Module 2 (Python 3)

Basic NLP Tasks with NLTK

In [3]:
```
import nltk
from nltk.book import *
```

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 [9]:
```
list(set(text7))[:10]
```

Out[9]:
```
['lately',
 'Have',
 'agrees',
 'arched',
 'reaping',
 'judged',
 'Express',
 'bedding',
 'homework',
 'tricky']
```

Frequency of words

In [10]:
```
dist = FreqDist(text7)
len(dist)
```

Out[10]:
```
12408
```

In [11]:
```
vocab1 = dist.keys()
#vocab1[:10]
# In Python 3 dict.keys() returns an iterable view instead of a list
list(vocab1)[:10]
```

Out[11]:
```
['Pierre', 'Vinken', ',', '61', 'years', 'old', 'will', 'join', 'the', 'board']
```

In [12]:
```
dist['four']
```

Out[12]:
```
20
```

In [13]:
```
freqwords = [w for w in vocab1 if len(w) > 5 and dist[w] > 100]
freqwords
```

Out[13]:
```
['billion',
 'company',
 'president',
 'because',
 'market',
 'million',
 'shares',
 'trading',
 'program']
```

Normalization and stemming

In [14]:
```
input1 = "List listed lists listing listings"
words1 = input1.lower().split(' ')
words1
```

Out[14]:
```
['list', 'listed', 'lists', 'listing', 'listings']
```

In [15]:
```
porter = nltk.PorterStemmer()
[porter.stem(t) for t in words1]
```

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

Lemmatization

In [16]:
```
udhr = nltk.corpus.udhr.words('English-Latin1')
udhr[:20]
```

Out[16]:
```
['Universal',
 'Declaration',
 'of',
 'Human',
 'Rights',
 'Preamble',
 'Whereas',
 'recognition',
 'of',
 'the',
 'inherent',
 'dignity',
 'and',
 'of',
 'the',
 'equal',
 'and',
 'inalienable',
 'rights',
 'of']
```

In [17]:
```
[porter.stem(t) for t in udhr[:20]] # Still Lemmatization
```

Out[17]:
```
['univers',
 'declar',
 'of',
 'human',
 'right',
 'preambl',
 'wherea',
 'recognit',
 'of',
 'the',
 'inher',
 'digniti',
 'and',
 'of',
 'the',
 'equal',
 'and',
 'inalien',
 'right',
 'of']
```

In [18]:
```
WNlemma = nltk.WordNetLemmatizer()
[WNlemma.lemmatize(t) for t in udhr[:20]]
```

Out[18]:
```
['Universal',
 'Declaration',
 'of',
 'Human',
 'Rights',
 'Preamble',
 'Whereas',
 'recognition',
 'of',
 'the',
 'inherent',
 'dignity',
 'and',
 'of',
 'the',
 'equal',
 'and',
 'inalienable',
 'right',
 'of']
```

Tokenization

In [19]:
```
text11 = "Children shouldn't drink a sugary drink before bed."
text11.split(' ')
```

Out[19]:
```
['Children', "shouldn't", 'drink', 'a', 'sugary', 'drink', 'before', 'bed.']
```

In [20]:
```
nltk.word_tokenize(text11)
```

Out[20]:
```
['Children',
 'should',
 "n't",
 'drink',
 'a',
 'sugary',
 'drink',
 'before',
 'bed',
 '.']
```

In [21]:
```
text12 = "This is the first sentence. A gallon of milk in the U.S. costs $2.99. Is this the th
sentences = nltk.sent_tokenize(text12)
len(sentences)
```

Out[21]:
```
4
```

In [22]:
```
sentences
```

Out[22]:
```
['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 [23]:
```
nltk.help.upenn_tagset('MD')
```

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

In [24]:
```
text13 = nltk.word_tokenize(text11)
nltk.pos_tag(text13)
```

Out[24]:
```
[('Children', 'NNP'),
 ('should', 'MD'),
 ("n't", 'RB'),
 ('drink', 'VB'),
 ('a', 'DT'),
 ('sugary', 'JJ'),
 ('drink', 'NN'),
 ('before', 'IN'),
 ('bed', 'NN'),
 ('.', '.')]
```

In [25]:
```
text14 = nltk.word_tokenize("Visiting aunts can be a nuisance")
nltk.pos_tag(text14)
```

Out[25]:
```
[('Visiting', 'VBG'),
 ('aunts', 'NNS'),
 ('can', 'MD'),
 ('be', 'VB'),
 ('a', 'DT'),
 ('nuisance', 'NN')]
```

In [26]:
```
# 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:
    print(tree)
```

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

In [27]:
```
text16 = nltk.word_tokenize("I saw the man with a telescope")
grammar1 = nltk.data.load('mygrammar.cfg')
grammar1
```

Out[27]:
```
<Grammar with 13 productions>
```

In [28]:
```
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 telescope))))))
```

In [29]:
```
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 director)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
```

POS tagging and parsing ambiguity

In [30]:
```
text18 = nltk.word_tokenize("The old man the boat")
nltk.pos_tag(text18)
```

Out[30]:
```
[('The', 'DT'), ('old', 'JJ'), ('man', 'NN'), ('the', 'DT'), ('boat', 'NN')]
```

In [31]:
```
text19 = nltk.word_tokenize("Colorless green ideas sleep furiously")
nltk.pos_tag(text19)
```

Out[31]:
```
[('Colorless', 'NNP'),
 ('green', 'JJ'),
 ('ideas', 'NNS'),
 ('sleep', 'VBP'),
 ('furiously', 'RB')]
```