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.

Credits

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

Tip: how to read a file from disk

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

```
from pathlib import Path

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))
```

```
C:\Users\coolm\OneDrive - Vrije Universiteit Amsterdam\University
documents\Third year\Text Mining\Assignments\ba-text-mining-master\ba-
text-mining-master\lab_sessions\lab1\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.

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

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

```
number of characters 1139
```

[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.

```
import nltk
from nltk.tokenize import sent_tokenize
from nltk import word_tokenize

sentences_nltk = sent_tokenize(text)

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.

```
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', '.']
```

[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!).

```
pos_tags_per_sentence = []
for tokens in tokens_per_sentence:
    tagged = nltk.pos_tag(tokens)
    pos_tags_per_sentence.append(tagged)
    print(tagged)
```

```
[('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'), ('.', '.')]
```

```
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'), ('.',
'.')]]
```

[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!).

```
from nltk.chunk import ne_chunk
ner_tags_per_sentence = []
for sentence in pos_tags_per_sentence:

    ner_tags_per_sentence.append(nltk.ne_chunk(sentence))

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'), ('.', '.')])]
```

[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!).

```
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)*''')

constituency_output_per_sentence = []
for sentence in pos_tags_per_sentence:

constituency_output_per_sentence.append(constituent_parser.parse(sente
nce))

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*

```
constituent_parser_v2 = nltk.RegexpParser('''

P: {<IN>}           # Preposition
```

```
V: {<V.*>}          # Verb
PP: {<P> <NP>}      # PP -> P NP
VP: {<V> <NP|PP>*}  # VP -> V (NP|PP)*
NEP: {<NNP>+ <CD|NNP|NN>*}
NP: {<DT>? <JJ>* <NN>*} # NP            # ???
''')

constituency_v2_output_per_sentence = []
for sentence in pos_tags_per_sentence:

constituency_v2_output_per_sentence.append(constituent_parser_v2.parse
(sentence))

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')]), Tree('NEP', [('San', 'NNP'), ('Jose',
'NNP')]), Tree('NP', [('federal', 'JJ'), ('court', 'NN')]), Tree('P',
[('in', 'IN')]), Tree('NEP', [('California', 'NNP')]), Tree('P',
[('on', 'IN')]), Tree('NEP', [('November', 'NNP'), ('23', 'CD'),
('list', 'NN'), ('six', 'CD'), ('Samsung', 'NNP')]), ('products',
'NNS'), Tree('VP', [Tree('V', [('running', 'VBG')])]), Tree('NP',
[('the', 'DT')]), ('``', '``'), ('Jelly', 'RB'), Tree('NEP', [('Bean',
'NNP')]), ("''", "''"), ('and', 'CC'), ('``', '``'), Tree('NEP',
[('Ice', 'NNP'), ('Cream', 'NNP'), ('Sandwich', 'NNP')]), ("''",
"''"), Tree('VP', [Tree('V', [('operating', 'VBG')])]), ('systems',
'NNS'), (',', ','), ('which', 'WDT'), Tree('NEP', [('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')]), Tree('NEP', [('Galaxy',
'NNP'), ('S', 'NNP'), ('III', 'NNP')]), (',', ','), Tree('VP',
[Tree('V', [('running', 'VBG')])]), Tree('NP', [('the', 'DT'), ('new',
'JJ')]), Tree('NEP', [('Jelly', 'NNP'), ('Bean', 'NNP'), ('system',
'NN')]), (',', ','), Tree('NP', [('the', 'DT')]), Tree('NEP',
[('Galaxy', 'NNP'), ('Tab', 'NNP'), ('8.9', 'CD'), ('Wifi', 'NNP'),
('tablet', 'NN')]), (',', ','), Tree('NP', [('the', 'DT')]),
Tree('NEP', [('Galaxy', 'NNP'), ('Tab', 'NNP'), ('2', 'CD'), ('10.1',
'CD')]), (',', ','), Tree('NEP', [('Galaxy', 'NNP'), ('Rugby', 'NNP'),
('Pro', 'NNP')]), ('and', 'CC'), Tree('NEP', [('Galaxy', 'NNP'), ('S',
'NNP'), ('III', 'NNP'), ('mini', 'NN')]), ('.', '.')]), Tree('S',
[Tree('NEP', [('Apple', 'NNP')]), Tree('VP', [Tree('V', [('stated',
'VBD')])]), ('it', 'PRP'), Tree('VP', [Tree('V', [('had', 'VBD')])]),
Tree('NEP', [('“', 'NNP')]), Tree('VP', [Tree('V', [('acted',
'VBD')])]), ('quickly', 'RB'), ('and', 'CC'), ('diligently', 'RB'),
```

```
("''", "''"), Tree('P', [('in', 'IN')]), Tree('NP', [('order',
'NN')]), ('to', 'TO'), ('``', '``'), Tree('VP', [Tree('V',
[('determine', 'VB')])]), 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('P', [('of', 'IN')]), Tree('NP',
[('the', 'DT'), ('same', 'JJ')]), ('claims', 'NNS'), ('already',
'RB'), Tree('VP', [Tree('V', [('asserted', 'VBN')])]), Tree('P',
[('by', 'IN')]), Tree('NEP', [('Apple', 'NNP')]), ('.', '.'), ("''",
"''")]), Tree('S', [Tree('P', [('In', 'IN')]), Tree('NEP', [('August',
'NNP')]), (',', ','), Tree('NEP', [('Samsung', 'NNP')]), Tree('VP',
[Tree('V', [('lost', 'VBD')])]), Tree('NP', [('a', 'DT')]),
Tree('NEP', [('US', 'NNP'), ('patent', 'NN'), ('case', 'NN')]), ('to',
'TO'), Tree('NEP', [('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$'), Tree('NP', [('rival', 'JJ')]), ('$', '$'), ('1.05bn',
'CD'), ('(', '('), Tree('NP', [('£0.66bn', 'NN')]), (')', ')'),
Tree('P', [('in', 'IN')]), ('damages', 'NNS'), Tree('P', [('for',
'IN')]), Tree('VP', [Tree('V', [('copying', 'VBG')])]), ('features',
'NNS'), Tree('P', [('of', 'IN')]), Tree('NP', [('the', 'DT'), ('iPad',
'NN')]), ('and', 'CC'), Tree('NP', [('iPhone', 'NN')]), Tree('P',
[('in', 'IN')]), ('its', 'PRP$'), Tree('NEP', [('Galaxy', 'NNP'),
('range', 'NN')]), Tree('P', [('of', 'IN')]), ('devices', 'NNS'),
('.', '.')]), Tree('S', [Tree('NEP', [('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('P', [('in', 'IN')]), Tree('NP', [('the', 'DT')]), Tree('NEP',
[('UK', 'NNP')]), Tree('VP', [Tree('V', [('found', 'VBD')])]),
Tree('P', [('in', 'IN')]), Tree('NEP', [('Samsung', 'NNP')]), ("'s",
'POS'), Tree('NP', [('favour', 'NN')]), ('and', 'CC'), Tree('VP',
[Tree('V', [('ordered', 'VBD')])]), Tree('NEP', [('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('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'), ('.', '.')])]
```

[total points: 1] Exercise 2: spaCy

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

```
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_sm')

doc = nlp(text) # insert code here
pos_tags_per_sentence = []
for sentence in doc.sents:
    pos_tags_per_sentence.append([(token.text, token.pos_) for token
in sentence])
print(pos_tags_per_sentence)

entities = [(ent.text, ent.label_) for ent in doc.ents]

ner_tags_per_sentence = []
for sentence in pos_tags_per_sentence:
    tokens = []
    for token in sentence:
        if any(t[0] == token[0] for t in entities):
            tokens.append((token[0], next((t[1] for t in entities if
t[0] == token[0]), None)))
        else:
            tokens.append(token[0])
    ner_tags_per_sentence.append(tokens)

print(ner_tags_per_sentence)

dependencies = {}
for sentence in doc.sents:
    for token in sentence:
        dependencies[token.head] = token.dep_

print(dependencies)

displacy.render(doc, jupyter=True, style='dep')


[[('https://www.telegraph.co.uk/technology/apple/9702716/Apple-
Samsung-lawsuit-six-more-products-under-scrutiny.html', 'PROPN'), ('\
n\n', 'SPACE'), ('Documents', 'PROPN'), ('filed', 'VERB'), ('to',
'ADP'), ('the', 'DET'), ('San', 'PROPN'), ('Jose', 'PROPN'),
('federal', 'ADJ'), ('court', 'NOUN'), ('in', 'ADP'), ('California',
'PROPN'), ('on', 'ADP'), ('November', 'PROPN'), ('23', 'NUM'),
('list', 'NOUN'), ('six', 'NUM'), ('Samsung', 'PROPN'), ('products',
'NOUN'), ('running', 'VERB'), ('the', 'DET'), ('"', 'PUNCT'),
('Jelly', 'PROPN'), ('Bean', 'PROPN'), ('"', 'PUNCT'), ('and',
'CCONJ'), ('"', 'PUNCT'), ('Ice', 'PROPN'), ('Cream', 'PROPN'),
```

```
('Sandwich', 'PROPN'), ('"', 'PUNCT'), ('operating', 'NOUN'),
('systems', 'NOUN'), (',', 'PUNCT'), ('which', 'PRON'), ('Apple',
'PROPN'), ('claims', 'VERB'), ('infringe', 'VERB'), ('its', 'PRON'),
('patents', 'NOUN'), ('.', 'PUNCT'), ('\n', 'SPACE')], [('The',
'DET'), ('six', 'NUM'), ('phones', 'NOUN'), ('and', 'CCONJ'),
('tablets', 'NOUN'), ('affected', 'VERB'), ('are', 'AUX'), ('the',
'DET'), ('Galaxy', 'PROPN'), ('S', 'PROPN'), ('III', 'PROPN'), (',',
'PUNCT'), ('running', 'VERB'), ('the', 'DET'), ('new', 'ADJ'),
('Jelly', 'PROPN'), ('Bean', 'PROPN'), ('system', 'NOUN'), (',',
'PUNCT'), ('the', 'DET'), ('Galaxy', 'PROPN'), ('Tab', 'PROPN'),
('8.9', 'NUM'), ('Wifi', 'PROPN'), ('tablet', 'NOUN'), (',', 'PUNCT'),
('the', 'DET'), ('Galaxy', 'PROPN'), ('Tab', 'PROPN'), ('2', 'NUM'),
('10.1', 'NUM'), (',', 'PUNCT'), ('Galaxy', 'PROPN'), ('Rugby',
'PROPN'), ('Pro', 'PROPN'), ('and', 'CCONJ'), ('Galaxy', 'PROPN'),
('S', 'PROPN'), ('III', 'PROPN'), ('mini', 'NOUN'), ('.', 'PUNCT'),
('\n', 'SPACE')], [('Apple', 'PROPN'), ('stated', 'VERB'), ('it',
'PRON'), ('had', 'AUX'), ('“', 'PUNCT'), ('acted', 'VERB'),
('quickly', 'ADV'), ('and', 'CCONJ'), ('diligently', 'ADV'), ('"',
'PUNCT'), ('in', 'ADP'), ('order', 'NOUN'), ('to', 'PART'), ('"',
'PUNCT'), ('determine', 'VERB'), ('that', 'SCONJ'), ('these', 'DET'),
('newly', 'ADV'), ('released', 'VERB'), ('products', 'NOUN'), ('do',
'AUX'), ('infringe', 'VERB'), ('many', 'ADJ'), ('of', 'ADP'), ('the',
'DET'), ('same', 'ADJ'), ('claims', 'NOUN'), ('already', 'ADV'),
('asserted', 'VERB'), ('by', 'ADP'), ('Apple', 'PROPN'), ('.',
'PUNCT')], [('"', 'PUNCT'), ('\n', 'SPACE'), ('In', 'ADP'), ('August',
'PROPN'), (',', 'PUNCT'), ('Samsung', 'PROPN'), ('lost', 'VERB'),
('a', 'DET'), ('US', 'PROPN'), ('patent', 'NOUN'), ('case', 'NOUN'),
('to', 'ADP'), ('Apple', 'PROPN'), ('and', 'CCONJ'), ('was', 'AUX'),
('ordered', 'VERB'), ('to', 'PART'), ('pay', 'VERB'), ('its', 'PRON'),
('rival', 'ADJ'), ('$', 'SYM'), ('1.05bn', 'NUM'), ('(', 'PUNCT'),
('£', 'SYM'), ('0.66bn', 'NUM'), (')', 'PUNCT'), ('in', 'ADP'),
('damages', 'NOUN'), ('for', 'ADP'), ('copying', 'VERB'), ('features',
'NOUN'), ('of', 'ADP'), ('the', 'DET'), ('iPad', 'PROPN'), ('and',
'CCONJ'), ('iPhone', 'PROPN'), ('in', 'ADP'), ('its', 'PRON'),
('Galaxy', 'PROPN'), ('range', 'NOUN'), ('of', 'ADP'), ('devices',
'NOUN'), ('.', 'PUNCT')], [('Samsung', 'PROPN'), (',', 'PUNCT'),
('which', 'PRON'), ('is', 'AUX'), ('the', 'DET'), ('world', 'NOUN'),
("'s", 'PART'), ('top', 'ADJ'), ('mobile', 'ADJ'), ('phone', 'NOUN'),
('maker', 'NOUN'), (',', 'PUNCT'), ('is', 'AUX'), ('appealing',
'VERB'), ('the', 'DET'), ('ruling', 'NOUN'), ('.', 'PUNCT'), ('\n',
'SPACE')], [('A', 'DET'), ('similar', 'ADJ'), ('case', 'NOUN'), ('in',
'ADP'), ('the', 'DET'), ('UK', 'PROPN'), ('found', 'VERB'), ('in',
'ADP'), ('Samsung', 'PROPN'), ("'s", 'PART'), ('favour', 'NOUN'),
('and', 'CCONJ'), ('ordered', 'VERB'), ('Apple', 'PROPN'), ('to',
'PART'), ('publish', 'VERB'), ('an', 'DET'), ('apology', 'NOUN'),
('making', 'NOUN'), ('clear', 'ADJ'), ('that', 'SCONJ'), ('the',
'DET'), ('South', 'ADJ'), ('Korean', 'ADJ'), ('firm', 'NOUN'), ('had',
'AUX'), ('not', 'PART'), ('copied', 'VERB'), ('its', 'PRON'), ('iPad',
'PROPN'), ('when', 'SCONJ'), ('designing', 'VERB'), ('its', 'PRON'),
```

```
('own', 'ADJ'), ('devices', 'NOUN'), ('.', 'PUNCT')]]
[['https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-
lawsuit-six-more-products-under-scrutiny.html', '\n\n', 'Documents',
'filed', 'to', 'the', 'San', 'Jose', 'federal', 'court', 'in',
('California', 'GPE'), 'on', 'November', '23', 'list', ('six',
'CARDINAL'), ('Samsung', 'ORG'), 'products', 'running', 'the', '"',
'Jelly', 'Bean', '"', 'and', '"', 'Ice', 'Cream', 'Sandwich', '"',
'operating', 'systems', ',', 'which', ('Apple', 'ORG'), 'claims',
'infringe', 'its', 'patents', '.', '\n'], ['The', ('six', 'CARDINAL'),
'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', '.', '\n'], [('Apple', 'ORG'), 'stated', 'it',
'had', '“', 'acted', 'quickly', 'and', 'diligently', '"', 'in',
'order', 'to', '"', 'determine', 'that', 'these', 'newly', 'released',
'products', 'do', 'infringe', 'many', 'of', 'the', 'same', 'claims',
'already', 'asserted', 'by', ('Apple', 'ORG'), '.'], ['"', '\n', 'In',
('August', 'DATE'), ',', ('Samsung', 'ORG'), 'lost', 'a', ('US',
'GPE'), 'patent', 'case', 'to', ('Apple', 'ORG'), 'and', 'was',
'ordered', 'to', 'pay', 'its', 'rival', '$', ('1.05bn', 'MONEY'), '(',
'£', ('0.66bn', 'MONEY'), ')', 'in', 'damages', 'for', 'copying',
'features', 'of', 'the', ('iPad', 'ORG'), 'and', ('iPhone', 'ORG'),
'in', 'its', 'Galaxy', 'range', 'of', 'devices', '.'], [('Samsung',
'ORG'), ',', 'which', 'is', 'the', 'world', "'s", 'top', 'mobile',
'phone', 'maker', ',', 'is', 'appealing', 'the', 'ruling', '.', '\n'],
['A', 'similar', 'case', 'in', 'the', ('UK', 'GPE'), 'found', 'in',
('Samsung', 'ORG'), "'s", 'favour', 'and', 'ordered', ('Apple',
'ORG'), 'to', 'publish', 'an', 'apology', 'making', 'clear', 'that',
'the', 'South', 'Korean', 'firm', 'had', 'not', 'copied', 'its',
('iPad', 'ORG'), 'when', 'designing', 'its', 'own', 'devices', '.']]
{Documents: 'compound',
https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-
lawsuit-six-more-products-under-scrutiny.html: 'dep', filed: 'punct',
court: 'prep', Jose: 'compound', to: 'pobj', in: 'pobj', on: 'pobj',
November: 'nummod', products: 'acl', list: 'appos', Bean: 'conj',
running: 'dobj', Sandwich: 'punct', Cream: 'compound', systems:
'relcl', claims: 'xcomp', patents: 'poss', infringe: 'dobj', .: 'dep',
phones: 'acl', are: 'punct', III: 'compound', system: 'compound',
Bean: 'compound', running: 'punct', tablet: 'appos', Tab: 'nummod',
Tab: 'appos', 10.1: 'nummod', Pro: 'conj', S: 'compound', III:
'compound', .: 'dep', stated: 'punct', acted: 'prep', quickly: 'conj',
in: 'pobj', determine: 'ccomp', order: 'acl', infringe: 'dobj',
products: 'amod', released: 'advmod', many: 'prep', claims: 'acl', of:
'pobj', asserted: 'agent', by: 'pobj', lost: 'punct', ": 'dep', In:
'pobj', case: 'compound', to: 'pobj', ordered: 'xcomp', pay: 'prep',
rival: 'appos', 1.05bn: 'punct', 0.66bn: 'nmod', in: 'pobj', damages:
'prep', for: 'pcomp', copying: 'prep', features: 'prep', iPad: 'conj',
of: 'pobj', range: 'prep', in: 'pobj', of: 'pobj', appealing: 'punct',
```

```
Samsung: 'punct', is: 'attr', world: 'case', maker: 'compound', phone:
'amod', ruling: 'det', .: 'dep', case: 'punct', UK: 'det', in: 'pobj',
found: 'conj', favour: 'poss', Samsung: 'case', in: 'pobj', ordered:
'ccomp', publish: 'dobj', making: 'amod', copied: 'advcl', firm:
'amod', Korean: 'amod', iPad: 'poss', designing: 'dobj', devices:
'amod'}

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

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.

[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?

[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?

NLTK and spaCy seem to split sentences in mostly the same way. A key difference to note is that spaCy uses tokens for whitespaces as well at times (e.g., ('\n\n', 'SPACE')), which implies that NLTK has a deeper focus on linguistic structure while spaCy preserves more of the formatting by comparison. Another point of contrast between the two is that NLTK seems to ignore punctuation characters (e.g.: (',', ',')), whereas spaCy tokenizes them as well (e.g.: ('.', 'PUNCT')). This suggests differences in the way that each method handles punctuation and/or special characters, leading further differences in the choice of implemented method for a given task. In other words, spaCy would be preferred in problems where the punctuation POS tag might also be significant, whereas NLTK would be preferred in cases where punctuation can mostly be discarded.

- After checking the sentence splitting, select a sentence for which you expect interesting results and perhaps differences. Motivate your choice.

It contains proper nouns (NNP vs. PROPN) like Galaxy S III and Jelly Bean, which may show differences. It includes common nouns (NNS vs. NOUN) like phones and tablets, useful for checking consistency. It has verbs (running), adjectives (new), and numbers (8.9, 10.1), which might be tagged differently.

The sentence considered for comparison is the following: "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". The motivation for this choice is primarily founded on the length and variety of POS tags within the sentence. As it contains both proper and common nouns, verbs, numbers, and adjectives, this diversity in its composition allows for much potential juxtaposition in each tokenization. It therefore also allows for an informal overall comparison of potential similarities and differences in the approach implemented by each toolkit.

- 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.

NLTK -- [('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'), ('.', '.')]

spaCy -- [('The', 'DET'), ('six', 'NUM'), ('phones', 'NOUN'), ('and', 'CCONJ'), ('tablets', 'NOUN'), ('affected', 'VERB'), ('are', 'AUX'), ('the', 'DET'), ('Galaxy', 'PROPN'), ('S', 'PROPN'), ('III', 'PROPN'), (',', 'PUNCT'), ('running', 'VERB'), ('the', 'DET'), ('new', 'ADJ'), ('Jelly', 'PROPN'), ('Bean', 'PROPN'), ('system', 'NOUN'), (',', 'PUNCT'), ('the', 'DET'), ('Galaxy', 'PROPN'), ('Tab', 'PROPN'), ('8.9', 'NUM'), ('Wifi', 'PROPN'), ('tablet', 'NOUN'), (',', 'PUNCT'), ('the', 'DET'), ('Galaxy', 'PROPN'), ('Tab', 'PROPN'), ('2', 'NUM'), ('10.1', 'NUM'), (',', 'PUNCT'), ('Galaxy', 'PROPN'), ('Rugby', 'PROPN'), ('Pro', 'PROPN'), ('and', 'CCONJ'), ('Galaxy', 'PROPN'), ('S', 'PROPN'), ('III', 'PROPN'), ('mini', 'NOUN'), ('.', 'PUNCT'), ('\n', 'SPACE')]

As can be seen from the respective tokenizations above, although different abbreviations are used for a given POS tag, the tokenization is almost identical. Ignoring differences in processing punctuation, the only differences observed in the two versions is found in the verb "are" towards the beginning of the sentece. NLTK classified this as a standard (present) verb, whereas spaCy classified it as an auxiliary verb. Both interpretations are technically correct, thus a preference for one method over the other would arise according to the context of the task in this case.

[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?

NLTK uses broader categories but tends to mislabel entities more often. For instance, it tags "Jelly Bean" and GPE and "Galaxy Rugby Pro" as PERSON. SpaCy tends to tag less, and as a result more accurately. NLTK also sometimes fails at recognizing values as MONEY (0.66bn, etc.) or DATE (August). Moreover, since NLTK relies on chunking, it sometimes groups entities incorrectly (as seen on the tree structures). Due to all these factors, spaCy seems to display an overall better performance at named entity recognition, with fewer mistakes and better recognition of phrase entities.

[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

Constituency parsing focuses on the formalism of context-free grammars, breaking sentences into (nested) phrase structures. It thus has a focus on phrase structures. Meanwhile, dependency parsing is based on grammatical relationships between sentence components. Based on these

relationships, it extracts dependencies to show how words relate each other, where each word depends on a head word. It thus has a focus on word relationships.

- describe differences between the output from NLTK and spaCy.

The sentence chosen was "Documents filed to the San Jose federal court in California on November 23 list six Samsung products running the "Jelly Bean" and "Ice Cream Sandwich" operating systems, which Apple claims infringe its patents."

NLTK breaks the sentence into hierarchical phrases, where as spaCy focuses on dependencies between head words and their associated words as aforementioned. A key difference is that NLTK breaks the sentence into nested units based on syntax. This can be seen in the way it groups "San Jose federal court" into a single noun phrase, or how it identifies "list six Samsung products running..." as a verb phrase. Meanwhile, spaCy focuses on direct grammatical relations between words. It assigns dependencies (e.g.: "nsubj" -- subject, "dobj" -- direct object), showing direct links between words rather than emphasizing a broader sentence structure.

End of this notebook