enforcement'
```

In [3]:
```
import nltk
nltk.download('maxent_ne_chunker')
```

```
[nltk_data] Error loading maxent_ne_chunker: <urlopen error [WinError
[nltk_data]     10060] A connection attempt failed because the
[nltk_data]     connected party did not properly respond after a
[nltk_data]     period of time, or established connection failed
[nltk_data]     because connected host has failed to respond>
```

Out[3]:
```
False
```

In [4]:
```
import nltk
nltk.download('words')
```

```
[nltk_data] Error loading words: <urlopen error [WinError 10060] A
[nltk_data]     connection attempt failed because the connected party
[nltk_data]     did not properly respond after a period of time, or
[nltk_data]     established connection failed because connected host
[nltk_data]     has failed to respond>
```

Out[4]:
```
False
```

In [5]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk

tokens = word_tokenize(sentence1)
tags = pos_tag(tokens)
ne_trees = ne_chunk(tags)
print(ne_trees)

ne_trees=ne_chunk(pos_tag(word_tokenize(sentence1)))
```

```
(S
  (PERSON Rajkumar/NNP)
  said/VBD
  on/IN
  Monday/NNP
  that/IN
  (ORGANIZATION WASHINGTON/NNP)
  --/:
  In/IN
  the/DT
  wake/NN
  of/IN
  a/DT
  string/NN
  of/IN
  abuses/NNS
  by/IN
  (GPE New/NNP York/NNP)
  police/NN
  officers/NNS
  in/IN
  the/DT
  1990s/CD
  ,/,
  (PERSON Loretta/NNP E./NNP Lynch/NNP)
  ,/,
  the/DT
  top/JJ
  federal/JJ
  prosecutor/NN
  in/IN
  (GPE Brooklyn/NNP)
  ,/,
  spoke/VBD
  forcefully/RB
  about/IN
  the/DT
  pain/NN
  of/IN
  abroken/JJ
  trust/NN
  that/IN
  Africa-Amercans/NNP
  felt/VBD
  and/CC
  said/VBD
  responsibility/NN
  for/IN
  repairing/VBG
  generations/NNS
  miscommunication/NN
  and/CC
  mistrust/NN
  fell/VBD
  to/TO
```

```
law/NN
enforcement/NN)
```

Question 1

In [17]:
```
import nltk
from collections import Counter
for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sentence1))):
    if hasattr(chunk, 'label'):
        print([Counter(label) for label in chunk])
```

```
[Counter({'Rajkumar': 1, 'NNP': 1})]
[Counter({'WASHINGTON': 1, 'NNP': 1})]
[Counter({'New': 1, 'NNP': 1}), Counter({'York': 1, 'NNP': 1})]
[Counter({'Loretta': 1, 'NNP': 1}), Counter({'E.': 1, 'NNP': 1}), Counter({'L
ynch': 1, 'NNP': 1})]
[Counter({'Brooklyn': 1, 'NNP': 1})]
```

Question 2

In [18]:
```
word = nltk.word_tokenize(sentence1)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<NN><NNS>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.T
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'New York', 'police officers', 'Loretta E. Lynch',
'Brooklyn']
```

Question 3

In [19]:
```
grammar = "NP: {<DT><JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.T
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'the wake', 'a string', 'New York', 'Loretta E. Ly
nch', 'the top federal prosecutor', 'Brooklyn', 'the pain']
```

In [20]:
```
parse = cp.parse(tags)
print(parse[:])
```

```
[('Rajkumar', 'NNP'), ('said', 'VBD'), ('on', 'IN'), ('Monday', 'NNP'), ('tha
t', 'IN'), ('WASHINGTON', 'NNP'), ('--', ':'), ('In', 'IN'), Tree('NP', [('th
e', 'DT'), ('wake', 'NN')]), ('of', 'IN'), Tree('NP', [('a', 'DT'), ('strin
g', 'NN')]), ('of', 'IN'), ('abuses', 'NNS'), ('by', 'IN'), ('New', 'NNP'),
('York', 'NNP'), ('police', 'NN'), ('officers', 'NNS'), ('in', 'IN'), ('the',
'DT'), ('1990s', 'CD'), (',', ','), ('Loretta', 'NNP'), ('E.', 'NNP'), ('Lync
h', 'NNP'), (',', ','), Tree('NP', [('the', 'DT'), ('top', 'JJ'), ('federal',
'JJ'), ('prosecutor', 'NN')]), ('in', 'IN'), ('Brooklyn', 'NNP'), (',', ','),
('spoke', 'VBD'), ('forcefully', 'RB'), ('about', 'IN'), Tree('NP', [('the',
'DT'), ('pain', 'NN')]), ('of', 'IN'), ('abroken', 'JJ'), ('trust', 'NN'),
('that', 'IN'), ('Africa-Amercans', 'NNP'), ('felt', 'VBD'), ('and', 'CC'),
('said', 'VBD'), ('responsibility', 'NN'), ('for', 'IN'), ('repairing', 'VB
G'), ('generations', 'NNS'), ('miscommunication', 'NN'), ('and', 'CC'), ('mis
trust', 'NN'), ('fell', 'VBD'), ('to', 'TO'), ('law', 'NN'), ('enforcement',
'NN')]
```

In [21]:
```
grammar = "NP: {<DT><JACJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.T
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'the wake', 'a string', 'New York', 'Loretta E. Ly
nch', 'Brooklyn', 'the pain']
```

Exercise-2

In [22]:
```
sentence2="European authrities fined google a record $5.1 billion on Wednesday
```

In [23]:
```
token=word_tokenize(sentence2)
tag=nltk.pos_tag(token)
ne_tree=ne_chunk(tag)
print(ne_tree[:])
```

```
[Tree('GPE', [('European', 'JJ')]), ('authrities', 'NNS'), ('fined', 'VBN'),
('google', 'VBP'), ('a', 'DT'), ('record', 'NN'), ('$', '$'), ('5.1', 'CD'),
('billion', 'CD'), ('on', 'IN'), ('Wednesday', 'NNP'), ('for', 'IN'), ('abusi
ng', 'VBG'), ('its', 'PRP$'), ('power', 'NN'), ('in', 'IN'), ('the', 'DT'),
('mobile', 'JJ'), ('phone', 'NN'), ('market', 'NN'), ('and', 'CC'), ('ordere
d', 'VBD'), ('the', 'DT'), ('company', 'NN'), ('to', 'TO'), ('alter', 'VB'),
('its', 'PRP$'), ('parctics', 'NNS')]
```

Question 1

In [24]:
```
word = nltk.word_tokenize(sentence2)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<CD>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.T
print (NE)
```

```
['European', '5.1', 'billion']
```

Question 2

In [25]:
```
word = nltk.word_tokenize(sentence2)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<DT><JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.T
print (NE)
```

```
['European', 'a record', 'the mobile phone', 'the company']
```

In [35]:
```
f=open("fr.txt",'r')
file=f.read()
file=str(file)
print(file)
```

```
1 1/2 cups dry red wine
3 cloves garlic
1 3/4 cups beef broth
1 1/4 cups chicken broth
1 1/2 tablespoons tomato paste
1 bay leaf
1 sprig thyme
8 ounces bacon cut into 1/4 inch pieces
1 tablespoon flour
1 tablespoon butter
4 1 inch rib-eye steaks
1 tablespoon bourbon whiskey
```

In [60]:
```
tokens =nltk.word_tokenize(file)
tags =nltk.pos_tag(tokens)
ne_trees = ne_chunk(tags)
print(ne_trees)
```

```
(S
  1/CD
  1/2/CD
  cups/NNS
  dry/JJ
  red/JJ
  wine/NN
  3/CD
  cloves/NNS
  garlic/JJ
  1/CD
  3/4/CD
  cups/NNS
  beef/VBD
  broth/DT
  1/CD
  1/4/CD
  cups/NNS
  chicken/VBP
  broth/DT
  1/CD
  1/2/CD
  tablespoons/NNS
  tomato/VBP
  paste/NN
  1/CD
  bay/NN
  leaf/NN
  1/CD
  sprig/NN
  thyme/NN
  8/CD
  ounces/NNS
  bacon/JJ
  cut/VBD
  into/IN
  1/4/CD
  inch/NN
  pieces/NNS
  1/CD
  tablespoon/RB
  flour/JJ
  1/CD
  tablespoon/NN
  butter/NN
  4/CD
  1/CD
  inch/JJ
  rib-eye/JJ
  steaks/NNS
  1/CD
  tablespoon/NN
  bourbon/NN
  whiskey/NN)
```

In [137]:
```
import nltk

# Define the regular expression pattern
pattern = 'FOOD: {(<JJ>* <NN.*>+ <IN>)? (<JJ>* <NN.*>+)}'

# Tokenize and part-of-speech tag the text
text = """1 1/2 cups dry red wine
3 cloves garlic
1 3/4 cups beef broth
1 1/4 cups chicken broth
1 1/2 tablespoons tomato paste
1 bay leaf
1 sprig thyme
8 ounces bacon cut into 1/4 inch pieces
1 tablespoon flour
1 tablespoon butter
4 1 inch rib-eye steaks
1 tablespoon bourbon whiskey"""
tokens = nltk.word_tokenize(text)
pos_tags = nltk.pos_tag(tokens)

# Apply the regular expression pattern to the POS tagged tokens
cp = nltk.RegexpParser(pattern)
tree = cp.parse(pos_tags)

# Extract the food recipes from the tree
food_recipes = []
for subtree in tree.subtrees():
    if subtree.label() == 'FOOD':
        recipe = ' '.join(word for word, tag in subtree.leaves())
        food_recipes.append(recipe)

# Print the extracted food recipes
print(food_recipes)
```

```
['cups', 'dry red wine', 'cloves', 'cups', 'cups', 'tablespoons', 'paste', 'b
ay leaf', 'sprig thyme', 'ounces', 'inch pieces', 'tablespoon butter', 'inch
rib-eye steaks', 'tablespoon bourbon whiskey']
```

In [ ]:

In [ ]: