Lab10.Named Entity Recognition

EXERCISE-1

In [43]:
```
rican-Americans felt and said the responsibility for repairing generations of micommunication and mitrust fell to law enforcement"
```

In [44]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
```

In [45]:
```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\1mscdsa13\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\1mscdsa13\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\1mscdsa13\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\1mscdsa13\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Out[45]: True

In [46]:
```
tokens=word_tokenize(Sentence1)
tags=pos_tag(tokens)
ne_tree=ne_chunk(tags)
print(ne_tree)
```

```
(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
  a/DT
  broken/JJ
  trust/NN
  that/IN
  African-Americans/NNP
  felt/VBD
  and/CC
  said/VBD
  the/DT
  responsibility/NN
  for/IN
  repairing/VBG
  generations/NNS
  of/IN
  micommunication/NN
  and/CC
  mitrust/NN
  fell/VBD
  to/TO
  law/NN
  enforcement/NN)
```

In [47]:
```
ne_tree=ne_chunk(pos_tag(word_tokenize(Sentence1)))
```

In [48]:
```
for i in ne_tree:
    print(i)
```

```
(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')
('a', 'DT')
('broken', 'JJ')
('trust', 'NN')
('that', 'IN')
('African-Americans', 'NNP')
('felt', 'VBD')
('and', 'CC')
('said', 'VBD')
('the', 'DT')
('responsibility', 'NN')
('for', 'IN')
('repairing', 'VBG')
('generations', 'NNS')
('of', 'IN')
('micommunication', 'NN')
('and', 'CC')
('mitrust', 'NN')
('fell', 'VBD')
('to', 'TO')
('law', 'NN')
('enforcement', 'NN')
```

Question - 1

Count and print the number of PERSON,LOCATION and ORGANIZATION in the given sentence

In [49]:
```
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({'Lynch': 1, 'NNP': 1})]
[Counter({'Brooklyn': 1, 'NNP': 1})]
```

Question-2

observe the results. Does named entitym "Police Officers" get recognized

In [50]:
```
my_sent = "Rajkumar said on Monday that WASHINGTON -- In the wake of a string of abuses by New York police officers in the 1990s,
word = nltk.word_tokenize(my_sent)
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.Tree)]
print (NE)
```

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

Write a regular expression patter to detect this.You will need nltk.RegexParser class to define pattern and parse terms to detect patterns

In [51]:
```
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.Tree)]
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'the wake', 'a string', 'New York', 'Loretta E. Lynch', 'the top federal prosecutor', 'Brooklyn', 't
he pain', 'a broken trust', 'the responsibility']
```

Question-3

Does the named entity,"the top federal prosecutor" get recognized?

In [52]:
```
parse = cp.parse(tags)
print(parse[:])
```

```
[('Rajkumar', 'NNP'), ('said', 'VBD'), ('on', 'IN'), ('Monday', 'NNP'), ('that', 'IN'), ('WASHINGTON', 'NNP'), ('--', ':'), ('I
n', 'IN'), Tree('NP', [('the', 'DT'), ('wake', 'NN')]), ('of', 'IN'), Tree('NP', [('a', 'DT'), ('string', 'NN')]), ('of', 'I
N'), ('abuses', 'NNS'), ('by', 'IN'), ('New', 'NNP'), ('York', 'NNP'), ('police', 'NN'), ('officers', 'NNS'), ('in', 'IN'), ('t
he', 'DT'), ('1990s', 'CD'), (',', ','), ('Loretta', 'NNP'), ('E.', 'NNP'), ('Lynch', 'NNP'), (',', ','), Tree('NP', [('the',
'DT'), ('top', 'JJ'), ('federal', 'JJ'), ('prosecutor', 'NN')]), ('in', 'IN'), ('Brooklyn', 'NNP'), (',', ','), ('spoke', 'VB
D'), ('forcefully', 'RB'), ('about', 'IN'), Tree('NP', [('the', 'DT'), ('pain', 'NN')]), ('of', 'IN'), Tree('NP', [('a', 'DT'),
('broken', 'JJ'), ('trust', 'NN')]), ('that', 'IN'), ('African-Americans', 'NNP'), ('felt', 'VBD'), ('and', 'CC'), ('said', 'VB
D'), Tree('NP', [('the', 'DT'), ('responsibility', 'NN')]), ('for', 'IN'), ('repairing', 'VBG'), ('generations', 'NNS'), ('of',
'IN'), ('micommunication', 'NN'), ('and', 'CC'), ('mitrust', 'NN'), ('fell', 'VBD'), ('to', 'TO'), ('law', 'NN'), ('enforcemen
t', 'NN')]
```

Write a regular expression pattern to detect this.

In [53]:
```
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.Tree)]
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'the wake', 'a string', 'New York', 'Loretta E. Lynch', 'Brooklyn', 'the pain', 'the responsibilit
y']
```

EXERCISE-2

Extract All Named Entities From The Following Text:

In [54]:
```
sentence2="European authorities fined Google a record $5.1 billion on wednesday for abusing its power in the mobile phone market a
```

Question-1

Observe the output.Does your code recognize the NE shown in BOLD?

In [55]:
```
token=word_tokenize(sentence2)
tag=nltk.pos_tag(token)
ne_tree=ne_chunk(tag)
print(ne_tree[:])
```

```
[Tree('GPE', [('European', 'JJ')]), ('authorities', 'NNS'), ('fined', 'VBD'), Tree('PERSON', [('Google', 'NNP')]), ('a', 'DT'),
('record', 'NN'), ('$', '$'), ('5.1', 'CD'), ('billion', 'CD'), ('on', 'IN'), ('wednesday', 'NN'), ('for', 'IN'), ('abusing',
'VBG'), ('its', 'PRP$'), ('power', 'NN'), ('in', 'IN'), ('the', 'DT'), ('mobile', 'JJ'), ('phone', 'NN'), ('market', 'NN'), ('a
nd', 'CC'), ('ordered', 'VBD'), ('the', 'DT'), ('company', 'NN'), ('to', 'TO'), ('alter', 'VB'), ('its', 'PRP$'), ('practices',
'NNS')]
```

Write a regular expression that recognizes the entity,"$5.1 billion" Detect and pritn this

In [56]:
```
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.Tree)]
print (NE)
```

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

Question-2

Write a regular expression that recognizes the entity,"the mobile phone" and similar to thius entity such as "the company"

In [57]:
```
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.Tree)]
print(NE)
```

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

In [ ]: