Santhosh.S

225229133

NATURAL LANGUAGE PRE-PROCESSING LAB

LAB_15. Text Processing using Spacy

In [1]:
```
import spacy
```

In [2]:
```
sci=spacy.load("en_core_web_sm")
```

Question 1. Print the tokens of the string

In [3]:
```
text=sci("welcome all of you for this NLP with spacy course")
```

In [4]:
```
for token in text:
    print(token.text)
```

```
welcome
all
of
you
for
this
NLP
with
spacy
course
```

Question2. Create a text file that contain the above string

In [5]:
```
with open("text2.txt","w")as text2:
    for k in text:
        text2.write(k.text)
        text2.write('\n')
text2.close()
```

```
In [6]: text2=open("text2.txt","r")
        r=text2.read()
        print(r)
```

```
welcome
all
of
you
for
this
NLP
with
spacy
course
```

Question 3. Consider the following sentences and print each sentence in one line

```
In [7]: my_text = ('Rajkumar Kannan is a ML developer currently'
                   ' working for a London-based Edtech'
                   ' company. He is interested in learning'
                   ' Natural Language Processing.'
                   ' He keeps organizing local Python meetups'
                   ' and several internal talks at his workplace.')
        my_text
```

```
Out[7]: 'Rajkumar Kannan is a ML developer currently working for a London-based Edtec
h company. He is interested in learning Natural Language Processing. He keeps
organizing local Python meetups and several internal talks at his workplace.'
```

Question4. For the list of strings from my_text,print the following for each token

In [8]:
```
text=sci(my_text)
for token in text:
    print(token,token.idx,token.text_with_ws,token.is_alpha,token.is_punct,tok
```

```
Rajkumar 0 Rajkumar  True False False Xxxxx False
Kannan 9 Kannan  True False False Xxxxx False
is 16 is  True False False xx True
a 19 a  True False False x True
ML 21 ML  True False False XX False
developer 24 developer  True False False xxxx False
currently 34 currently  True False False xxxx False
working 44 working  True False False xxxx False
for 52 for  True False False xxx True
a 56 a  True False False x True
London 58 London True False False Xxxxx False
- 64 - False True False - False
based 65 based  True False False xxxx False
Edtech 71 Edtech  True False False Xxxxx False
company 78 company True False False xxxx False
. 85 .  False True False . False
He 87 He  True False False Xx True
is 90 is  True False False xx True
interested 93 interested  True False False xxxx False
in 104 in  True False False xx True
learning 107 learning  True False False xxxx False
Natural 116 Natural  True False False Xxxxx False
Language 124 Language  True False False Xxxxx False
Processing 133 Processing True False False Xxxxx False
. 143 .  False True False . False
He 145 He  True False False Xx True
keeps 148 keeps  True False False xxxx False
organizing 154 organizing  True False False xxxx False
local 165 local  True False False xxxx False
Python 171 Python  True False False Xxxxx False
meetups 178 meetups  True False False xxxx False
and 186 and  True False False xxx True
several 190 several  True False False xxxx True
internal 198 internal  True False False xxxx False
talks 207 talks  True False False xxxx False
at 213 at  True False False xx True
his 216 his  True False False xxx True
workplace 220 workplace True False False xxxx False
. 229 . False True False . False
```

q5.Detect and print hypernated words from my_text.for example,london-based

In [9]:
```
import re
from spacy.tokenizer import Tokenizer
from spacy.util import *
```

In [10]:
```
def custom_tokenizer(sci):
    infix_re=re.compile(r'''[.\/\?\:\...\'\'\"\"\'~]''')
    prefix_re=compile_prefix_regex(sci.Defaults.prefixes)
    suffix_re=compile_suffix_regex(sci.Defaults.suffixes)
    return Tokenizer(sci.vocab,prefix_search=prefix_re.search,suffix_search=su
                       infix_finditer=infix_re.finditer,token_match=None)
    sci.Tokenizer=custom_tokenizer(sci)
```

In [11]:
```
text2=sci(my_text)
[token.text for token in text2]
```

Out[11]:
```
['Rajkumar',
 'Kannan',
 'is',
 'a',
 'ML',
 'developer',
 'currently',
 'working',
 'for',
 'a',
 'London',
 '-',
 'based',
 'Edtech',
 'company',
 '.',
 'He',
 'is',
 'interested',
 'in',
 'learning',
 'Natural',
 'Language',
 'Processing',
 '.',
 'He',
 'keeps',
 'organizing',
 'local',
 'Python',
 'meetups',
 'and',
 'several',
 'internal',
 'talks',
 'at',
 'his',
 'workplace',
 '.']
```

Question 6. Print all stop words defined in SpaCy

```
In [12]: sci.Defaults.stop_words
```

```
Out[12]: {"'d",
          "'ll",
          "'m",
          "'re",
          "'s",
          "'ve",
          'a',
          'about',
          'above',
          'across',
          'after',
          'afterwards',
          'again',
          'against',
          'all',
          'almost',
          'alone',
          'along',
          'already',
```

Question 7. Remove all stop words and print the rest of tokens from, my_text

```
In [13]: all_stopwords=sci.Defaults.stop_words
         [token.text for token in text if not token.text in all_stopwords]
```

```
Out[13]: ['Rajkumar',
          'Kannan',
          'ML',
          'developer',
          'currently',
          'working',
          'London',
          '-',
          'based',
          'Edtech',
          'company',
          '.',
          'He',
          'interested',
          'learning',
          'Natural',
          'Language',
          'Processing',
          '.',
          'He',
          'keeps',
          'organizing',
          'local',
          'Python',
          'meetups',
          'internal',
          'talks',
          'workplace',
          '.']
```

Question 8. Print all lemma from my_text

```
In [14]: for token in text2:
             print(token,token.lemma_)
```

```
Rajkumar Rajkumar
Kannan Kannan
is be
a a
ML ML
developer developer
currently currently
working work
for for
a a
London London
- -
based base
Edtech Edtech
company company
. .
He he
is be
interested interested
in in
learning learn
Natural Natural
Language Language
Processing processing
. .
He he
keeps keep
organizing organize
local local
Python Python
meetups meetup
and and
several several
internal internal
talks talk
at at
his his
workplace workplace
. .
```

Question 9. Perform Part of Speech Tagging on my_text and print the following tag informations.

```
In [15]: text2=sci(my_text)
         for token in text2:
             print(token.text,token.pos_,token.tag,spacy.explain(token.tag_))
```

```
Rajkumar PROPN 15794550382381185553 noun, proper singular
Kannan PROPN 15794550382381185553 noun, proper singular
is AUX 13927759927860985106 verb, 3rd person singular present
a DET 15267657372422890137 determiner
ML PROPN 15794550382381185553 noun, proper singular
developer NOUN 15308085513773655218 noun, singular or mass
currently ADV 164681854541413346 adverb
working VERB 1534113631682161808 verb, gerund or present participle
for ADP 1292078113972184607 conjunction, subordinating or preposition
a DET 15267657372422890137 determiner
London PROPN 15794550382381185553 noun, proper singular
- PUNCT 8214596291009089021 punctuation mark, hyphen
based VERB 3822385049556375858 verb, past participle
Edtech PROPN 15794550382381185553 noun, proper singular
company NOUN 15308085513773655218 noun, singular or mass
. PUNCT 12646065887601541794 punctuation mark, sentence closer
He PRON 13656873538139661788 pronoun, personal
is AUX 13927759927860985106 verb, 3rd person singular present
interested ADJ 10554686591937588953 adjective (English), other noun-modifier
(Chinese)
in ADP 1292078113972184607 conjunction, subordinating or preposition
learning VERB 1534113631682161808 verb, gerund or present participle
Natural PROPN 15794550382381185553 noun, proper singular
Language PROPN 15794550382381185553 noun, proper singular
Processing NOUN 15308085513773655218 noun, singular or mass
. PUNCT 12646065887601541794 punctuation mark, sentence closer
He PRON 13656873538139661788 pronoun, personal
keeps VERB 13927759927860985106 verb, 3rd person singular present
organizing VERB 1534113631682161808 verb, gerund or present participle
local ADJ 10554686591937588953 adjective (English), other noun-modifier (Chin
ese)
Python PROPN 15794550382381185553 noun, proper singular
meetups NOUN 783433942507015291 noun, plural
and CCONJ 17571114184892886314 conjunction, coordinating
several ADJ 10554686591937588953 adjective (English), other noun-modifier (Ch
inese)
internal ADJ 10554686591937588953 adjective (English), other noun-modifier (C
hinese)
talks NOUN 783433942507015291 noun, plural
at ADP 1292078113972184607 conjunction, subordinating or preposition
his PRON 4062917326063685704 pronoun, possessive
workplace NOUN 15308085513773655218 noun, singular or mass
. PUNCT 12646065887601541794 punctuation mark, sentence closer
```

Question 10. How many NOUN and ADJ are there in my_text?. Print them and its count.

```
In [16]: nouns=[]
         for noun in text2:
             if noun.pos_=='NOUN':
                 nouns.append(noun)
         print(len(nouns),nouns)

         6 [developer, company, Processing, meetups, talks, workplace]
```

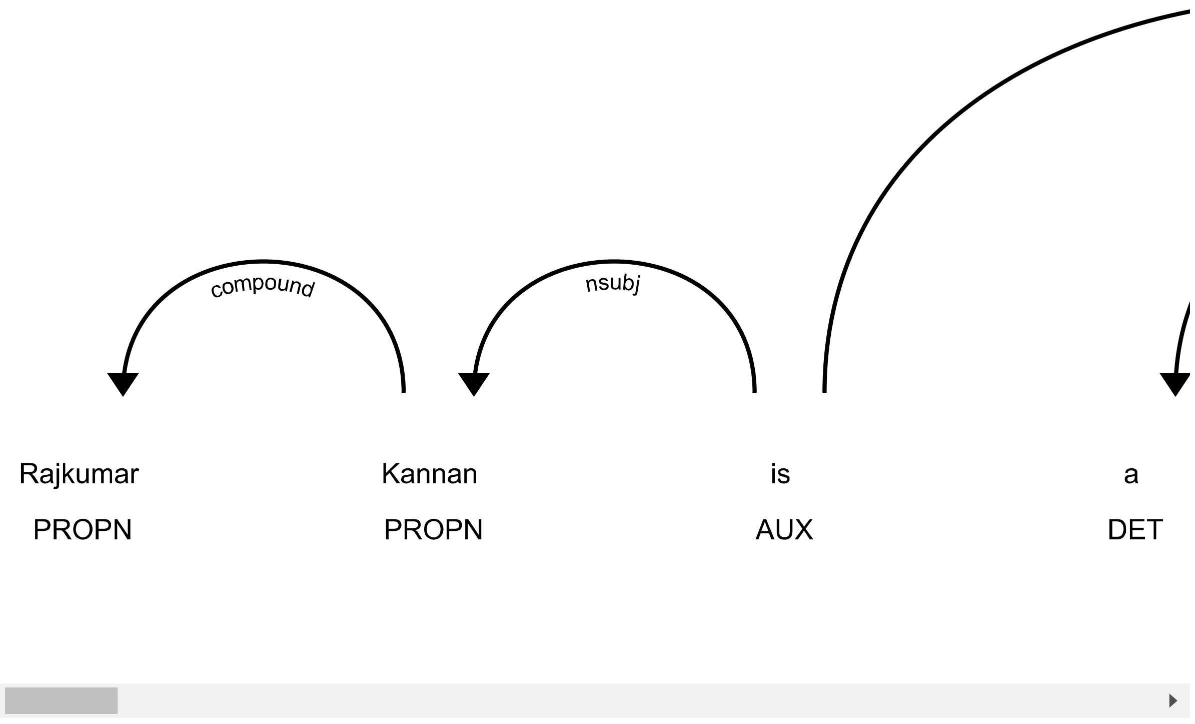
```
In [17]: adj=[]
         for adjec in text2:
             if token.pos_=="ADJ":
                 adj.append(adjec)
         print(len(adj),adj)

         0 []
```

Question 11. Visualize POS tags of a sentence, my_text, using displaCy

```
In [18]: from spacy import displacy
```

```
In [19]: displacy.render(text2,style='dep',jupyter=True)
```



Question 12. Extract and print First Name and Last Name from my_text using Matcher.

In [20]:
```
from spacy.matcher import Matcher
from spacy.tokens import Span
match=Matcher(sci.vocab)
match.add("PERSON",[[{"lower":"rajkumar"},{"lower":"kannan"}]])
matches=match(text2)
```

In [21]:
```
for match_id,start,end in matches:
    span=Span(text2,start,end,label=match_id)
    print(span.text,span.label_)
```

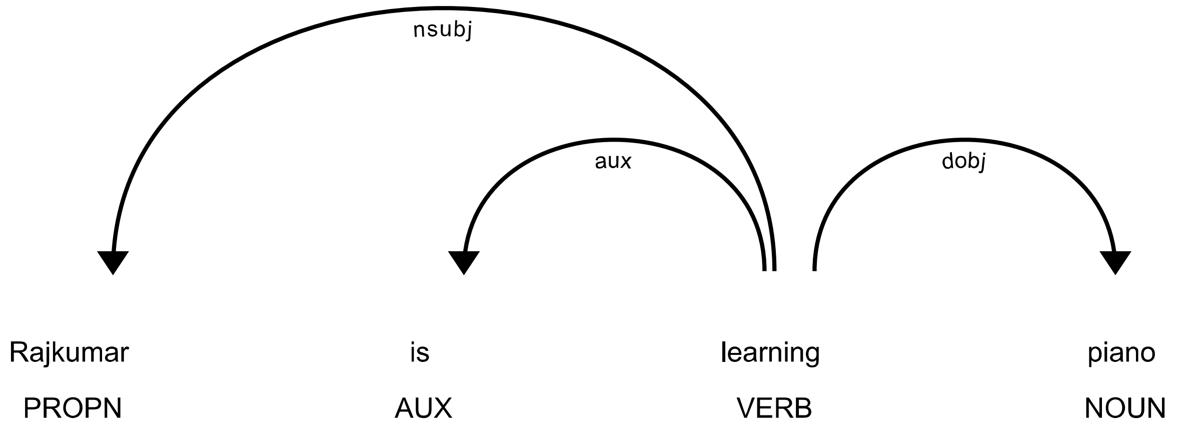
```
Rajkumar Kannan PERSON
```

Question 13. Print the dependency parse tag values for the text,

In [22]:
```
text=sci(u'Rajkumar is learning piano')
for tok in text:
    print(tok.text,tok.dep_)
```

```
Rajkumar nsubj
is aux
learning ROOT
piano dobj
```

In [23]:
```
displacy.render(text,style='dep',jupyter=True)
```



Question 14. Consider the following string.

```
In [24]: d_text=('Sam Peter is a Python developer currently working for a London-based I
```

```
In [25]: #a) children of "Developer"

         tex=sci(d_text)
         for text in tex[5].children:
             print(text.text)

         a
         Python
         working
```

```
In [26]: # previous neighbor

         tex[5].nbor(-1)

Out[26]: Python
```

```
In [27]: # next neighbor

         tex[5].nbor()

Out[27]: currently
```

```
In [28]: #d) all tokens on the left

         [text.text for text in tex[5].lefts]

Out[28]: ['a', 'Python']
```

```
In [29]: #e) tokens on the right

         [text.text for text in tex[5].rights]

Out[29]: ['working']
```

```
In [30]: #f) subtree of "Developer"

         [text.text for text in tex[5].subtree]

Out[30]: ['a',
          'Python',
          'developer',
          'currently',
          'working',
          'for',
          'a',
          'London',
          '-',
          'based',
          'Fintech',
          'company']
```

Q15.

```
In [31]: con_text=('There is a developer conference happenning on 21 July 2020 in New De
         con_tex=sci(con_text)
         for chunk in con_tex.noun_chunks:
             print(chunk)
```

```
a developer conference happenning
21 July
New Delhi
```

Question 16. Print all Verb Phrases in the text (you need to install textacy).

```
In [33]: import textacy
```

```
In [34]: about_talk_text=('The talk will introduce reader about Use'' cases of Natural I
```

```
In [35]: about_talk_text
```

```
Out[35]: 'The talk will introduce reader about Use cases of Natural Language Processin
         g in Fintech'
```

```
In [36]: about_talk_doc = sci(about_talk_text)
         pattern = re.compile(r'(<V>?<ADV>*<V>+)')

         for sentence in about_talk_doc.sents:
             verb_phrases = [chunk.text for chunk in sentence.noun_chunks if pattern.se
             print(verb_phrases)
```

```
[]
```

Question 17. Print all Named Entities in the text

```
In [37]: piano_class_text = ('Great Piano Academy is situated'
                             ' in Mayfair or the City of London and has'
                             ' world-class piano instructors.')
```

```
In [38]: piano_class_doc = sci(piano_class_text)
         for ent in piano_class_doc.ents:
             print(ent.text, ent.start_char, ent.end_char, ent.label_, spacy.explain(en
```

```
Great Piano Academy 0 19 ORG Companies, agencies, institutions, etc.
Mayfair 35 42 GPE Countries, cities, states
the City of London 46 64 GPE Countries, cities, states
```