Module 2 (Python 3)

Basic NLP Tasks with NLTK ¶

```
In [1]:  import nltk
         from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Counting vocabulary of words

```
In [4]:  text7
```

```
Out[4]:  <Text: Wall Street Journal>
```

```
In [5]:  sent7
```

```
Out[5]:  ['Pierre',
          'Vinken',
          ',',
          '61',
          'years',
          'old',
          ',',
          'will',
          'join',
          'the',
          'board',
          'as',
          'a',
          'nonexecutive',
          'director',
          'Nov.',
          '29',
          '.']
```

```
In [6]:  len(sent7)
```

```
Out[6]:  18
```

```
In [7]: len(text7)

Out[7]: 100676


In [8]: len(set(text7))

Out[8]: 12408


In [10]: list(set(text7))[:10]

Out[10]: ['school-board',
         'Exports',
         'disorders',
         'chocolate',
         'guilty',
         'takeover',
         'Kawasaki',
         'youth',
         '*T*-249',
         'Larry']
```

Frequency of words

```
In [11]: dist = FreqDist(text7)
         len(dist)

Out[11]: 12408


In [12]: vocab1 = dist.keys()
         #vocab1[:10]
         # In Python 3 dict.keys() returns an iterable view instead of a l
         ist
         list(vocab1)[:10]

Out[12]: ['Pierre', 'Vinken', ',', '61', 'years', 'old', 'will', 'join',
         'the', 'board']


In [13]: dist['four']

Out[13]: 20


In [17]: freqwords = [w for w in vocab1 if len(w) > 5 and dist[w] > 100]
         freqwords

Out[17]: ['billion',
         'company',
         'president',
         'because',
         'market',
         'million',
         'shares',
         'trading',
         'program']
```

Normalization and stemming

```
In [22]: input1 = "List listed lists listing listings"
         words1 = input1.lower().split(' ')
         words1
```

```
Out[22]: ['list', 'listed', 'lists', 'listing', 'listings']
```

```
In [23]: porter = nltk.PorterStemmer()
         [porter.stem(t) for t in words1]
```

```
Out[23]: ['list', 'list', 'list', 'list', 'list']
```

Lemmatization

```
In [26]: udhr = nltk.corpus.udhr.words('English-Latin1')
         udhr[:20]
```

```
Out[26]: ['Universal',
          'Declaration',
          'of',
          'Human',
          'Rights',
          'Preamble',
          'Whereas',
          'recognition',
          'of',
          'the',
          'inherent',
          'dignity',
          'and',
          'of',
          'the',
          'equal',
          'and',
          'inalienable',
          'rights',
          'of']
```

```
In [24]: [porter.stem(t) for t in udhr[:20]] # Still Lemmatization
```

```
Out[24]: ['univers',
          'declar',
          'of',
          'human',
          'right',
          'preambl',
          'wherea',
          'recognit',
          'of',
          'the',
          'inher',
          'digniti',
          'and',
          'of',
          'the',
```

```
          'equal',
          'and',
          'inalien',
          'right',
          'of']
```

In [25]:
```
WNlemma = nltk.WordNetLemmatizer()
[WNlemma.lemmatize(t) for t in udhr[:20]]
```

Out[25]:
```
['Universal',
 'Declaration',
 'of',
 'Human',
 'Rights',
 'Preamble',
 'Whereas',
 'recognition',
 'of',
 'the',
 'inherent',
 'dignity',
 'and',
 'of',
 'the',
 'equal',
 'and',
 'inalienable',
 'right',
 'of']
```

Tokenization

In [28]:
```
text11 = "Children shouldn't drink a sugary drink before bed."
text11.split(' ')
```

Out[28]:
```
['Children', "shouldn't", 'drink', 'a', 'sugary', 'drink', 'before', 'bed.']
```

In [29]:
```
nltk.word_tokenize(text11)
```

Out[29]:
```
['Children',
 'should',
 "n't",
 'drink',
 'a',
 'sugary',
 'drink',
 'before',
 'bed',
 '.']
```

In [30]:
```
text12 = "This is the first sentence. A gallon of milk in the U.
S. costs $2.99. Is this the third sentence? Yes, it is!"
sentences = nltk.sent_tokenize(text12)
len(sentences)
```

Out[30]: 4

```
In [31]:   sentences
```

```
Out[31]:   ['This is the first sentence.',
            'A gallon of milk in the U.S. costs $2.99.',
            'Is this the third sentence?',
            'Yes, it is!']
```

Advanced NLP Tasks with NLTK

POS tagging

```
In [33]:   nltk.help.upenn_tagset('MD')
```

```
           MD: modal auxiliary
               can cannot could couldn't dare may might must need ought shal
           l should
               shouldn't will would
```

```
In [34]:   text13 = nltk.word_tokenize(text11)
           nltk.pos_tag(text13)
```

```
Out[34]:   [('Children', 'NNP'),
            ('should', 'MD'),
            ("n't", 'RB'),
            ('drink', 'VB'),
            ('a', 'DT'),
            ('sugary', 'JJ'),
            ('drink', 'NN'),
            ('before', 'IN'),
            ('bed', 'NN'),
            ('.', '.')]
```

```
In [35]:   text14 = nltk.word_tokenize("Visiting aunts can be a nuisance")
           nltk.pos_tag(text14)
```

```
Out[35]:   [('Visiting', 'VBG'),
            ('aunts', 'NNS'),
            ('can', 'MD'),
            ('be', 'VB'),
            ('a', 'DT'),
            ('nuisance', 'NN')]
```

```
In [37]:   # Parsing sentence structure
           text15 = nltk.word_tokenize("Alice loves Bob")
           grammar = nltk.CFG.fromstring("""
           S -> NP VP
           VP -> V NP
           NP -> 'Alice' | 'Bob'
           V -> 'loves'
           """)

           parser = nltk.ChartParser(grammar)
           trees = parser.parse_all(text15)
           for tree in trees:
```

```
for tree in trees:
    print(tree)
```

```
(S (NP Alice) (VP (V loves) (NP Bob)))
```

In [40]:
```
text16 = nltk.word_tokenize("I saw the man with a telescope")
grammar1 = nltk.data.load('mygrammar.cfg')
grammar1
```

Out[40]: `<Grammar with 13 productions>`

In [41]:
```
parser = nltk.ChartParser(grammar1)
trees = parser.parse_all(text16)
for tree in trees:
    print(tree)
```

```
(S
  (NP I)
  (VP
    (VP (V saw) (NP (Det the) (N man)))
    (PP (P with) (NP (Det a) (N telescope)))))
(S
  (NP I)
  (VP
    (V saw)
    (NP (Det the) (N man) (PP (P with) (NP (Det a) (N telescop
e))))))
```

In [42]:
```
from nltk.corpus import treebank
text17 = treebank.parsed_sents('wsj_0001.mrg')[0]
print(text17)
```

```
(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
    (ADJP (NP (CD 61) (NNS years)) (JJ old))
    (, ,))
  (VP
    (MD will)
    (VP
      (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN directo
r)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
```

POS tagging and parsing ambiguity