Group8_lab1

February 16, 2025

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.

[1]:
```
from pathlib import Path
```

[2]:
```
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/joanapetkova/Documents/VU AI/Text
Mining/TextMiningGroup8/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.

[3]:
```
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.

[4]:
```
# Imports
import nltk
from nltk.tokenize import sent_tokenize
from nltk import word_tokenize
```

[5]:
```
# Sentences splitting with NLTK
sentences_nltk = sent_tokenize(text)
```

[6]:
```
# Tokenization with NLTK
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.

[7]:
```
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!).

[8]:
```
# Part-of-speech tagging with NLTK
pos_tags_per_sentence = []

for tokens in tokens_per_sentence:
    pos_tags = nltk.pos_tag(tokens)
    pos_tags_per_sentence.append(pos_tags)
```

```
    print(pos_tags)
```

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

[9]:
```
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!).

[10]:
```
# Named entity recognition with NLTK
ner_tags_per_sentence = []

for pos_tags in pos_tags_per_sentence:
    ner_tags = nltk.ne_chunk(pos_tags)
    ner_tags_per_sentence.append(ner_tags)
    print(ner_tags)
```

```
(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
  (ORGANIZATION San/NNP Jose/NNP)
  federal/JJ
  court/NN
  in/IN
  (GPE California/NNP)
  on/IN
  November/NNP
  23/CD
  list/NN
  six/CD
  (ORGANIZATION Samsung/NNP)
  products/NNS
  running/VBG
  the/DT
  ``/``
  Jelly/RB
  (GPE Bean/NNP)
  ''/''
  and/CC
```

```
  ``/``
  Ice/NNP
  Cream/NNP
  Sandwich/NNP
  ''/''
  operating/VBG
  systems/NNS
  ,/,
  which/WDT
  (PERSON Apple/NNP)
  claims/VBZ
  infringe/VB
  its/PRP$
  patents/NNS
  ./.)
(S
  The/DT
  six/CD
  phones/NNS
  and/CC
  tablets/NNS
  affected/VBN
  are/VBP
  the/DT
  (ORGANIZATION Galaxy/NNP)
  S/NNP
  III/NNP
  ,/,
  running/VBG
  the/DT
  new/JJ
  (PERSON Jelly/NNP Bean/NNP)
  system/NN
  ,/,
  the/DT
  (ORGANIZATION Galaxy/NNP)
  Tab/NNP
  8.9/CD
  Wifi/NNP
  tablet/NN
  ,/,
  the/DT
  (ORGANIZATION Galaxy/NNP)
  Tab/NNP
  2/CD
  10.1/CD
  ,/,
  (PERSON Galaxy/NNP Rugby/NNP Pro/NNP)
```

```
  and/CC
  (PERSON Galaxy/NNP S/NNP)
  III/NNP
  mini/NN
  ./.)
(S
  (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
  (PERSON Apple/NNP)
  ./.
  ''/'')
(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
  ./.)
(S
  (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
  ./.)
(S
  A/DT
  similar/JJ
  case/NN
  in/IN
  the/DT
  (ORGANIZATION UK/NNP)
  found/VBD
  in/IN
  (GPE Samsung/NNP)
  's/POS
  favour/NN
  and/CC
  ordered/VBD
  (PERSON Apple/NNP)
  to/TO
  publish/VB
  an/DT
  apology/NN
  making/VBG
  clear/JJ
  that/IN
  the/DT
  (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
  ./.)
```

[11]:
```
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!).

[12]:
```
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)*''')
```

[13]:
```
# Constituency parsing with NLTK
constituency_output_per_sentence = []

for pos_tags in pos_tags_per_sentence:
    constituency_tree = constituent_parser.parse(pos_tags)
    constituency_output_per_sentence.append(constituency_tree)
    print(constituency_tree)
```

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

```
  ``/``
  Ice/NNP
  Cream/NNP
  Sandwich/NNP
  ''/''
  (VP (V operating/VBG))
  systems/NNS
  ,/,
  which/WDT
  Apple/NNP
  (VP (V claims/VBZ))
  (VP (V infringe/VB))
  its/PRP$
  patents/NNS
  ./.)
(S
  (NP The/DT)
  six/CD
  phones/NNS
  and/CC
  tablets/NNS
  (VP (V affected/VBN))
  (VP (V are/VBP) (NP the/DT))
  Galaxy/NNP
  S/NNP
  III/NNP
  ,/,
  (VP (V running/VBG) (NP the/DT new/JJ))
  Jelly/NNP
  Bean/NNP
  (NP system/NN)
  ,/,
  (NP the/DT)
  Galaxy/NNP
  Tab/NNP
  8.9/CD
  Wifi/NNP
  (NP tablet/NN)
  ,/,
  (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
  (NP mini/NN)
  ./.)
(S
  Apple/NNP
  (VP (V stated/VBD))
  it/PRP
  (VP (V had/VBD))
  "/NNP
  (VP (V acted/VBD))
  quickly/RB
  and/CC
  diligently/RB
  ''/''
  (PP (P in/IN) (NP order/NN))
  to/TO
  ``/``
  (VP (V determine/VB) (PP (P that/IN) (NP these/DT)))
  newly/RB
  (VP (V released/VBN))
  products/NNS
  (VP (V do/VBP))
  (VP
    (V infringe/VB)
    (NP many/JJ)
    (PP (P of/IN) (NP the/DT same/JJ)))
  claims/NNS
  already/RB
  (VP (V asserted/VBN))
  (P by/IN)
  Apple/NNP
  ./.
  ''/'')
(S
  (P In/IN)
  August/NNP
  ,/,
  Samsung/NNP
  (VP (V lost/VBD) (NP a/DT))
  US/NNP
  (NP patent/NN case/NN)
  to/TO
  Apple/NNP
  and/CC
  (VP (V was/VBD))
```

```
  (VP (V ordered/VBN))
  to/TO
  (VP (V pay/VB))
  its/PRP$
  (NP rival/JJ)
  $/$
  1.05bn/CD
  (/(
  (NP £0.66bn/NN)
  )/)
  (P in/IN)
  damages/NNS
  (P for/IN)
  (VP (V copying/VBG))
  features/NNS
  (PP (P of/IN) (NP the/DT iPad/NN))
  and/CC
  (NP iPhone/NN)
  (P in/IN)
  its/PRP$
  Galaxy/NNP
  (NP range/NN)
  (P of/IN)
  devices/NNS
  ./.)
(S
  Samsung/NNP
  ,/,
  which/WDT
  (VP (V is/VBZ) (NP the/DT world/NN))
  's/POS
  (NP top/JJ mobile/NN phone/NN maker/NN)
  ,/,
  (VP (V is/VBZ))
  (VP (V appealing/VBG) (NP the/DT ruling/NN))
  ./.)
(S
  (NP A/DT similar/JJ case/NN)
  (PP (P in/IN) (NP the/DT))
  UK/NNP
  (VP (V found/VBD))
  (P in/IN)
  Samsung/NNP
  's/POS
  (NP favour/NN)
  and/CC
  (VP (V ordered/VBD))
  Apple/NNP
```

```
  to/TO
  (VP (V publish/VB) (NP an/DT apology/NN))
  (VP
    (V making/VBG)
    (NP clear/JJ)
    (PP (P that/IN) (NP the/DT South/JJ Korean/JJ firm/NN)))
  (VP (V had/VBD))
  not/RB
  (VP (V copied/VBN))
  its/PRP$
  (NP iPad/NN)
  when/WRB
  (VP (V designing/VBG))
  its/PRP$
  (NP own/JJ)
  devices/NNS
  ./.)
```

[14]:
```
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*

[15]:
```
# Extended grammar for constituency parsing with NLTK
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: {<RB>? <NNP>+ <CD>* <NNP|NN>* <NP>?}    # NEP -> (RB)? (NNP)+ (CD)*␣
 ↪(NNP|NN)* (NP)?''')
# The inclusion of an <NP> position in the NEP pattern produces an unexpected␣
 ↪result (NEP November/NNP 23/CD (NP list/NN)),
# where "list" should not be part of the entity. This happens because NLTK␣
 ↪misclassifies "list" as a noun, instead of a verb,
# during POS tagging and then this noun is identified as a noun phrase and␣
 ↪therefore also included in the NEP.
# Another similar issue is with (NEP "/NNP), where " is also tagged as a NEP,␣
 ↪due to NLTK incorrectly classifying it as a
# proper noun (NNP) during POS tagging.
```

[16]:
```
# Improved constituency parsing with NLTK
constituency_v2_output_per_sentence = []

for pos_tags in pos_tags_per_sentence:
    constituency_tree_v2 = constituent_parser_v2.parse(pos_tags)
    constituency_v2_output_per_sentence.append(constituency_tree_v2)
    print(constituency_tree_v2)
```

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

```
  ``/``
  (NEP Ice/NNP Cream/NNP Sandwich/NNP)
  ''/''
  (VP (V operating/VBG))
  systems/NNS
  ,/,
  which/WDT
  (NEP Apple/NNP)
  (VP (V claims/VBZ))
  (VP (V infringe/VB))
  its/PRP$
  patents/NNS
  ./.)
(S
  The/DT
  six/CD
  phones/NNS
  and/CC
  tablets/NNS
  (VP (V affected/VBN))
  (VP (V are/VBP))
  the/DT
  (NEP Galaxy/NNP S/NNP III/NNP)
  ,/,
  (VP (V running/VBG))
  the/DT
  new/JJ
  (NEP Jelly/NNP Bean/NNP (NP system/NN))
  ,/,
  the/DT
  (NEP Galaxy/NNP Tab/NNP 8.9/CD Wifi/NNP (NP tablet/NN))
  ,/,
  the/DT
  (NEP Galaxy/NNP Tab/NNP 2/CD 10.1/CD)
  ,/,
  (NEP Galaxy/NNP Rugby/NNP Pro/NNP)
  and/CC
  (NEP Galaxy/NNP S/NNP III/NNP (NP mini/NN))
  ./.)
(S
  (NEP Apple/NNP)
  (VP (V stated/VBD))
  it/PRP
  (VP (V had/VBD))
  (NEP "/NNP)
  (VP (V acted/VBD))
  quickly/RB
  and/CC
```

```
  diligently/RB
  ''/''
  (PP (P in/IN) (NP order/NN))
  to/TO
  ``/``
  (VP (V determine/VB))
  (P that/IN)
  these/DT
  newly/RB
  (VP (V released/VBN))
  products/NNS
  (VP (V do/VBP))
  (VP (V infringe/VB))
  many/JJ
  (P of/IN)
  the/DT
  same/JJ
  claims/NNS
  already/RB
  (VP (V asserted/VBN))
  (P by/IN)
  (NEP Apple/NNP)
  ./.
  ''/'')
(S
  (P In/IN)
  (NEP August/NNP)
  ,/,
  (NEP Samsung/NNP)
  (VP (V lost/VBD))
  a/DT
  (NEP US/NNP (NP patent/NN case/NN))
  to/TO
  (NEP Apple/NNP)
  and/CC
  (VP (V was/VBD))
  (VP (V ordered/VBN))
  to/TO
  (VP (V pay/VB))
  its/PRP$
  rival/JJ
  $/$
  1.05bn/CD
  (/(
  (NP £0.66bn/NN)
  )/)
  (P in/IN)
  damages/NNS
```

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

```
its/PRP$
own/JJ
devices/NNS
./.)
```

[17]:
```
print(constituency_v2_output_per_sentence)
```

```
[Tree('S', [Tree('NP', [('https', 'NN')]), (':', ':'),
('//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'), ('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'),
Tree('NP', [('list', 'NN')])]), ('six', 'CD'), Tree('NEP', [('Samsung',
'NNP')]), ('products', 'NNS'), Tree('VP', [Tree('V', [('running', 'VBG')])]),
('the', 'DT'), ('``', '``'), Tree('NEP', [('Jelly', 'RB'), ('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', [('The', 'DT'), ('six', 'CD'), ('phones',
'NNS'), ('and', 'CC'), ('tablets', 'NNS'), Tree('VP', [Tree('V', [('affected',
'VBN')])]), Tree('VP', [Tree('V', [('are', 'VBP')])]), ('the', 'DT'),
Tree('NEP', [('Galaxy', 'NNP'), ('S', 'NNP'), ('III', 'NNP')]), (',', ','),
Tree('VP', [Tree('V', [('running', 'VBG')])]), ('the', 'DT'), ('new', 'JJ'),
Tree('NEP', [('Jelly', 'NNP'), ('Bean', 'NNP'), Tree('NP', [('system',
'NN')])]), (',', ','), ('the', 'DT'), Tree('NEP', [('Galaxy', 'NNP'), ('Tab',
'NNP'), ('8.9', 'CD'), ('Wifi', 'NNP'), Tree('NP', [('tablet', 'NN')])]), (',',
','), ('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'), Tree('NP', [('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('PP', [Tree('P', [('in',
'IN')]), Tree('NP', [('order', 'NN')])]), ('to', 'TO'), ('``', '``'), Tree('VP',
[Tree('V', [('determine', 'VB')])]), Tree('P', [('that', 'IN')]), ('these',
'DT'), ('newly', 'RB'), Tree('VP', [Tree('V', [('released', 'VBN')])]),
('products', 'NNS'), Tree('VP', [Tree('V', [('do', 'VBP')])]), Tree('VP',
[Tree('V', [('infringe', 'VB')])]), ('many', 'JJ'), Tree('P', [('of', 'IN')]),
('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')])]), ('a', 'DT'), Tree('NEP',
```

```
[('US', 'NNP'), Tree('NP', [('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$'), ('rival', 'JJ'),
('$', '$'), ('1.05bn', 'CD'), ('(', '('), Tree('NP', [('£0.66bn', 'NN')]), (')',
')'), Tree('P', [('in', 'IN')]), ('damages', 'NNS'), Tree('P', [('for', 'IN')]),
Tree('VP', [Tree('V', [('copying', 'VBG')])]), ('features', 'NNS'), Tree('PP',
[Tree('P', [('of', 'IN')]), Tree('NP', [('the', 'DT'), ('iPad', 'NN')])]),
('and', 'CC'), Tree('NP', [('iPhone', 'NN')]), Tree('P', [('in', 'IN')]),
('its', 'PRP$'), Tree('NEP', [('Galaxy', 'NNP'), Tree('NP', [('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')]), ('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')])]), ('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$'), Tree('NP', [('iPad', 'NN')]), ('when', 'WRB'),
Tree('VP', [Tree('V', [('designing', 'VBG')])]), ('its', 'PRP$'), ('own', 'JJ'),
('devices', 'NNS'), ('.', '.')])]
```

1.4 [total points: 1] Exercise 2: spaCy

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

[18]:
```
# Imports
import spacy
nlp = spacy.load('en_core_web_sm')
```

[19]:
```
doc = nlp(text)
sentences_spacy = list(doc.sents)

#Part-of-speech tagging with spaCy
tokens_tags = []
for sents in sentences_spacy:
    tokens_tags_per_sent = []
    for token in sents:
        token_tag = (token.text, token.tag_)
```

```
        tokens_tags_per_sent.append(token_tag)
    tokens_tags.append(tokens_tags_per_sent)
    print(tokens_tags_per_sent)
```

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

[20]:
```
# Named Entity Recognition with spaCy
entities = []
for sents in sentences_spacy:
    for ent in sents.ents:
        entity = (ent.text, ent.label_)
        entities.append(entity)
        print(entity)
```

```
('San Jose', 'GPE')
('California', 'GPE')
('November 23', 'DATE')
('six', 'CARDINAL')
('Samsung', 'ORG')
('Jelly Bean', 'WORK_OF_ART')
('Apple', 'ORG')
('six', 'CARDINAL')
('the Galaxy S III', 'PERSON')
('Jelly Bean', 'ORG')
('Galaxy Tab 8.9 Wifi', 'PERSON')
('Galaxy Tab 2 10.1', 'PERSON')
('Apple', 'ORG')
('Apple', 'ORG')
('August', 'DATE')
('Samsung', 'ORG')
('US', 'GPE')
('Apple', 'ORG')
('1.05bn', 'MONEY')
('0.66bn', 'MONEY')
('iPad', 'ORG')
('iPhone', 'ORG')
('Samsung', 'ORG')
('UK', 'GPE')
('Samsung', 'ORG')
('Apple', 'ORG')
('South Korean', 'NORP')
('iPad', 'ORG')
```

[ ]:
```
# Dependency parsing with spaCy
from spacy import displacy

i = 0
```

```
for sent in sentences_spacy:
    displacy.render(sent, jupyter=True, style='dep') # render the dependency␣
 ↪tree in the notebook

    # uncomment to also save the dependency tree as a file
    # tree_structure = displacy.render(sent, jupyter=False, style='dep')
    # output_path = f'spaCy_sentence{i}_dependency_parsing.svg'
    # i += 1
    # with open(output_path, 'w') as outfile:
    #     outfile.write(tree_structure)
```

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

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

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

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

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

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

ATTENTION!!! The PDF export of the Jupyter notebook does not work when we include SVG files, hence, we have included the images produced by the code above in the zip along with the Jupyter notebook and its pdf.

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.

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?

Both *NLTK* and *spaCy* produce nearly identical sentence splittings for the given text, correctly dividing it into sentences, with almost no differences in how they identify where one sentence ends and another begins. The only difference we noticed is between the third and fourth sentences. While *NLTK* correctly considers the quotations marks at the end of third sentence to be part of third sentence, *spaCy* identifies them as the beginning of fourth sentence. More noticeable differences between the approaches of the two libraries can be observed at the tokenization level, where sentences are split into different subparts. One noticable example of such a difference is the way URLs are handled, with *NLTK* splitting them into multiple tokens, while *spaCy* preserves them in a single token, or line breaks handling (*NLTK* skips them, while *spaCy* tokenizes them with `_SP`). Nonetheless, for overall sentence segmentation, they perform consistently with each other.

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

We chose to compare the outputs of both *NLTK* and *spaCy* on sentence 4 (at index 3): > *"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."*

The main reason is that this sentence contains a variety of parts of speech (e.g, singular proper nouns, possesive pronouns, gerund verbs, past tense verbs, past participle verbs) that make it an interesting case for comparison. We also wanted to observe how the two approaches differ in handling the presence of numeric values associated with currencies (e.g., £0.66bn).

[22]:
```
print("\nNLTK:")
print(pos_tags_per_sentence[3])
print("\nspaCy:")
print(tokens_tags[3])
```

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

spaCy:
[('"', "'"), ('\n', '_SP'), ('In', 'IN'), ('August', 'NNP'), (',', ','),
('Samsung', 'NNP'), ('lost', 'VBD'), ('a', 'DT'), ('US', 'NNP'), ('patent',
'NN'), ('case', 'NN'), ('to', 'IN'), ('Apple', 'NNP'), ('and', 'CC'), ('was',
'VBD'), ('ordered', 'VBN'), ('to', 'TO'), ('pay', 'VB'), ('its', 'PRP$'),
('rival', 'JJ'), ('$', '$'), ('1.05bn', 'CD'), ('(', '-LRB-'), ('£', '$'),
('0.66bn', 'CD'), (')', '-RRB-'), ('in', 'IN'), ('damages', 'NNS'), ('for',
'IN'), ('copying', 'VBG'), ('features', 'NNS'), ('of', 'IN'), ('the', 'DT'),
('iPad', 'NNP'), ('and', 'CC'), ('iPhone', 'NNP'), ('in', 'IN'), ('its',
'PRP$'), ('Galaxy', 'NNP'), ('range', 'NN'), ('of', 'IN'), ('devices', 'NNS'),
('.', '.')]
```

- 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.

Upon comparing the tokens and their assigned tags by both NLTK and spaCy in the sentence, we found that most tags were consistent, showing good agreement on parts of speech. However, a few key differences were observed: * *spaCy* uses special token `_SP` for line breaks, whereas *NLTK* skips or ignores line breaks during tokenization. * *NLTK* correctly tagged the first "to" preposition as `TO` (seperate tag reserved for the word "to"), while *spaCy* tagged it as `IN` (general tag for prepositions).

* Parentheses in *NLTK* are tagged with simple punctuation labels, while *spaCy* uses `-LRB-` (left round bracket) and `-RRB-` (right round bracket). * *NLTK* tagged the currency "£0.66bn" as a noun (`NN`), whereas *spaCy* tagged the currency symbol £ as `$` (currency), and the number 0.66bn as `CD` (cardinal number). This misalignment does not occur when the currency is $, in which case both *NLTK* and *spaCy* tag it correctly as a currency and a cardinal number. * "iPad" and "iPhone" are incorrectly tagged by *NLTK* as `NN` (common noun), while *spaCy* correctly tags them as proper nouns `NNP` since they are product names.

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?

[23]:
```
# Named entities produced by NLTK
print("NLTK:\n")

for ner_tree in ner_tags_per_sentence:
    for subtree in ner_tree:
        if hasattr(subtree, 'label'):
            entity_name = ""
            for token, pos in subtree.leaves():
                entity_name += f"{token} "

            entity_type = subtree.label()
            print(f"{entity_name.strip()}: {entity_type}")
```

```
NLTK:

San Jose: ORGANIZATION
California: GPE
Samsung: ORGANIZATION
Bean: GPE
Apple: PERSON
Galaxy: ORGANIZATION
Jelly Bean: PERSON
Galaxy: ORGANIZATION
Galaxy: ORGANIZATION
Galaxy Rugby Pro: PERSON
Galaxy S: PERSON
Apple: PERSON
Apple: PERSON
August: GPE
Samsung: PERSON
US: GSP
Apple: GPE
iPad: ORGANIZATION
iPhone: ORGANIZATION
Galaxy: GPE
```

```
Samsung: GPE
UK: ORGANIZATION
Samsung: GPE
Apple: PERSON
South Korean: LOCATION
```

[24]:
```python
# Named entities produced by spaCy
print("spaCy:\n")
for entity in entities:
    print(f"{entity[0]}: {entity[1]}")
```

```
spaCy:

San Jose: GPE
California: GPE
November 23: DATE
six: CARDINAL
Samsung: ORG
Jelly Bean: WORK_OF_ART
Apple: ORG
six: CARDINAL
the Galaxy S III: PERSON
Jelly Bean: ORG
Galaxy Tab 8.9 Wifi: PERSON
Galaxy Tab 2 10.1: PERSON
Apple: ORG
Apple: ORG
August: DATE
Samsung: ORG
US: GPE
Apple: ORG
1.05bn: MONEY
0.66bn: MONEY
iPad: ORG
iPhone: ORG
Samsung: ORG
UK: GPE
Samsung: ORG
Apple: ORG
South Korean: NORP
iPad: ORG
```

NLTK and spaCy exhibit many differences in how they process text and identify entities. The primary differences that we noticed are the following: - *NLTK* fails to recognize the two dates present in the text, while *spaCY* correctly labels both of them. More precisely, *NLTK* does not identify "November 23" as an entity at all and misclassifies "August" as a GPE (geopolitical entity), while *spaCy* identifies both as DATE entities. - *NLTK* fails to identify the financial values ($1.05bn and £0.66bn) present in the text as entities, while *spaCy* labels both of them correctly as MONEY entities. - *NLTK* misclassifies the companies Apple and Samsung, often labeling them

inconsistently as PERSON, ORGANIZATION, or GPE, while *spaCy* shows more consistency in classifying these entities, correctly identifying Apple and Samsung as ORG (organization) entities. - *NLTK* misclassifies "UK" as an ORGANIZATION entity, while *spaCy* correctly identifies it as a GPE (geopolitical entity). - *spaCy* has a seperate category NORP (Nationalities or Religious/Political groups) for entities that express belonging to a certain country like "South Korean", while *NLTK* only has the general label LOCATION that it uses to refer to such entities. - Both *NLTK* and *spaCy* misclassify products, such as the Galaxy Rugby Pro, iPad and iPhone, as PERSON or ORG/ORGANIZATION. However, *spaCy* is able to identify the full names of most of the products, while *NLTK* has problems doing so (e.g. "Galaxy Tab 2 10.1" vs. "Galaxy") **without a manually defined grammar for named entity phrases (see exercise 1c)**. - *spaCy* identifies the number "six" in the text as a CARDINAL (cardinal number) entity, while *NLTK* does not recognize it as an entity at all.

Overall, we believe that *spaCy* is better than *NLTK* due to its greater accuracy and consistency and more detailed entity list (e.g., `WORK_OF_ART`, `NORP`). However, both approaches seem to have their weaknesses and might exhibit variable performance depending on the text type.

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

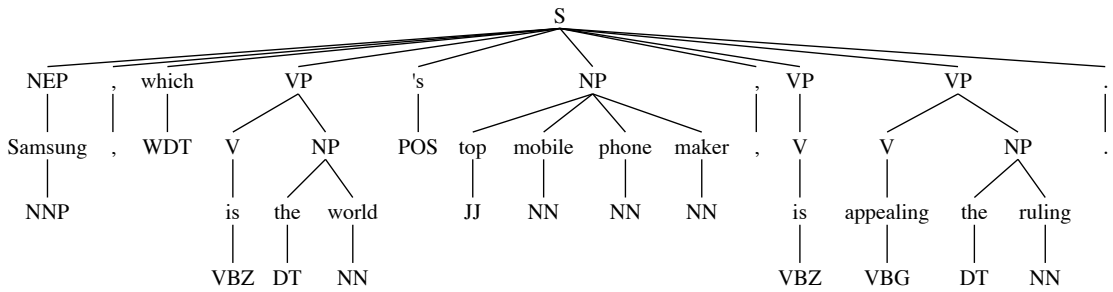
Choose one sentence from the text and run constituency parsing using NLTK and dependency parsing using spaCy. * describe briefly the difference between constituency parsing and dependency parsing * describe differences between the output from NLTK and spaCy.

We use the sentence below as an example:

"Samsung, which is the world's top mobile phone maker, is appealing the ruling."

[ ]:
```
# Constituency parsing produced by NLTK
constituency_v2_output_per_sentence[4]
```

[ ]:



[ ]:
```
# Dependency parsing produced by spaCy
displacy.render(sentences_spacy[4], jupyter=True, style='dep')
```

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

ATTENTION!!! The PDF export of the Jupyter notebook does not work when we include SVG files, hence, we have included the image produced by the code above (spaCy_sentence4_dependency_parsing.svg) in the zip along with the Jupyter notebook and its pdf.

1.5.4 Difference between constituency parsing and dependency parsing

- **Constituency parsing** divides a sentence into nested phrases (NP for noun phrase, VP for verb phrase). It shows hierarchical groupings such as `(NP top/JJ mobile/NN phone/NN maker/NN)` to illustrate how words build up to larger units (subjects, objects, and etc).
- **Dependency parsing** focuses on direct word-to-word relations via arrows or links. Each word is a "node," and edges label grammatical functions: for instance, "Samsung" might be the subject (`nsubj`), "appealing" the verb (`ROOT`), and "the ruling" the object (`dobj`). It does not explicitly chunk words into NPs or VPs, but it shows direct "who modifies whom" relationships.

1.5.5 Differences in the output from NLTK and spaCy

- **NLTK (Constituency)**: In the RegexpParser output, the sentence might appear bracketed like:

```
(S
  (NP Samsung/NNP)
  ,/,
  which/WDT
  (VP (V is/VBZ) (NP the/DT world/NN))
  's/POS
  (NP top/JJ mobile/NN phone/NN maker/NN)
  ,/,
  (VP (V is/VBZ))
  (VP (V appealing/VBG) (NP the/DT ruling/NN))
  ./.)
```

  This structure can also be visualized as a tree, as seen in the output from the first code cell above. This hierarchical nested structure, where the sentence (S) represents the root, emphasizes how words combine into labeled phrases, such as noun phrases `(NP ...)` and verb phrases `(VP ...)`.

- **spaCy (Dependency):** The `displacy.render` visualization creates a graph with arrows from each "head" word to its "dependent" words, representing their grammatical dependency. You would see labels like `nsubj` from "Samsung" to the verb "appealing," or `det` from "ruling" to "the." Instead of large nested phrases, spaCy shows direct grammatical functions (subjects, objects, modifiers) among individual words. A key difference from the NLTK constituency tree is that spaCy identifies a single root word for each sentence. In this case, the root is the main verb "appealing", with all other words being directly or indirectly dependent on it.

2 End of this notebook