Name: Arul Kumar ARK

225229103

Lab 15:

In [ ]:
```
                                Text Processing using SpaCy
```

Exercises

In [1]:
```
import spacy
nlp = spacy.load("en_core_web_sm")
```

In [2]:
```
# Question 1. Print the tokens of the string, “welcome all of you for this NLP

doc = nlp("welcome all of you for this NLP with spacy course")
for token in doc:
    print(token.text, token.pos_, token.dep_)
```

```
welcome VERB ROOT
all PRON dobj
of ADP prep
you PRON pobj
for ADP prep
this DET det
NLP NOUN pobj
with ADP prep
spacy NOUN compound
course NOUN pobj
```

In [3]:
```
# Question 2. Create a text file that contains the above string, open that file

with open("toks.txt",'w') as fp:
    for i in doc:
        fp.write(i.text)
        fp.write("\n")
fp.close()
```

In [4]:
```
f = open("toks.txt", "r")
print(f.read())
```

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

In [5]:
```
# Question 3. Consider the following sentences and print each sentence in one

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.')
```

In [8]:
```
# Question 4. For the list of strings from my_text, print the following for ea

doc1=nlp(my_text)
for token in doc1:
    print(token.text,token.lemma_,token.pos_,token.tag_,token.dep_,token.shape_
```

```
Rajkumar Rajkumar PROPN NNP compound Xxxxx True False
Kannan Kannan PROPN NNP nsubj Xxxxx True False
is be AUX VBZ ROOT xx True True
a a DET DT det x True True
ML ML PROPN NNP compound XX True False
developer developer NOUN NN attr xxxx True False
currently currently ADV RB advmod xxxx True False
working work VERB VBG acl xxxx True False
for for ADP IN prep xxx True True
a a DET DT det x True True
London London PROPN NNP npadvmod Xxxxx True False
- - PUNCT HYPH punct - False False
based base VERB VBN amod xxxx True False
Edtech Edtech PROPN NNP compound Xxxxx True False
company company NOUN NN pobj xxxx True False
. . PUNCT . punct . False False
He he PRON PRP nsubj Xx True True
is be AUX VBZ ROOT xx True True
interested interested ADJ JJ acomp xxxx True False
in in ADP IN prep xx True True
learning learn VERB VBG pcomp xxxx True False
Natural Natural PROPN NNP compound Xxxxx True False
Language Language PROPN NNP compound Xxxxx True False
Processing Processing PROPN NNP dobj Xxxxx True False
. . PUNCT . punct . False False
He he PRON PRP nsubj Xx True True
keeps keep VERB VBZ ROOT xxxx True False
organizing organize VERB VBG xcomp xxxx True False
local local ADJ JJ amod xxxx True False
Python Python PROPN NNP compound Xxxxx True False
meetups meetup NOUN NNS dobj xxxx True False
and and CCONJ CC cc xxx True True
several several ADJ JJ amod xxxx True True
internal internal ADJ JJ amod xxxx True False
talks talk NOUN NNS conj xxxx True False
at at ADP IN prep xx True True
his his PRON PRP$ poss xxx True True
workplace workplace NOUN NN pobj xxxx True False
. . PUNCT . punct . False False
```

In [12]:
```
# Question 5. Detect and print hyphenated words from my_text. For example, Lond

import re
import spacy
from spacy.tokenizer import Tokenizer
from spacy.util import *
def custom_tokenizer(nlp):
    infix_re = re.compile(r'''[.\,\?\:\;\...\‘\’\`\“\”\"\'~]''')
    prefix_re = compile_prefix_regex(nlp.Defaults.prefixes)
    suffix_re = compile_suffix_regex(nlp.Defaults.suffixes)

    return Tokenizer(nlp.vocab, prefix_search=prefix_re.search,
                     suffix_search=suffix_re.search,
                     infix_finditer=infix_re.finditer,
                     token_match=None)

nlp = spacy.load("en_core_web_sm")
nlp.tokenizer = custom_tokenizer(nlp)
```

In [13]:
```
doc = nlp(my_text)
[token.text for token in doc]
```

Out[13]:
```
['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',
 '.']
```

In [14]:
```
# Question 6. Print all stop words defined in SpaCy

print(nlp.Defaults.stop_words)
```

```
{'ours', 'up', 'take', 'although', 'did', 'yourself', 'besides', 'and', 'ha
d', 'another', 'here', 'whoever', 'fifteen', 'side', 'were', 'hundred', 'ar
e', 'put', 'can', 'we', 'who', 'ca', 'six', 'such', 'already', 'alone', 'here
upon', 'several', 'moreover', 'mostly', 'my', 'this', 'through', 'anyway', 't
ogether', 'has', 'otherwise', 'else', 'from', 'never', 'along', 'per', '‘re',
'twenty', 'other', 'whatever', 'seemed', 'at', 'just', 'nobody', 'ten', 'mos
t', 'part', 'when', 'throughout', 'how', 'then', 'it', 'by', 'someone', 'else
where', 're', 'beside', 'they', 'get', 'because', 'thereafter', 'almost',
'n‘t', 'once', 'them', 'latterly', 'nor', 'less', 'nothing', 'upon', 'into',
'which', 'three', 'whereby', 'during', 'some', 'himself', 'bottom', 'what',
'forty', '’s', 'these', 'noone', '‘ve', 'via', "'m", 'namely', 'amount', 'aft
er', 'our', 'give', 'whether', 'every', 'used', '‘s', 'or', 'same', 'whereupo
n', 'somehow', 'was', 'five', 'please', 'she', 'see', '’ve', 'i', 'third', 'c
ould', 'their', '’ll', 'under', "'d", 'its', 'everyone', 'there', 'be', 'whol
e', 'unless', 'against', 'onto', 'been', 'neither', 'seem', 'became', 'former
ly', 'latter', "'ll", 'but', 'that', 'whenever', 'regarding', 'always', 'eve
n', 'for', 'call', 'off', 'nevertheless', 'make', 'towards', 'various', '‘l
l', '’m', 'thru', 'eleven', 'yours', 'those', 'name', 'fifty', 'out', 'beyon
d', "'re", 'say', 'two', 'seems', 'nine', 'among', 'really', 'everything', 'e
verywhere', 'well', 'the', 'toward', 'between', 'below', 'much', 'over', 'six
ty', 'me', 'least', 'does', 'thereupon', 'sometimes', 'herein', 'twelve', 'wh
ere', 'us', 'perhaps', 'enough', 'he', 'each', 'move', 'beforehand', 'acros
s', 'cannot', 'itself', 'hereafter', 'four', 'ourselves', 'very', 'while', 'y
our', 'made', 'whose', 'since', 'as', 'next', 'only', 'a', 'to', 'so', 'all',
'except', 'go', 'further', "'ve", 'too', 'also', '’d', 'of', 'show', 'shoul
d', 'often', 'though', 'anything', 'above', 'therefore', 'whence', 'none', 'i
ndeed', 'become', 'until', 'being', 'however', 'not', 'whereafter', 'now',
"'s", 'using', 'front', 'than', 'back', 'due', '‘m', 'is', 'serious', 'anyho
w', 'hence', 'his', 'no', 'therein', 'would', 'am', 'about', 'themselves', 'w
ithin', 'thus', 'becoming', 'sometime', 'around', 'will', '’re', 'done', 'now
here', 'amongst', 'quite', 'before', 'ever', 'somewhere', 'myself', 'meanwhil
e', 'own', 'becomes', 'seeming', 'former', 'either', 'n’t', 'both', 'yourselv
es', 'if', 'in', 'something', 'doing', 'behind', 'must', 'you', '‘d', 'more',
'full', 'top', 'on', 'thereby', 'wherein', 'hers', 'have', 'herself', 'him',
'do', 'thence', 'many', 'anywhere', 'empty', 'any', 'one', 'whom', 'why', 'he
reby', 'others', 'keep', 'down', 'an', 'mine', 'with', 'wherever', 'may', 'ye
t', 'again', 'whereas', 'first', 'few', 'whither', "n't", 'rather', 'might',
'still', 'last', 'without', 'eight', 'her', 'afterwards', 'anyone'}
```

In [15]:
```
# Question 7. Remove all stop words and print the rest of tokens from, my_text

all_stopwords = nlp.Defaults.stop_words
[token.text for token in doc if not token.text in all_stopwords]
```

Out[15]:
```
['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',
 '.']
```

In [16]:
```
# Question 8. Print all lemma from my_text

for token in doc:
    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-based london-based
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
. .
```

In [18]:
```
# Question 9. Perform Part of Speech Tagging on my_text and print the following
# spacy.explain(token.tag_)

doc=nlp(my_text)
for token in doc:
    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-based ADJ 10554686591937588953 adjective (English), other noun-modifie
r (Chinese)
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 PROPN 15794550382381185553 noun, proper singular
. 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
```

In [20]:
```
# Question 10. How many NOUN and ADJ are there in my_text?. Print them and its

nouns = []
for token in doc:
    if token.pos_ == 'NOUN':
        nouns.append(token)
print(len(nouns),nouns)
```

```
5 [developer, company, meetups, talks, workplace]
```

In [21]:
```
adjectives = []
for token in doc:
    if token.pos_ == 'ADJ':
        adjectives.append(token)
print(len(adjectives),adjectives)
```

```
5 [London-based, interested, local, several, internal]
```

In [22]:
```
# Question 11. Visualize POS tags of a sentence, my_text, using displaCy

from spacy import displacy
displacy.render(doc, style='dep',jupyter=True)
```

Rajkumar PROPN Kannan PROPN is AUX a DET ML PROPN developer NOUN currently ADV working VERB for ADP a DET London-based ADJ Edtech PROPN company. NOUN He PRON is AUX interested ADJ in ADP learning VERB Natural PROPN Language PROPN Processing. PROPN He PRON keeps VERB organizing VERB local ADJ Python PROPN meetups NOUN and CCONJ several ADJ internal ADJ talks NOUN at ADP his PRON workplace. NOUN compound nsubj det compound attr advmod acl prep det amod compound pobj nsubj acomp prep pcomp compound compound dobj nsubj xcomp amod compound dobj cc amod amod conj prep poss pobj

In [23]:
```
# Question 12. Extract and print First Name and Last Name from my_text using M

from spacy.matcher import Matcher
from spacy.tokens import Span
matcher = Matcher(nlp.vocab)
matcher.add("PERSON", [[{"lower": "rajkumar"}, {"lower": "kannan"}]])
matches = matcher(doc)
for match_id, start, end in matches:
 # Create the matched span and assign the match_id as a label
    span = Span(doc, start, end, label=match_id)
    print(span.text, span.label_)
```

```
Rajkumar Kannan PERSON
```

In [24]:
```
# Question 13. Print the dependency parse tag values for the text,
# “Rajkumar is learning piano”. Also, display dependency parse tree using disp

doc = nlp(u'Rajkumar is learning piano')
for token in doc:
    print(token.text, token.dep_)
displacy.render(doc, style='dep',jupyter=True)
```

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

Rajkumar PROPN is AUX learning VERB piano NOUN nsubj aux dobj

In [25]:
```
# Question 14. Consider the following string.

# a. Print the children of developer

d_text = ('Sam Peter is a Python developer currently working for a London-base

doc = nlp(d_text)
for t in doc[5].children:
    print(t.text)
```

```
a
Python
working
```

In [26]:
```
# b. Print the previous neighboring node of developer

print (doc[5].nbor(-1))
```

```
Python
```

In [27]:
```
# c. Print the next neighboring node of developer

print (doc[5].nbor())
```

```
currently
```

In [28]:
```
# d. Print the all tokens on the left of developer

[t.text for t in doc[5].lefts]
```

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

In [29]:
```
# e. Print the tokens on the right of developer

[t.text for t in doc[5].rights]
```

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

In [30]:
```
# f. Print the Print subtree of developer

[t.text for t in doc[5].subtree]
```

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

In [31]:
```
# Question 15. Print all Noun Phrases in the text

conference_text = ('There is a developer conference happening on 21 July 2020 :
conference_doc = nlp(conference_text)
for chunk in conference_doc.noun_chunks:
    print(chunk)
```

```
a developer conference
July
New Delhi
```

In [35]:
```
# Question 16. Print all Verb Phrases in the text (you need to install textacy)

'''import spacy,en_core_web_sm
import textacy
about_talk_text = ('The talk will introduce reader about Use'
 ' cases of Natural Language Processing in'
' Fintech')
pattern = r'(<VERB>?<ADV>*<VERB>+)'
about_talk_doc = textacy.make_spacy_doc(about_talk_text, lang='en_core_web_sm'
verb_phrases = textacy.extract.pos_regex_matches(about_talk_doc, pattern)
for chunk in verb_phrases:
     print(chunk.text)
for chunk in about_talk_doc.noun_chunks:
     print(chunk)
'''
```

Out[35]:
```
"import spacy,en_core_web_sm\nimport textacy\nabout_talk_text = ('The talk wi
ll introduce reader about Use'\n ' cases of Natural Language Processing i
n'\n' Fintech')\npattern = r'(<VERB>?<ADV>*<VERB>+)'\nabout_talk_doc = textac
y.make_spacy_doc(about_talk_text, lang='en_core_web_sm')\nverb_phrases = text
acy.extract.pos_regex_matches(about_talk_doc, pattern)\nfor chunk in verb_phr
ases:\n     print(chunk.text)\nfor chunk in about_talk_doc.noun_chunks:\n
print(chunk)\n"
```

In [36]:
```
# Question 17. Print all Named Entities in the text

piano_class_text = ('Great Piano Academy is situated'
 ' in Mayfair or the City of London and has'
 ' world-class piano instructors.')
piano_class_doc = nlp(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 LOC Non-GPE locations, mountain ranges, bodies of water
the City of London 46 64 GPE Countries, cities, states
```

In [ ]: