Group1_Lab1

February 16, 2024

1 Lab1-Assignment

This notebook describes the assignment for Lab 1 of the text mining course.

Points: each exercise is prefixed with the number of points you can obtain for the exercise.

We assume you have worked through the following notebooks: * **Lab1.1-introduction** * **Lab1.2-introduction-to-NLTK** * **Lab1.3-introduction-to-spaCy**

In this assignment, you will process an English text (**Lab1-apple-samsung-example.txt**) with both NLTK and spaCy and discuss the similarities and differences.

1.1 Credits

The notebooks in this block have been originally created by Marten Postma. Adaptations were made by Filip Ilievski.

1.2 Tip: how to read a file from disk

Let's open the file **Lab1-apple-samsung-example.txt** from disk.

[4]:
```
from pathlib import Path
```

[5]:
```
cur_dir = Path().resolve() # this should provide you with the folder in which␣
↪this notebook is placed
path_to_file = Path.joinpath(cur_dir, 'Lab1-apple-samsung-example.txt')
print(path_to_file)
print('does path exist? ->', Path.exists(path_to_file))
```

```
/Users/stefaniaconte/Desktop/newenv/Lab1-apple-samsung-example.txt
does path exist? -> True
```

If the output from the code cell above states that **does path exist? -> False**, please check that the file **Lab1-apple-samsung-example.txt** is in the same directory as this notebook.

[6]:
```
with open(path_to_file) as infile:
    text = infile.read()

print('number of characters', len(text))
```

```
number of characters 1139
```

1.3 [total points: 4] Exercise 1: NLTK

In this exercise, we use NLTK to apply **Part-of-speech (POS) tagging**, **Named Entity Recognition (NER)**, and **Constituency parsing**. The following code snippet already performs sentence splitting and tokenization.

[7]:
```
import nltk
from nltk.tokenize import sent_tokenize
from nltk import word_tokenize
```

[8]:
```
sentences_nltk = sent_tokenize(text)
```

[9]:
```
tokens_per_sentence = []
for sentence_nltk in sentences_nltk:
    sent_tokens = word_tokenize(sentence_nltk)
    tokens_per_sentence.append(sent_tokens)
```

We will use lists to keep track of the output of the NLP tasks. We can hence inspect the output for each task using the index of the sentence.

[10]:
```
sent_id = 1
print('SENTENCE', sentences_nltk[sent_id])
print('TOKENS', tokens_per_sentence[sent_id])
```

```
SENTENCE The six phones and tablets affected are the Galaxy S III, running the
new Jelly Bean system, the Galaxy Tab 8.9 Wifi tablet, the Galaxy Tab 2 10.1,
Galaxy Rugby Pro and Galaxy S III mini.
TOKENS ['The', 'six', 'phones', 'and', 'tablets', 'affected', 'are', 'the',
'Galaxy', 'S', 'III', ',', 'running', 'the', 'new', 'Jelly', 'Bean', 'system',
',', 'the', 'Galaxy', 'Tab', '8.9', 'Wifi', 'tablet', ',', 'the', 'Galaxy',
'Tab', '2', '10.1', ',', 'Galaxy', 'Rugby', 'Pro', 'and', 'Galaxy', 'S', 'III',
'mini', '.']
```

1.3.1 [point: 1] Exercise 1a: Part-of-speech (POS) tagging

Use `nltk.pos_tag` to perform part-of-speech tagging on each sentence.

Use `print` to **show** the output in the notebook (and hence also in the exported PDF!).

[11]:
```
pos_tags_per_sentence = [] #stores POS tags for each sentence List[Tuple]
for tokens in tokens_per_sentence:
    pos_tags = nltk.pos_tag(tokens) #each tuple consists of a token and its␣
 ↪corresponding POS tag
    pos_tags_per_sentence.append(pos_tags)
    #print(pos_tags) #prints POS-tagged tokens for one sentence at a time
```

[12]:
```
print(pos_tags_per_sentence)
```

```
[[('https', 'NN'), (':', ':'),
('//www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-more-
products-under-scrutiny.html', 'JJ'), ('Documents', 'NNS'), ('filed', 'VBN'),
('to', 'TO'), ('the', 'DT'), ('San', 'NNP'), ('Jose', 'NNP'), ('federal', 'JJ'),
('court', 'NN'), ('in', 'IN'), ('California', 'NNP'), ('on', 'IN'), ('November',
'NNP'), ('23', 'CD'), ('list', 'NN'), ('six', 'CD'), ('Samsung', 'NNP'),
('products', 'NNS'), ('running', 'VBG'), ('the', 'DT'), ('``', '``'), ('Jelly',
'RB'), ('Bean', 'NNP'), ("''", "''"), ('and', 'CC'), ('``', '``'), ('Ice',
'NNP'), ('Cream', 'NNP'), ('Sandwich', 'NNP'), ("''", "''"), ('operating',
'VBG'), ('systems', 'NNS'), (',', ','), ('which', 'WDT'), ('Apple', 'NNP'),
('claims', 'VBZ'), ('infringe', 'VB'), ('its', 'PRP$'), ('patents', 'NNS'),
('.', '.')], [('The', 'DT'), ('six', 'CD'), ('phones', 'NNS'), ('and', 'CC'),
('tablets', 'NNS'), ('affected', 'VBN'), ('are', 'VBP'), ('the', 'DT'),
('Galaxy', 'NNP'), ('S', 'NNP'), ('III', 'NNP'), (',', ','), ('running', 'VBG'),
('the', 'DT'), ('new', 'JJ'), ('Jelly', 'NNP'), ('Bean', 'NNP'), ('system',
'NN'), (',', ','), ('the', 'DT'), ('Galaxy', 'NNP'), ('Tab', 'NNP'), ('8.9',
'CD'), ('Wifi', 'NNP'), ('tablet', 'NN'), (',', ','), ('the', 'DT'), ('Galaxy',
'NNP'), ('Tab', 'NNP'), ('2', 'CD'), ('10.1', 'CD'), (',', ','), ('Galaxy',
'NNP'), ('Rugby', 'NNP'), ('Pro', 'NNP'), ('and', 'CC'), ('Galaxy', 'NNP'),
('S', 'NNP'), ('III', 'NNP'), ('mini', 'NN'), ('.', '.')], [('Apple', 'NNP'),
('stated', 'VBD'), ('it', 'PRP'), ('had', 'VBD'), ('"', 'NNP'), ('acted',
'VBD'), ('quickly', 'RB'), ('and', 'CC'), ('diligently', 'RB'), ("''", "''"),
('in', 'IN'), ('order', 'NN'), ('to', 'TO'), ('``', '``'), ('determine', 'VB'),
('that', 'IN'), ('these', 'DT'), ('newly', 'RB'), ('released', 'VBN'),
('products', 'NNS'), ('do', 'VBP'), ('infringe', 'VB'), ('many', 'JJ'), ('of',
'IN'), ('the', 'DT'), ('same', 'JJ'), ('claims', 'NNS'), ('already', 'RB'),
('asserted', 'VBN'), ('by', 'IN'), ('Apple', 'NNP'), ('.', '.'), ("''", "''")],
[('In', 'IN'), ('August', 'NNP'), (',', ','), ('Samsung', 'NNP'), ('lost',
'VBD'), ('a', 'DT'), ('US', 'NNP'), ('patent', 'NN'), ('case', 'NN'), ('to',
'TO'), ('Apple', 'NNP'), ('and', 'CC'), ('was', 'VBD'), ('ordered', 'VBN'),
('to', 'TO'), ('pay', 'VB'), ('its', 'PRP$'), ('rival', 'JJ'), ('$', '$'),
('1.05bn', 'CD'), ('(', '('), ('£0.66bn', 'NN'), (')', ')'), ('in', 'IN'),
('damages', 'NNS'), ('for', 'IN'), ('copying', 'VBG'), ('features', 'NNS'),
('of', 'IN'), ('the', 'DT'), ('iPad', 'NN'), ('and', 'CC'), ('iPhone', 'NN'),
('in', 'IN'), ('its', 'PRP$'), ('Galaxy', 'NNP'), ('range', 'NN'), ('of', 'IN'),
('devices', 'NNS'), ('.', '.')], [('Samsung', 'NNP'), (',', ','), ('which',
'WDT'), ('is', 'VBZ'), ('the', 'DT'), ('world', 'NN'), ("'s", 'POS'), ('top',
'JJ'), ('mobile', 'NN'), ('phone', 'NN'), ('maker', 'NN'), (',', ','), ('is',
'VBZ'), ('appealing', 'VBG'), ('the', 'DT'), ('ruling', 'NN'), ('.', '.')],
[('A', 'DT'), ('similar', 'JJ'), ('case', 'NN'), ('in', 'IN'), ('the', 'DT'),
('UK', 'NNP'), ('found', 'VBD'), ('in', 'IN'), ('Samsung', 'NNP'), ("'s",
'POS'), ('favour', 'NN'), ('and', 'CC'), ('ordered', 'VBD'), ('Apple', 'NNP'),
('to', 'TO'), ('publish', 'VB'), ('an', 'DT'), ('apology', 'NN'), ('making',
'VBG'), ('clear', 'JJ'), ('that', 'IN'), ('the', 'DT'), ('South', 'JJ'),
('Korean', 'JJ'), ('firm', 'NN'), ('had', 'VBD'), ('not', 'RB'), ('copied',
'VBN'), ('its', 'PRP$'), ('iPad', 'NN'), ('when', 'WRB'), ('designing', 'VBG'),
('its', 'PRP$'), ('own', 'JJ'), ('devices', 'NNS'), ('.', '.')]]
```

1.3.2 [point: 1] Exercise 1b: Named Entity Recognition (NER)

Use `nltk.chunk.ne_chunk` to perform Named Entity Recognition (NER) on each sentence.

Use `print` to **show** the output in the notebook (and hence also in the exported PDF!).

[13]:
```
from nltk.chunk import ne_chunk
```

[14]:
```
ner_tags_per_sentence = []
for pos_tags in pos_tags_per_sentence: #from before (List[Tuple])
    ner_tree = ne_chunk(pos_tags)
    ner_tags_per_sentence.append(ner_tree)
    #print(ner_tree)
```

[15]:
```
print(ner_tags_per_sentence)
```

```
[Tree('S', [('https', 'NN'), (':', ':'),
('//www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-more-
products-under-scrutiny.html', 'JJ'), ('Documents', 'NNS'), ('filed', 'VBN'),
('to', 'TO'), ('the', 'DT'), Tree('ORGANIZATION', [('San', 'NNP'), ('Jose',
'NNP')]), ('federal', 'JJ'), ('court', 'NN'), ('in', 'IN'), Tree('GPE',
[('California', 'NNP')]), ('on', 'IN'), ('November', 'NNP'), ('23', 'CD'),
('list', 'NN'), ('six', 'CD'), Tree('ORGANIZATION', [('Samsung', 'NNP')]),
('products', 'NNS'), ('running', 'VBG'), ('the', 'DT'), ('``', '``'), ('Jelly',
'RB'), Tree('GPE', [('Bean', 'NNP')]), ("''", "''"), ('and', 'CC'), ('``',
'``'), ('Ice', 'NNP'), ('Cream', 'NNP'), ('Sandwich', 'NNP'), ("''", "''"),
('operating', 'VBG'), ('systems', 'NNS'), (',', ','), ('which', 'WDT'),
Tree('PERSON', [('Apple', 'NNP')]), ('claims', 'VBZ'), ('infringe', 'VB'),
('its', 'PRP$'), ('patents', 'NNS'), ('.', '.')]), Tree('S', [('The', 'DT'),
('six', 'CD'), ('phones', 'NNS'), ('and', 'CC'), ('tablets', 'NNS'),
('affected', 'VBN'), ('are', 'VBP'), ('the', 'DT'), Tree('ORGANIZATION',
[('Galaxy', 'NNP')]), ('S', 'NNP'), ('III', 'NNP'), (',', ','), ('running',
'VBG'), ('the', 'DT'), ('new', 'JJ'), Tree('PERSON', [('Jelly', 'NNP'), ('Bean',
'NNP')]), ('system', 'NN'), (',', ','), ('the', 'DT'), Tree('ORGANIZATION',
[('Galaxy', 'NNP')]), ('Tab', 'NNP'), ('8.9', 'CD'), ('Wifi', 'NNP'), ('tablet',
'NN'), (',', ','), ('the', 'DT'), Tree('ORGANIZATION', [('Galaxy', 'NNP')]),
('Tab', 'NNP'), ('2', 'CD'), ('10.1', 'CD'), (',', ','), Tree('PERSON',
[('Galaxy', 'NNP'), ('Rugby', 'NNP'), ('Pro', 'NNP')]), ('and', 'CC'),
Tree('PERSON', [('Galaxy', 'NNP'), ('S', 'NNP')]), ('III', 'NNP'), ('mini',
'NN'), ('.', '.')]), Tree('S', [Tree('PERSON', [('Apple', 'NNP')]), ('stated',
'VBD'), ('it', 'PRP'), ('had', 'VBD'), ('"', 'NNP'), ('acted', 'VBD'),
('quickly', 'RB'), ('and', 'CC'), ('diligently', 'RB'), ("''", "''"), ('in',
'IN'), ('order', 'NN'), ('to', 'TO'), ('``', '``'), ('determine', 'VB'),
('that', 'IN'), ('these', 'DT'), ('newly', 'RB'), ('released', 'VBN'),
('products', 'NNS'), ('do', 'VBP'), ('infringe', 'VB'), ('many', 'JJ'), ('of',
'IN'), ('the', 'DT'), ('same', 'JJ'), ('claims', 'NNS'), ('already', 'RB'),
('asserted', 'VBN'), ('by', 'IN'), Tree('PERSON', [('Apple', 'NNP')]), ('.',
'.'), ("''", "''")]), Tree('S', [('In', 'IN'), Tree('GPE', [('August', 'NNP')]),
(',', ','), Tree('PERSON', [('Samsung', 'NNP')]), ('lost', 'VBD'), ('a', 'DT'),
```

```
Tree('GSP', [('US', 'NNP')]), ('patent', 'NN'), ('case', 'NN'), ('to', 'TO'),
Tree('GPE', [('Apple', 'NNP')]), ('and', 'CC'), ('was', 'VBD'), ('ordered',
'VBN'), ('to', 'TO'), ('pay', 'VB'), ('its', 'PRP$'), ('rival', 'JJ'), ('$',
'$'), ('1.05bn', 'CD'), ('(', '('), ('£0.66bn', 'NN'), (')', ')'), ('in', 'IN'),
('damages', 'NNS'), ('for', 'IN'), ('copying', 'VBG'), ('features', 'NNS'),
('of', 'IN'), ('the', 'DT'), Tree('ORGANIZATION', [('iPad', 'NN')]), ('and',
'CC'), Tree('ORGANIZATION', [('iPhone', 'NN')]), ('in', 'IN'), ('its', 'PRP$'),
Tree('GPE', [('Galaxy', 'NNP')]), ('range', 'NN'), ('of', 'IN'), ('devices',
'NNS'), ('.', '.')]), Tree('S', [Tree('GPE', [('Samsung', 'NNP')]), (',', ','),
('which', 'WDT'), ('is', 'VBZ'), ('the', 'DT'), ('world', 'NN'), ("'s", 'POS'),
('top', 'JJ'), ('mobile', 'NN'), ('phone', 'NN'), ('maker', 'NN'), (',', ','),
('is', 'VBZ'), ('appealing', 'VBG'), ('the', 'DT'), ('ruling', 'NN'), ('.',
'.')]), Tree('S', [('A', 'DT'), ('similar', 'JJ'), ('case', 'NN'), ('in', 'IN'),
('the', 'DT'), Tree('ORGANIZATION', [('UK', 'NNP')]), ('found', 'VBD'), ('in',
'IN'), Tree('GPE', [('Samsung', 'NNP')]), ("'s", 'POS'), ('favour', 'NN'),
('and', 'CC'), ('ordered', 'VBD'), Tree('PERSON', [('Apple', 'NNP')]), ('to',
'TO'), ('publish', 'VB'), ('an', 'DT'), ('apology', 'NN'), ('making', 'VBG'),
('clear', 'JJ'), ('that', 'IN'), ('the', 'DT'), Tree('LOCATION', [('South',
'JJ'), ('Korean', 'JJ')]), ('firm', 'NN'), ('had', 'VBD'), ('not', 'RB'),
('copied', 'VBN'), ('its', 'PRP$'), ('iPad', 'NN'), ('when', 'WRB'),
('designing', 'VBG'), ('its', 'PRP$'), ('own', 'JJ'), ('devices', 'NNS'), ('.',
'.')])]
```

1.3.3 [points: 2] Exercise 1c: Constituency parsing

Use the `nltk.RegexpParser` to perform constituency parsing on each sentence.

Use `print` to **show** the output in the notebook (and hence also in the exported PDF!).

[16]:
```
constituent_parser = nltk.RegexpParser('''
NP: {<DT>? <JJ>* <NN>*} # NP
P: {<IN>}           # Preposition
V: {<V.*>}          # Verb
PP: {<P> <NP>}      # PP -> P NP
VP: {<V> <NP|PP>*}  # VP -> V (NP|PP)*''')
```

[17]:
```
constituency_output_per_sentence = []
for pos_tags in pos_tags_per_sentence:
    # Parse the POS-tagged sentence using the defined grammar
    parse_tree = constituent_parser.parse(pos_tags)
    constituency_output_per_sentence.append(parse_tree)
    #print(parse_tree)
```

[18]:
```
print(constituency_output_per_sentence)
```

```
[Tree('S', [Tree('NP', [('https', 'NN')]), (':', ':'), Tree('NP',
[('//www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-
more-products-under-scrutiny.html', 'JJ')]), ('Documents', 'NNS'), Tree('VP',
[Tree('V', [('filed', 'VBN')])]), ('to', 'TO'), Tree('NP', [('the', 'DT')]),
```

```
('San', 'NNP'), ('Jose', 'NNP'), Tree('NP', [('federal', 'JJ'), ('court',
'NN')]), Tree('P', [('in', 'IN')]), ('California', 'NNP'), Tree('P', [('on',
'IN')]), ('November', 'NNP'), ('23', 'CD'), Tree('NP', [('list', 'NN')]),
('six', 'CD'), ('Samsung', 'NNP'), ('products', 'NNS'), Tree('VP', [Tree('V',
[('running', 'VBG')]), Tree('NP', [('the', 'DT')])]), ('``', '``'), ('Jelly',
'RB'), ('Bean', 'NNP'), ("''", "''"), ('and', 'CC'), ('``', '``'), ('Ice',
'NNP'), ('Cream', 'NNP'), ('Sandwich', 'NNP'), ("''", "''"), Tree('VP',
[Tree('V', [('operating', 'VBG')])]), ('systems', 'NNS'), (',', ','), ('which',
'WDT'), ('Apple', 'NNP'), Tree('VP', [Tree('V', [('claims', 'VBZ')])]),
Tree('VP', [Tree('V', [('infringe', 'VB')])]), ('its', 'PRP$'), ('patents',
'NNS'), ('.', '.')]), Tree('S', [Tree('NP', [('The', 'DT')]), ('six', 'CD'),
('phones', 'NNS'), ('and', 'CC'), ('tablets', 'NNS'), Tree('VP', [Tree('V',
[('affected', 'VBN')])]), Tree('VP', [Tree('V', [('are', 'VBP')]), Tree('NP',
[('the', 'DT')])]), ('Galaxy', 'NNP'), ('S', 'NNP'), ('III', 'NNP'), (',', ','),
Tree('VP', [Tree('V', [('running', 'VBG')]), Tree('NP', [('the', 'DT'), ('new',
'JJ')])]), ('Jelly', 'NNP'), ('Bean', 'NNP'), Tree('NP', [('system', 'NN')]),
(',', ','), Tree('NP', [('the', 'DT')]), ('Galaxy', 'NNP'), ('Tab', 'NNP'),
('8.9', 'CD'), ('Wifi', 'NNP'), Tree('NP', [('tablet', 'NN')]), (',', ','),
Tree('NP', [('the', 'DT')]), ('Galaxy', 'NNP'), ('Tab', 'NNP'), ('2', 'CD'),
('10.1', 'CD'), (',', ','), ('Galaxy', 'NNP'), ('Rugby', 'NNP'), ('Pro', 'NNP'),
('and', 'CC'), ('Galaxy', 'NNP'), ('S', 'NNP'), ('III', 'NNP'), Tree('NP',
[('mini', 'NN')]), ('.', '.')]), Tree('S', [('Apple', 'NNP'), Tree('VP',
[Tree('V', [('stated', 'VBD')])]), ('it', 'PRP'), Tree('VP', [Tree('V', [('had',
'VBD')])]), ('"', 'NNP'), Tree('VP', [Tree('V', [('acted', 'VBD')])]),
('quickly', 'RB'), ('and', 'CC'), ('diligently', 'RB'), ("''", "''"), Tree('PP',
[Tree('P', [('in', 'IN')]), Tree('NP', [('order', 'NN')])]), ('to', 'TO'),
('``', '``'), Tree('VP', [Tree('V', [('determine', 'VB')]), Tree('PP',
[Tree('P', [('that', 'IN')]), Tree('NP', [('these', 'DT')])])]), ('newly',
'RB'), Tree('VP', [Tree('V', [('released', 'VBN')])]), ('products', 'NNS'),
Tree('VP', [Tree('V', [('do', 'VBP')])]), Tree('VP', [Tree('V', [('infringe',
'VB')]), Tree('NP', [('many', 'JJ')]), Tree('PP', [Tree('P', [('of', 'IN')]),
Tree('NP', [('the', 'DT'), ('same', 'JJ')])])]), ('claims', 'NNS'), ('already',
'RB'), Tree('VP', [Tree('V', [('asserted', 'VBN')])]), Tree('P', [('by',
'IN')]), ('Apple', 'NNP'), ('.', '.'), ("''", "''")]), Tree('S', [Tree('P',
[('In', 'IN')]), ('August', 'NNP'), (',', ','), ('Samsung', 'NNP'), Tree('VP',
[Tree('V', [('lost', 'VBD')]), Tree('NP', [('a', 'DT')])]), ('US', 'NNP'),
Tree('NP', [('patent', 'NN'), ('case', 'NN')]), ('to', 'TO'), ('Apple', 'NNP'),
('and', 'CC'), Tree('VP', [Tree('V', [('was', 'VBD')])]), Tree('VP', [Tree('V',
[('ordered', 'VBN')])]), ('to', 'TO'), Tree('VP', [Tree('V', [('pay', 'VB')])]),
('its', 'PRP$'), Tree('NP', [('rival', 'JJ')]), ('$', '$'), ('1.05bn', 'CD'),
('(', '('), Tree('NP', [('£0.66bn', 'NN')]), (')', ')'), Tree('P', [('in',
'IN')]), ('damages', 'NNS'), Tree('P', [('for', 'IN')]), Tree('VP', [Tree('V',
[('copying', 'VBG')])]), ('features', 'NNS'), Tree('PP', [Tree('P', [('of',
'IN')]), Tree('NP', [('the', 'DT'), ('iPad', 'NN')])]), ('and', 'CC'),
Tree('NP', [('iPhone', 'NN')]), Tree('P', [('in', 'IN')]), ('its', 'PRP$'),
('Galaxy', 'NNP'), Tree('NP', [('range', 'NN')]), Tree('P', [('of', 'IN')]),
('devices', 'NNS'), ('.', '.')]), Tree('S', [('Samsung', 'NNP'), (',', ','),
('which', 'WDT'), Tree('VP', [Tree('V', [('is', 'VBZ')]), Tree('NP', [('the',
```

```
'DT'), ('world', 'NN')])]), ("'s", 'POS'), Tree('NP', [('top', 'JJ'), ('mobile',
'NN'), ('phone', 'NN'), ('maker', 'NN')]), (',', ','), Tree('VP', [Tree('V',
[('is', 'VBZ')])]), Tree('VP', [Tree('V', [('appealing', 'VBG')]), Tree('NP',
[('the', 'DT'), ('ruling', 'NN')])]), ('.', '.')]), Tree('S', [Tree('NP', [('A',
'DT'), ('similar', 'JJ'), ('case', 'NN')]), Tree('PP', [Tree('P', [('in',
'IN')]), Tree('NP', [('the', 'DT')])]), ('UK', 'NNP'), Tree('VP', [Tree('V',
[('found', 'VBD')])]), Tree('P', [('in', 'IN')]), ('Samsung', 'NNP'), ("'s",
'POS'), Tree('NP', [('favour', 'NN')]), ('and', 'CC'), Tree('VP', [Tree('V',
[('ordered', 'VBD')])]), ('Apple', 'NNP'), ('to', 'TO'), Tree('VP', [Tree('V',
[('publish', 'VB')]), Tree('NP', [('an', 'DT'), ('apology', 'NN')])]),
Tree('VP', [Tree('V', [('making', 'VBG')]), Tree('NP', [('clear', 'JJ')]),
Tree('PP', [Tree('P', [('that', 'IN')]), Tree('NP', [('the', 'DT'), ('South',
'JJ'), ('Korean', 'JJ'), ('firm', 'NN')])])]), Tree('VP', [Tree('V', [('had',
'VBD')])]), ('not', 'RB'), Tree('VP', [Tree('V', [('copied', 'VBN')])]), ('its',
'PRP$'), Tree('NP', [('iPad', 'NN')]), ('when', 'WRB'), Tree('VP', [Tree('V',
[('designing', 'VBG')])]), ('its', 'PRP$'), Tree('NP', [('own', 'JJ')]),
('devices', 'NNS'), ('.', '.')])]
```

Augment the RegexpParser so that it also detects Named Entity Phrases (NEP), e.g., that it detects *Galaxy S III* and *Ice Cream Sandwich*

[19]:
```
constituent_parser_v2 = nltk.RegexpParser('''
NP: {<DT>? <JJ>* <NN>*} # NP
P: {<IN>}           # Preposition
V: {<V.*>}          # Verb
PP: {<P> <NP>}      # PP -> P NP
VP: {<V> <NP|PP>*}  # VP -> V (NP|PP)*
NEP: {}             # ???''')
```

[20]:
```
constituency_v2_output_per_sentence = []
for pos_tags in pos_tags_per_sentence:
    # Parse the POS-tagged sentence using the defined grammar
    parse_tree = constituent_parser_v2.parse(pos_tags)
    constituency_v2_output_per_sentence.append(parse_tree)
    #print(parse_tree)
```

[21]:
```
print(constituency_v2_output_per_sentence)
```

```
[Tree('S', [Tree('NP', [('https', 'NN')]), (':', ':'), Tree('NP',
[('//www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-
more-products-under-scrutiny.html', 'JJ')]), ('Documents', 'NNS'), Tree('VP',
[Tree('V', [('filed', 'VBN')])]), ('to', 'TO'), Tree('NP', [('the', 'DT')]),
('San', 'NNP'), ('Jose', 'NNP'), Tree('NP', [('federal', 'JJ'), ('court',
'NN')]), Tree('P', [('in', 'IN')]), ('California', 'NNP'), Tree('P', [('on',
'IN')]), ('November', 'NNP'), ('23', 'CD'), Tree('NP', [('list', 'NN')]),
('six', 'CD'), ('Samsung', 'NNP'), ('products', 'NNS'), Tree('VP', [Tree('V',
[('running', 'VBG')]), Tree('NP', [('the', 'DT')])]), ('``', '``'), ('Jelly',
'RB'), ('Bean', 'NNP'), ("''", "''"), ('and', 'CC'), ('``', '``'), ('Ice',
```

'NNP'), ('Cream', 'NNP'), ('Sandwich', 'NNP'), ("'", "'"), Tree('VP',
[Tree('V', [('operating', 'VBG')])]), ('systems', 'NNS'), (',', ','), ('which',
'WDT'), ('Apple', 'NNP'), Tree('VP', [Tree('V', [('claims', 'VBZ')])]),
Tree('VP', [Tree('V', [('infringe', 'VB')])]), ('its', 'PRP$'), ('patents',
'NNS'), ('.', '.')]), Tree('S', [Tree('NP', [('The', 'DT')]), ('six', 'CD'),
('phones', 'NNS'), ('and', 'CC'), ('tablets', 'NNS'), Tree('VP', [Tree('V',
[('affected', 'VBN')])]), Tree('VP', [Tree('V', [('are', 'VBP')]), Tree('NP',
[('the', 'DT')])]), ('Galaxy', 'NNP'), ('S', 'NNP'), ('III', 'NNP'), (',', ','),
Tree('VP', [Tree('V', [('running', 'VBG')]), Tree('NP', [('the', 'DT'), ('new',
'JJ')])]), ('Jelly', 'NNP'), ('Bean', 'NNP'), Tree('NP', [('system', 'NN')]),
(',', ','), Tree('NP', [('the', 'DT')]), ('Galaxy', 'NNP'), ('Tab', 'NNP'),
('8.9', 'CD'), ('Wifi', 'NNP'), Tree('NP', [('tablet', 'NN')]), (',', ','),
Tree('NP', [('the', 'DT')]), ('Galaxy', 'NNP'), ('Tab', 'NNP'), ('2', 'CD'),
('10.1', 'CD'), (',', ','), ('Galaxy', 'NNP'), ('Rugby', 'NNP'), ('Pro', 'NNP'),
('and', 'CC'), ('Galaxy', 'NNP'), ('S', 'NNP'), ('III', 'NNP'), Tree('NP',
[('mini', 'NN')]), ('.', '.')]), Tree('S', [('Apple', 'NNP'), Tree('VP',
[Tree('V', [('stated', 'VBD')])]), ('it', 'PRP'), Tree('VP', [Tree('V', [('had',
'VBD')])]), ('"', 'NNP'), Tree('VP', [Tree('V', [('acted', 'VBD')])]),
('quickly', 'RB'), ('and', 'CC'), ('diligently', 'RB'), ("'", "'"), Tree('PP',
[Tree('P', [('in', 'IN')]), Tree('NP', [('order', 'NN')])]), ('to', 'TO'),
('``', '``'), Tree('VP', [Tree('V', [('determine', 'VB')]), Tree('PP',
[Tree('P', [('that', 'IN')]), Tree('NP', [('these', 'DT')])])]), ('newly',
'RB'), Tree('VP', [Tree('V', [('released', 'VBN')])]), ('products', 'NNS'),
Tree('VP', [Tree('V', [('do', 'VBP')])]), Tree('VP', [Tree('V', [('infringe',
'VB')]), Tree('NP', [('many', 'JJ')]), Tree('PP', [Tree('P', [('of', 'IN')]),
Tree('NP', [('the', 'DT'), ('same', 'JJ')])])]), ('claims', 'NNS'), ('already',
'RB'), Tree('VP', [Tree('V', [('asserted', 'VBN')])]), Tree('P', [('by',
'IN')]), ('Apple', 'NNP'), ('.', '.'), ("'", "'")]), Tree('S', [Tree('P',
[('In', 'IN')]), ('August', 'NNP'), (',', ','), ('Samsung', 'NNP'), Tree('VP',
[Tree('V', [('lost', 'VBD')]), Tree('NP', [('a', 'DT')])]), ('US', 'NNP'),
Tree('NP', [('patent', 'NN'), ('case', 'NN')]), ('to', 'TO'), ('Apple', 'NNP'),
('and', 'CC'), Tree('VP', [Tree('V', [('was', 'VBD')])]), Tree('VP', [Tree('V',
[('ordered', 'VBN')])]), ('to', 'TO'), Tree('VP', [Tree('V', [('pay', 'VB')])]),
('its', 'PRP$'), Tree('NP', [('rival', 'JJ')]), ('$', '$'), ('1.05bn', 'CD'),
('(', '('), Tree('NP', [('£0.66bn', 'NN')]), (')', ')'), Tree('P', [('in',
'IN')]), ('damages', 'NNS'), Tree('P', [('for', 'IN')]), Tree('VP', [Tree('V',
[('copying', 'VBG')])]), ('features', 'NNS'), Tree('PP', [Tree('P', [('of',
'IN')]), Tree('NP', [('the', 'DT'), ('iPad', 'NN')])]), ('and', 'CC'),
Tree('NP', [('iPhone', 'NN')]), Tree('P', [('in', 'IN')]), ('its', 'PRP$'),
('Galaxy', 'NNP'), Tree('NP', [('range', 'NN')]), Tree('P', [('of', 'IN')]),
('devices', 'NNS'), ('.', '.')]), Tree('S', [('Samsung', 'NNP'), (',', ','),
('which', 'WDT'), Tree('VP', [Tree('V', [('is', 'VBZ')]), Tree('NP', [('the',
'DT'), ('world', 'NN')])]), ("'s", 'POS'), Tree('NP', [('top', 'JJ'), ('mobile',
'NN'), ('phone', 'NN'), ('maker', 'NN')]), (',', ','), Tree('VP', [Tree('V',
[('is', 'VBZ')])]), Tree('VP', [Tree('V', [('appealing', 'VBG')]), Tree('NP',
[('the', 'DT'), ('ruling', 'NN')])]), ('.', '.')]), Tree('S', [Tree('NP', [('A',
'DT'), ('similar', 'JJ'), ('case', 'NN')]), Tree('PP', [Tree('P', [('in',
'IN')]), Tree('NP', [('the', 'DT')])]), ('UK', 'NNP'), Tree('VP', [Tree('V',

```
[('found', 'VBD')])]), Tree('P', [('in', 'IN')]), ('Samsung', 'NNP'), ("'s",
'POS'), Tree('NP', [('favour', 'NN')]), ('and', 'CC'), Tree('VP', [Tree('V',
[('ordered', 'VBD')])]), ('Apple', 'NNP'), ('to', 'TO'), Tree('VP', [Tree('V',
[('publish', 'VB')]), Tree('NP', [('an', 'DT'), ('apology', 'NN')])]),
Tree('VP', [Tree('V', [('making', 'VBG')]), Tree('NP', [('clear', 'JJ')]),
Tree('PP', [Tree('P', [('that', 'IN')]), Tree('NP', [('the', 'DT'), ('South',
'JJ'), ('Korean', 'JJ'), ('firm', 'NN')])])]), Tree('VP', [Tree('V', [('had',
'VBD')])]), ('not', 'RB'), Tree('VP', [Tree('V', [('copied', 'VBN')])]), ('its',
'PRP$'), Tree('NP', [('iPad', 'NN')]), ('when', 'WRB'), Tree('VP', [Tree('V',
[('designing', 'VBG')])]), ('its', 'PRP$'), Tree('NP', [('own', 'JJ')]),
('devices', 'NNS'), ('.', '.')])]
```

1.4 [total points: 1] Exercise 2: spaCy

Use Spacy to process the same text as you analyzed with NLTK.

[22]:
```
import spacy
nlp = spacy.load('en_core_web_sm')
```

[23]:
```
#part-of-speech tagging
doc = nlp(text)
for token in doc:
    print(f'{token.text:15} {token.lemma_:15} {token.pos_:5} {token.tag_:5}␣
 ↪{token.dep_:7}')
#name entity recognition
for ent in doc.ents:
    print(f'{ent.text:15} {ent.start_char:5} {ent.end_char:5} {ent.label_:5}')
#dependency parsing
for token in doc:
    print(f"{token.text:10} {token.dep_:10} {token.head.text:10} {token.head.
 ↪pos_:5} {list(token.children)}")
#visualisation of dependency parse
#displacy.render(doc, style="dep", jupyter=True, options={'distance': 90})
```

```
https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-
more-products-under-scrutiny.html
https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-
more-products-under-scrutiny.html NOUN  NNS   amod
```

```
              SPACE _SP   dep
Documents       document        NOUN  NNS   nsubj
filed           file            VERB  VBD   ROOT
to              to              ADP   IN    prep
the             the             DET   DT    det
San             San             PROPN NNP   nmod
```

```
Jose            Jose            PROPN NNP   nmod
federal         federal         ADJ   JJ    amod
court           court           NOUN  NN    pobj
in              in              ADP   IN    prep
California      California      PROPN NNP   pobj
on              on              ADP   IN    prep
November        November        PROPN NNP   pobj
23              23              NUM   CD    nummod
list            list            NOUN  NN    compound
six             six             NUM   CD    nummod
Samsung         Samsung         PROPN NNP   compound
products        product         NOUN  NNS   dobj
running         run             VERB  VBG   acl
the             the             DET   DT    det
"               "               PUNCT ``    punct
Jelly           Jelly           PROPN NNP   compound
Bean            Bean            PROPN NNP   dobj
"               "               PUNCT ''    punct
and             and             CCONJ CC    cc
"               "               PUNCT ``    punct
Ice             Ice             PROPN NNP   compound
Cream           Cream           PROPN NNP   compound
Sandwich        sandwich        NOUN  NN    nmod
"               "               PUNCT ''    punct
operating       operating       NOUN  NN    compound
systems         system          NOUN  NNS   conj
,               ,               PUNCT ,     punct
which           which           PRON  WDT   nsubj
Apple           Apple           PROPN NNP   compound
claims          claim           VERB  VBZ   nsubj
infringe        infringe        VERB  VBP   relcl
its             its             PRON  PRP$  poss
patents         patent          NOUN  NNS   dobj
.               .               PUNCT .     punct


              SPACE _SP   dep
The             the             DET   DT    det
six             six             NUM   CD    nummod
phones          phone           NOUN  NNS   nsubj
and             and             CCONJ CC    cc
tablets         tablet          NOUN  NNS   conj
affected        affect          VERB  VBN   acl
are             be              AUX   VBP   ROOT
the             the             DET   DT    det
Galaxy          Galaxy          PROPN NNP   compound
S               S               PROPN NNP   compound
III             III             PROPN NNP   attr
```

```
,               ,               PUNCT ,     punct
running         run             VERB  VBG   advcl
the             the             DET   DT    det
new             new             ADJ   JJ    amod
Jelly           Jelly           PROPN NNP   compound
Bean            Bean            PROPN NNP   compound
system          system          NOUN  NN    dobj
,               ,               PUNCT ,     punct
the             the             DET   DT    det
Galaxy          Galaxy          PROPN NNP   compound
Tab             Tab             PROPN NNP   nmod
8.9             8.9             NUM   CD    nummod
Wifi            Wifi            PROPN NNP   compound
tablet          tablet          NOUN  NN    appos
,               ,               PUNCT ,     punct
the             the             DET   DT    det
Galaxy          Galaxy          PROPN NNP   compound
Tab             Tab             PROPN NNP   conj
2               2               NUM   CD    compound
10.1            10.1            NUM   CD    nummod
,               ,               PUNCT ,     punct
Galaxy          Galaxy          PROPN NNP   compound
Rugby           Rugby           PROPN NNP   compound
Pro             Pro             PROPN NNP   conj
and             and             CCONJ CC    cc
Galaxy          Galaxy          PROPN NNP   compound
S               S               PROPN NNP   compound
III             III             PROPN NNP   conj
mini            mini            NOUN  NN    appos
.               .               PUNCT .     punct


              SPACE _SP   dep
Apple           Apple           PROPN NNP   nsubj
stated          state           VERB  VBD   ROOT
it              it              PRON  PRP   nsubj
had             have            AUX   VBD   aux
"               "               PUNCT ``    punct
acted           act             VERB  VBN   ccomp
quickly         quickly         ADV   RB    advmod
and             and             CCONJ CC    cc
diligently      diligently      ADV   RB    conj
"               "               PUNCT ''    punct
in              in              ADP   IN    prep
order           order           NOUN  NN    pobj
to              to              PART  TO    aux
"               "               PUNCT ``    punct
determine       determine       VERB  VB    acl
```

```
that             that             SCONJ IN    mark
these            these            DET   DT    det
newly            newly            ADV   RB    advmod
released         release          VERB  VBN   amod
products         product          NOUN  NNS   nsubj
do               do               AUX   VBP   aux
infringe         infringe         VERB  VB    ccomp
many             many             ADJ   JJ    dobj
of               of               ADP   IN    prep
the              the              DET   DT    det
same             same             ADJ   JJ    amod
claims           claim            NOUN  NNS   pobj
already          already          ADV   RB    advmod
asserted         assert           VERB  VBN   acl
by               by               ADP   IN    agent
Apple            Apple            PROPN NNP   pobj
.                .                PUNCT .     punct
"                "                PUNCT ''    punct


                 SPACE _SP   dep
In               in               ADP   IN    prep
August           August           PROPN NNP   pobj
,                ,                PUNCT ,     punct
Samsung          Samsung          PROPN NNP   nsubj
lost             lose             VERB  VBD   ROOT
a                a                DET   DT    det
US               US               PROPN NNP   compound
patent           patent           NOUN  NN    compound
case             case             NOUN  NN    dobj
to               to               ADP   IN    prep
Apple            Apple            PROPN NNP   pobj
and              and              CCONJ CC    cc
was              be               AUX   VBD   auxpass
ordered          order            VERB  VBN   conj
to               to               PART  TO    aux
pay              pay              VERB  VB    xcomp
its              its              PRON  PRP$  poss
rival            rival            NOUN  NN    dative
$                $                SYM   $     nmod
1.05bn           1.05bn           NUM   CD    dobj
(                (                PUNCT -LRB- punct
£                £                SYM   $     nmod
0.66bn           0.66bn           NOUN  NN    appos
)                )                PUNCT -RRB- punct
in               in               ADP   IN    prep
damages          damage           NOUN  NNS   pobj
for              for              ADP   IN    prep
```

```
copying         copy            VERB  VBG   pcomp
features        feature         NOUN  NNS   dobj
of              of              ADP   IN    prep
the             the             DET   DT    det
iPad            iPad            PROPN NNP   pobj
and             and             CCONJ CC    cc
iPhone          iPhone          PROPN NNP   conj
in              in              ADP   IN    prep
its             its             PRON  PRP$  poss
Galaxy          Galaxy          PROPN NNP   compound
range           range           NOUN  NN    pobj
of              of              ADP   IN    prep
devices         device          NOUN  NNS   pobj
.               .               PUNCT .     punct
Samsung         Samsung         PROPN NNP   nsubj
,               ,               PUNCT ,     punct
which           which           PRON  WDT   nsubj
is              be              AUX   VBZ   relcl
the             the             DET   DT    det
world           world           NOUN  NN    poss
's              's              PART  POS   case
top             top             ADJ   JJ    amod
mobile          mobile          ADJ   JJ    amod
phone           phone           NOUN  NN    compound
maker           maker           NOUN  NN    attr
,               ,               PUNCT ,     punct
is              be              AUX   VBZ   aux
appealing       appeal          VERB  VBG   ROOT
the             the             DET   DT    det
ruling          ruling          NOUN  NN    dobj
.               .               PUNCT .     punct


                SPACE _SP   dep
A               a               DET   DT    det
similar         similar         ADJ   JJ    amod
case            case            NOUN  NN    nsubj
in              in              ADP   IN    prep
the             the             DET   DT    det
UK              UK              PROPN NNP   pobj
found           find            VERB  VBN   ROOT
in              in              ADP   IN    prep
Samsung         Samsung         PROPN NNP   poss
's              's              PART  POS   case
favour          favour          NOUN  NN    pobj
and             and             CCONJ CC    cc
ordered         order           VERB  VBD   conj
Apple           Apple           PROPN NNP   dobj
```

```
to              to              PART  TO    aux
publish         publish         VERB  VB    xcomp
an              an              DET   DT    det
apology         apology         NOUN  NN    dobj
making          make            VERB  VBG   acl
clear           clear           ADJ   JJ    acomp
that            that            SCONJ IN    mark
the             the             DET   DT    det
South           south           ADJ   JJ    amod
Korean          korean          ADJ   JJ    amod
firm            firm            NOUN  NN    nsubj
had             have            AUX   VBD   aux
not             not             PART  RB    neg
copied          copy            VERB  VBN   ccomp
its             its             PRON  PRP$  poss
iPad            iPad            PROPN NNP   dobj
when            when            SCONJ WRB   advmod
designing       design          VERB  VBG   advcl
its             its             PRON  PRP$  poss
own             own             ADJ   JJ    amod
devices         device          NOUN  NNS   dobj
.               .               PUNCT .     punct
https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-
more-products-under-scrutiny.html     0   112 TIME
San Jose          137   145 GPE
California        163   173 GPE
November 23       177   188 DATE
six               194   197 CARDINAL
Samsung           198   205 ORG
the "Jelly Bean   223   238 LAW
Apple             290   295 ORG
six               329   332 CARDINAL
the Galaxy S III   365   381 ORG
Jelly Bean        399   409 ORG
8.9               433   436 CARDINAL
2 10.1            465   471 DATE
Galaxy Rugby Pro   473   489 ORG
Galaxy S III      494   506 PERSON
Apple             513   518 ORG
Apple             678   683 ORG
August            689   695 DATE
Samsung           697   704 ORG
US                712   714 GPE
Apple             730   735 ORG
1.05bn            770   776 MONEY
0.66bn            779   785 MONEY
iPad              826   830 ORG
Galaxy            849   855 FAC
```

```
Samsung           874   881 ORG
UK                975   977 GPE
Samsung           987   994 ORG
Apple            1016  1021 ORG
South Korean     1066  1078 NORP
iPad             1103  1107 ORG
https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-
more-products-under-scrutiny.html amod       Documents  NOUN  [

]


         dep        https://www.telegraph.co.uk/technology/apple/9702716/Apple-
Samsung-lawsuit-six-more-products-under-scrutiny.html NOUN  []
Documents  nsubj      filed      VERB
[https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-
more-products-under-scrutiny.html]
filed      ROOT       filed      VERB  [Documents, to, on, products, .]
to         prep       filed      VERB  [court]
the        det        court      NOUN  []
San        nmod       Jose       PROPN []
Jose       nmod       court      NOUN  [San]
federal    amod       court      NOUN  []
court      pobj       to         ADP   [the, Jose, federal, in]
in         prep       court      NOUN  [California]
California pobj       in         ADP   []
on         prep       filed      VERB  [November]
November   pobj       on         ADP   [23]
23         nummod     November   PROPN []
list       compound   products   NOUN  []
six        nummod     products   NOUN  []
Samsung    compound   products   NOUN  []
products   dobj       filed      VERB  [list, six, Samsung, running]
running    acl        products   NOUN  [Bean]
the        det        Bean       PROPN []
"          punct      Bean       PROPN []
Jelly      compound   Bean       PROPN []
Bean       dobj       running    VERB  [the, ", Jelly, ", and, systems]
"          punct      Bean       PROPN []
and        cc         Bean       PROPN []
"          punct      Sandwich   NOUN  []
Ice        compound   Cream      PROPN []
Cream      compound   Sandwich   NOUN  [Ice]
Sandwich   nmod       systems    NOUN  [", Cream, "]
"          punct      Sandwich   NOUN  []
operating  compound   systems    NOUN  []
systems    conj       Bean       PROPN [Sandwich, operating, ,, infringe]
,          punct      systems    NOUN  []
```

```
which      nsubj      infringe   VERB  []
Apple      compound   claims     VERB  []
claims     nsubj      infringe   VERB  [Apple]
infringe   relcl      systems    NOUN  [which, claims, patents]
its        poss       patents    NOUN  []
patents    dobj       infringe   VERB  [its]
.          punct      filed      VERB  [
]

           dep        .          PUNCT []
The        det        phones     NOUN  []
six        nummod     phones     NOUN  []
phones     nsubj      are        AUX   [The, six, and, tablets]
and        cc         phones     NOUN  []
tablets    conj       phones     NOUN  [affected]
affected   acl        tablets    NOUN  []
are        ROOT       are        AUX   [phones, III, ,, running, .]
the        det        III        PROPN []
Galaxy     compound   III        PROPN []
S          compound   III        PROPN []
III        attr       are        AUX   [the, Galaxy, S]
,          punct      are        AUX   []
running    advcl      are        AUX   [system]
the        det        system     NOUN  []
new        amod       system     NOUN  []
Jelly      compound   Bean       PROPN []
Bean       compound   system     NOUN  [Jelly]
system     dobj       running    VERB  [the, new, Bean, ,, tablet]
,          punct      system     NOUN  []
the        det        tablet     NOUN  []
Galaxy     compound   tablet     NOUN  []
Tab        nmod       tablet     NOUN  []
8.9        nummod     tablet     NOUN  []
Wifi       compound   tablet     NOUN  []
tablet     appos      system     NOUN  [the, Galaxy, Tab, 8.9, Wifi, ,, Tab]
,          punct      tablet     NOUN  []
the        det        Tab        PROPN []
Galaxy     compound   Tab        PROPN []
Tab        conj       tablet     NOUN  [the, Galaxy, 10.1, ,, Pro]
2          compound   10.1       NUM   []
10.1       nummod     Tab        PROPN [2]
,          punct      Tab        PROPN []
Galaxy     compound   Pro        PROPN []
Rugby      compound   Pro        PROPN []
Pro        conj       Tab        PROPN [Galaxy, Rugby, and, III, mini]
and        cc         Pro        PROPN []
Galaxy     compound   III        PROPN []
S          compound   III        PROPN []
```

```
III         conj       Pro         PROPN [Galaxy, S]
mini        appos      Pro         PROPN []
.           punct      are         AUX   [
]

           dep         .           PUNCT []
Apple       nsubj      stated      VERB  []
stated      ROOT       stated      VERB  [Apple, acted, .]
it          nsubj      acted       VERB  []
had         aux        acted       VERB  []
"           punct      acted       VERB  []
acted       ccomp      stated      VERB  [it, had, ", quickly, ", in]
quickly     advmod     acted       VERB  [and, diligently]
and         cc         quickly     ADV   []
diligently  conj       quickly     ADV   []
"           punct      acted       VERB  []
in          prep       acted       VERB  [order]
order       pobj       in          ADP   [determine]
to          aux        determine   VERB  []
"           punct      determine   VERB  []
determine   acl        order       NOUN  [to, ", infringe]
that        mark       infringe    VERB  []
these       det        products    NOUN  []
newly       advmod     released    VERB  []
released    amod       products    NOUN  [newly]
products    nsubj      infringe    VERB  [these, released]
do          aux        infringe    VERB  []
infringe    ccomp      determine   VERB  [that, products, do, many]
many        dobj       infringe    VERB  [of]
of          prep       many        ADJ   [claims]
the         det        claims      NOUN  []
same        amod       claims      NOUN  []
claims      pobj       of          ADP   [the, same, asserted]
already     advmod     asserted    VERB  []
asserted    acl        claims      NOUN  [already, by]
by          agent      asserted    VERB  [Apple]
Apple       pobj       by          ADP   []
.           punct      stated      VERB  []
"           punct      lost        VERB  [
]

           dep         "           PUNCT []
In          prep       lost        VERB  [August]
August      pobj       In          ADP   []
,           punct      lost        VERB  []
Samsung     nsubj      lost        VERB  []
lost        ROOT       lost        VERB  [", In, ,, Samsung, case, to, and,
ordered, .]
```

```
a           det         case        NOUN  []
US          compound    case        NOUN  []
patent      compound    case        NOUN  []
case        dobj        lost        VERB  [a, US, patent]
to          prep        lost        VERB  [Apple]
Apple       pobj        to          ADP   []
and         cc          lost        VERB  []
was         auxpass     ordered     VERB  []
ordered     conj        lost        VERB  [was, pay]
to          aux         pay         VERB  []
pay         xcomp       ordered     VERB  [to, rival, 1.05bn, in]
its         poss        rival       NOUN  []
rival       dative      pay         VERB  [its]
$           nmod        1.05bn      NUM   []
1.05bn      dobj        pay         VERB  [$, (, 0.66bn, )]
(           punct       1.05bn      NUM   []
£           nmod        0.66bn      NOUN  []
0.66bn      appos       1.05bn      NUM   [£]
)           punct       1.05bn      NUM   []
in          prep        pay         VERB  [damages]
damages     pobj        in          ADP   [for]
for         prep        damages     NOUN  [copying]
copying     pcomp       for         ADP   [features, in]
features    dobj        copying     VERB  [of]
of          prep        features    NOUN  [iPad]
the         det         iPad        PROPN []
iPad        pobj        of          ADP   [the, and, iPhone]
and         cc          iPad        PROPN []
iPhone      conj        iPad        PROPN []
in          prep        copying     VERB  [range]
its         poss        range       NOUN  []
Galaxy      compound    range       NOUN  []
range       pobj        in          ADP   [its, Galaxy, of]
of          prep        range       NOUN  [devices]
devices     pobj        of          ADP   []
.           punct       lost        VERB  []
Samsung     nsubj       appealing   VERB  [,, is, ,]
,           punct       Samsung     PROPN []
which       nsubj       is          AUX   []
is          relcl       Samsung     PROPN [which, maker]
the         det         world       NOUN  []
world       poss        maker       NOUN  [the, 's]
's          case        world       NOUN  []
top         amod        maker       NOUN  []
mobile      amod        phone       NOUN  []
phone       compound    maker       NOUN  [mobile]
maker       attr        is          AUX   [world, top, phone]
,           punct       Samsung     PROPN []
```

```
is          aux         appealing   VERB  []
appealing   ROOT        appealing   VERB  [Samsung, is, ruling, .]
the         det         ruling      NOUN  []
ruling      dobj        appealing   VERB  [the]
.           punct       appealing   VERB  [
]

           dep         .           PUNCT []
A           det         case        NOUN  []
similar     amod        case        NOUN  []
case        nsubj       found       VERB  [A, similar, in]
in          prep        case        NOUN  [UK]
the         det         UK          PROPN []
UK          pobj        in          ADP   [the]
found       ROOT        found       VERB  [case, in, and, ordered, .]
in          prep        found       VERB  [favour]
Samsung     poss        favour      NOUN  ['s]
's          case        Samsung     PROPN []
favour      pobj        in          ADP   [Samsung]
and         cc          found       VERB  []
ordered     conj        found       VERB  [Apple, publish]
Apple       dobj        ordered     VERB  []
to          aux         publish     VERB  []
publish     xcomp       ordered     VERB  [to, apology]
an          det         apology     NOUN  []
apology     dobj        publish     VERB  [an, making]
making      acl         apology     NOUN  [clear, copied]
clear       acomp       making      VERB  []
that        mark        copied      VERB  []
the         det         firm        NOUN  []
South       amod        Korean      ADJ   []
Korean      amod        firm        NOUN  [South]
firm        nsubj       copied      VERB  [the, Korean]
had         aux         copied      VERB  []
not         neg         copied      VERB  []
copied      ccomp       making      VERB  [that, firm, had, not, iPad, designing]
its         poss        iPad        PROPN []
iPad        dobj        copied      VERB  [its]
when        advmod      designing   VERB  []
designing   advcl       copied      VERB  [when, devices]
its         poss        devices     NOUN  []
own         amod        devices     NOUN  []
devices     dobj        designing   VERB  [its, own]
.           punct       found       VERB  []
```

small tip: You can use **sents = list(doc.sents)** to be able to use the index to access a sentence like **sents[2]** for the third sentence.

[24]:
```
'''
sents = list(doc.sents)
# Access a sentence
specific_sentence = sents[2] #in this case the third one
print(specific_sentence.text)

# Part-of-Speech tagging for specific sentence
for token in specific_sentence:
    print(f'{token.text:15} {token.pos_:5}')

# Named Entity Recognition for specific sentence
for ent in specific_sentence.ents:
    print(f'{ent.text:15} {ent.label_:5}')

#dependency parse for specific sentence
displacy.render(specific_sentence, style="dep", jupyter=True,␣
 ↪options={'distance': 90})
'''
```

[24]:
```
'\nsents = list(doc.sents)\n# Access a sentence\nspecific_sentence = sents[2]
#in this case the third one \nprint(specific_sentence.text)\n\n# Part-of-Speech
tagging for specific sentence\nfor token in specific_sentence:\n
print(f\'{token.text:15} {token.pos_:5}\')\n\n# Named Entity Recognition for
specific sentence\nfor ent in specific_sentence.ents:\n
print(f\'{ent.text:15} {ent.label_:5}\')\n    \n#dependency parse for specific
sentence\ndisplacy.render(specific_sentence, style="dep", jupyter=True,
options={\'distance\': 90})\n'
```

1.5 [total points: 7] Exercise 3: Comparison NLTK and spaCy

We will now compare the output of NLTK and spaCy, i.e., in what do they differ?

1.5.1 [points: 3] Exercise 3a: Part of speech tagging

Compare the output from NLTK and spaCy regarding part of speech tagging.

- To compare, you probably would like to compare sentence per sentence. Describe if the sentence splitting is different for NLTK than for spaCy. If not, where do they differ?
- After checking the sentence splitting, select a sentence for which you expect interesting results and perhaps differences. Motivate your choice.
- Compare the output in `token.tag` from spaCy to the part of speech tagging from NLTK for each token in your selected sentence. Are there any differences? This is not a trick question; it is possible that there are no differences.

When we look at how sentences are divided, both NLTK and spaCy aim to break down text into individual sentences. However, they rely on distinct algorithms and models for this task, leading to some differences in their sentence-splitting behavior. Interestingly, the primary difference noted in the provided example doesn't lie in how sentences are split but in the way specific tokens, especially those involving currency symbols and amounts, are identified within a sentence.

"In August, Samsung lost a US patent case to Apple and was ordered to pay its rival $1.05bn (£0.66bn) in damages for copying features of the iPad and iPhone in its Galaxy range of devices."

We chose this sentence because it consists of difficult parts and overall has a difficult structure. The only difference in splitting that we could notice is '£0.66bn'.NLTK treats this as a single token, which makes sense since it represents a unified monetary amount. Conversely, spaCy breaks it down into two separate tokens: '£' and '0.66bn'. This action divides the currency symbol from its associated value. This distinction showcases the different approaches and tokenization rules that each library applies, with spaCy taking a more detailed route in breaking down tokens in this scenario.

When comparing spacy's token.tag_ output with NLTK's part-of-speech tagging for each word in our chosen sentence, we should keep in mind that the differences we notice stem from the unique tagging conventions and models each library uses. We noticed that the way "was" gets tagged could be a good exaple of how differently these tools can see language. SpaCy might label it as VBD (verb, past tense) or AUX (auxiliary verb), based on the context and spaCy model version. NLTK could also tag it as a past tense verb, but it might not always make a clear distinction for auxiliary verbs without specific settings.

1.5.2 [points: 2] Exercise 3b: Named Entity Recognition (NER)

- Describe differences between the output from NLTK and spaCy for Named Entity Recognition. Which one do you think performs better?

[25]:
```
text = """In August, Samsung lost a US patent case to Apple and was ordered to␣
 ↪pay its rival $1.05bn (£0.66bn) in damages for copying features of the iPad␣
 ↪and iPhone in its Galaxy range of devices."""
doc = nlp(text)
from spacy import displacy
displacy.render(doc, jupyter=True, style='ent')


sentences = nltk.sent_tokenize(text)
for sentence in sentences:

    tokens = nltk.word_tokenize(sentence)
    tokens_pos_tagged = nltk.pos_tag(tokens)
    tokens_pos_tagged_and_named_entities = ne_chunk(tokens_pos_tagged)
    print()
    print('ORIGINAL SENTENCE', sentence)
    print('NAMED ENTITY RECOGNITION OUTPUT',␣
 ↪tokens_pos_tagged_and_named_entities)


pos_tags_per_sentence = [] #stores POS tags for each sentence List[Tuple]
for tokens in tokens_per_sentence:
    pos_tags = nltk.pos_tag(tokens) #each tuple consists of a token and its␣
 ↪corresponding POS tag
    pos_tags_per_sentence.append(pos_tags)
```

```
    #print(pos_tags) #prints POS-tagged tokens for one sentence at a time
```

```
<IPython.core.display.HTML object>


ORIGINAL SENTENCE In August, Samsung lost a US patent case to Apple and was
ordered to pay its rival $1.05bn (£0.66bn) in damages for copying features of
the iPad and iPhone in its Galaxy range of devices.
NAMED ENTITY RECOGNITION OUTPUT (S
  In/IN
  (GPE August/NNP)
  ,/,
  (PERSON Samsung/NNP)
  lost/VBD
  a/DT
  (GSP US/NNP)
  patent/NN
  case/NN
  to/TO
  (GPE Apple/NNP)
  and/CC
  was/VBD
  ordered/VBN
  to/TO
  pay/VB
  its/PRP$
  rival/JJ
  $/$
  1.05bn/CD
  (/(
  £0.66bn/NN
  )/)
  in/IN
  damages/NNS
  for/IN
  copying/VBG
  features/NNS
  of/IN
  the/DT
  (ORGANIZATION iPad/NN)
  and/CC
  (ORGANIZATION iPhone/NN)
  in/IN
  its/PRP$
  (GPE Galaxy/NNP)
  range/NN
  of/IN
  devices/NNS
```

```
./.)
```

The differences between the output in the given sentence are:

Entities Identified: SpaCy correctly identifies more specific types of named entities, organizations ("Apple"), and monetary values ("$1.05bn", "£0.66bn"). NLTK, on the other hand, identifies fewer specific entity types.

Accuracy of Labels: SpaCy provides more accurate entity labels by assigning specific types such as DATE, ORG, GPE, and MONEY. Although it makes a mistake with labeling Galaxy device type and detects iPad as an organiztion. NLTK labels entities like "Samsung" as an Person, and "Apple" and "Galaxy" as an GPE(geo-political entities), which is less accurate and informative.

Overall, we wuld say that SpaCy performs better in Named Entity Recognition for the given sentence due to its ability to recognize a wider range of entity types and provide more accurate and informative labels.

1.5.3 [points: 2] Exercise 3c: Constituency/dependency parsing

Choose one sentence from the text and run constituency parsing using NLTK and dependency parsing using spaCy. * describe briefly the difference between constituency parsing and dependency parsing * describe differences between the output from NLTK and spaCy.

Dependency parsing and constituency parsing are two methods used to represent the structure of sentences in natural language processing.

- Using a hierarchical structure that is usually represented by a parse tree, **constituency parsing** attempts to dissect a sentence into its grammatical components or phrases.When using constituency parsing, the sentence's layered structure of phrases is reflected in the parse tree's structure, where each node denotes a component, and each edge denotes a grammatical link.Constituency parsing finds the noun, verb, prepositional, and other phrase constituents in a sentence (more language independent as it focuses on universal syntactic relationships) , as well as their hierarchical relationships.When comparing it to dependency parsing in terms of handling ambiguity, the former may resolve certain syntactic ambiguiguities, while the latter might face more challenges with structurals ambiguities (a sentence could be represented by multiple valid parse trees making it harder to discern the intended stucture).
- In **dependency parsing**, each word is viewed as a node, and the connections between them are shown as directed edges.Unlike constituency parsing, dependency parsing does not rely on phrasal constituents or sub-phrases. Rather, it represents the syntax of a sentence through relationships between words, specifically directed and typed edges within a graph. The goal of dependency parsing is to determine the links between words in a phrase.

In summary constituency parsing is a layered architecture of phrase-based constituents, while dependency parsing is a network of word-to-word interactions.

Difference in output between the two :

NLTK The NLTK output is demonstrated through tree structures and focuses on the categorization of syntax in the different parts of the sentences, such as the noun phrases and verb phrases, without emphesazing the named entities within the text. This structure is more aligned with showing sentence grammar and structure rather than identifying and categorizing named entites explicitly.

spaCy This output appears to be more straight forward when it comes to identifying named entities. SpaCy pairs words or phraases with their corresponding entity types such as 'organizations(ORG), geopolitical entities(GPE), DATE, MONEY etc'. It also provides a more detailed analysis on each words role in a sentnece, including the words relationship to other words in the sentence. In addition, it identifies named entities with specific labeles, which makes it easier to extract information about the people, organizations, locations and more.

Conclusion Overall, spaCy is generally perceived to perform better for Named Entity Recognition tasks, as opposed to NLTK. It identifies the entities more explicitly and categorizes them into predefined classes, which can be more useful when the user is trying to extract information, analyze data or enhance search algorithms. Thus, SpaCy's efficiency in processing and its ability to handle complex NER tasks with a higher degree of accuracy, makes it a preferred choice for NLP applications focused on named entity identification and categorization.

2 End of this notebook

[ ]: