

NLP_lab-10_205229118_Mahalakshmi.S

May 25, 2021

0.0.1 Lab10.Named Entity Recognition

0.0.2 EXERCISE-1

0.0.3 Extract all named entities from the following text:

```
[1]: import nltk
      from nltk.tree import Tree
      from nltk.tokenize import word_tokenize
      from nltk.tag import pos_tag
      from nltk.chunk import ne_chunk
      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\Mahalakshmi\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\Mahalakshmi\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\Mahalakshmi\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\Mahalakshmi\AppData\Roaming\nltk_data...
[nltk_data] Package words is already up-to-date!
```

```
[1]: True
```

```
[2]: Sentence1 = "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 prosecutor in Brooklyn, spoke forcefully about the pain of a broken
      ↳trust that African-Americans felt and said the responsibility for repairing
      ↳generations of miscommunication and mistrust fell to law enforcement."
```

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

```
[Tree('PERSON', [('Rajkumar', 'NNP')]), ('said', 'VBD'), ('on', 'IN'),
('Monday', 'NNP'), ('that', 'IN'), Tree('ORGANIZATION', [('WASHINGTON',
'NNP')]), ('--', ':'), ('In', 'IN'), ('the', 'DT'), ('wake', 'NN'), ('of',
'IN'), ('a', 'DT'), ('string', 'NN'), ('of', 'IN'), ('abuses', 'NNS'), ('by',
'IN'), Tree('GPE', [('New', 'NNP'), ('York', 'NNP')]), ('police', 'NN'),
('officers', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('1990s', 'CD'), (',', ','),
Tree('PERSON', [('Loretta', 'NNP'), ('E.', 'NNP'), ('Lynch', 'NNP')]), (',',
'), ('the', 'DT'), ('top', 'JJ'), ('federal', 'JJ'), ('prosecutor', 'NN'),
('in', 'IN'), Tree('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'), ('miscommunication', 'NN'), ('and', 'CC'),
('mistrust', 'NN'), ('fell', 'VBD'), ('to', 'TO'), ('law', 'NN'),
('enforcement', 'NN'), ('.', '.')]

```

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

```
[5]: 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')
 ('miscommunication', 'NN')
 ('and', 'CC')
 ('mistrust', 'NN')
 ('fell', 'VBD')
 ('to', 'TO')
 ('law', 'NN')
 ('enforcement', 'NN')
 ('.', '.')

0.0.4 Question-1

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

```
[6]: import nltk
      from collections import Counter
      for chunk in ne_tree:
          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})]
```

0.0.6 Question 2

0.0.7 Observe the results. Does named entity, “police officers” get recognized?.

```
[13]: word = nltk.word_tokenize(Sentence1)
      pos_tag = nltk.pos_tag(word)
      chunk = nltk.ne_chunk(pos_tag)
      grammar = "NP: {<NN><NNS>}"
      nl = nltk.RegexpParser(grammar)
      result = nl.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']
```

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

```
[14]: grammar = "NP: {<NN><NNS>}"
      nl = nltk.RegexpParser(grammar)
      result = nl.parse(ne_tree)
      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']
```

0.0.9 Question-3

0.0.10 Does the named entity, “the top federal prosecutor” get recognized?.

```
[15]: top=nl.parse(tags)
      print(top[:])

[('Rajkumar', 'NNP'), ('said', 'VBD'), ('on', 'IN'), ('Monday', 'NNP'), ('that', 'IN'), ('WASHINGTON', 'NNP'), ('--', ':'), ('In', 'IN'), ('the', 'DT'), ('wake', 'NN'), ('of', 'IN'), ('a', 'DT'), ('string', 'NN'), ('of', 'IN'), ('abuses', 'NNS'), ('by', 'IN'), ('New', 'NNP'), ('York', 'NNP'), Tree('NP', [('police', 'NN'), ('officers', 'NNS')]), ('in', 'IN'), ('the', 'DT'), ('1990s', 'CD'), ('.', 'P'), ('Loretta', 'NNP'), ('E.', 'NNP'), ('Lynch', 'NNP'), ('.', 'P'), ('the', 'DT'), ('top', 'JJ'), ('federal', 'JJ'), ('prosecutor', 'NN'), ('in', 'IN'), ('Brooklyn', 'NNP'), ('.', 'P'), ('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'), ('miscommunication', 'NN'), ('and', 'CC'), ('mistrust', 'NN'), ('fell', 'VBD'), ('to', 'TO'), ('law', 'NN'), ('enforcement', 'NN'), ('.', 'P'), ('.', 'P')]
```

0.0.11 Write a regular expression pattern to detect this.

```
[16]: grammar = "NP: {<DT><JJ>*<NN>}"
      nl = nltk.RegexpParser(grammar)
      result = nl.parse(ne_tree)
      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', 'the pain', 'a broken trust', 'the responsibility']
```

0.0.12 EXERCISE-2

0.0.13 Question-1

0.0.14 Observe the output. Does your code recognize the NE shown in BOLD?(\$5.1 billion, the mobile phone, the company)

```
[11]: Sentence2 = "European authorities fined Google a record $5.1 billion on_
      ↳Wednesday for abusing its power in the mobile phone market and ordered the_
      ↳company to alter its practices"
```

```
[12]: toks = word_tokenize(Sentence2)
      tagged = nltk.pos_tag(toks)
      ne_tree_3 = nltk.ne_chunk(tagged,binary=False)
```

```
print(ne_tree_3[:])
```

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

0.0.15 Write a regular expression that recognizes the entity.

```
[17]: word = nltk.word_tokenize(Sentence2)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<CD>|<DT><JJ>*<NN>}"
nl = nltk.RegexpParser(grammar)
result = nl.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', '5.1', 'billion', 'the mobile phone', 'the
company']
```

0.0.16 Question-2

0.0.17 Write a regular expression that recognizes the entity, “the mobile phone” and similar to this entity such as “the company

```
[18]: word = nltk.word_tokenize(Sentence2)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<DT><JJ>*<NN>}"
nl = nltk.RegexpParser(grammar)
result = nl.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']
```