Module 2 (Python 3)

April 29, 2022

1 Module 2 (Python 3)

1.1 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
```

1.1.1 Counting vocabulary of words

```
In [3]: text7

Out[3]: <Text: Wall Street Journal>

In [4]: sent7

Out[4]: ['Pierre',
         'Vinken',
         ',',
         '61',
         'years',
         'old',
         ',',
         'will',
         'join',
         'the',
         'board',
         'as',
         'a',
         'nonexecutive',
         'director',
         'Nov.',
         '29',
         '.']
```

```
In [5]: len(sent7)
```

```
Out[5]: 18
```

```
In [6]: len(text7)
```

```
Out[6]: 100676
```

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

```
Out[7]: 12408
```

```
In [8]: list(set(text7))[:10]
```

```
Out[8]: ['bottom',
         'Richmond',
         'tension',
         'limits',
         'Wedtech',
         'most',
         'boost',
         '143.80',
         'Dale',
         'refunded']
```

1.1.2 Frequency of words

```
In [9]: dist = FreqDist(text7)
        len(dist)
```

```
Out[9]: 12408
```

```
In [15]: vocab1 = dist.keys()
         #vocab1[:10]
         # In Python 3 dict.keys() returns an iterable view instead of a list
         list(vocab1)[:10]
```

```
Out[15]: ['Pierre', 'Vinken', ',', '61', 'years', 'old', 'will', 'join', 'the', 'board']
```

```
In [16]: dist['four']

Out[16]: 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']
```

1.1.3 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']
```

1.1.4 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']
```

1.1.5 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
         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!']
```

1.2 Advanced NLP Tasks with NLTK

1.2.1 POS tagging

```
In [33]: 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 [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:
             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 telescope))))))
```

```
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 director)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
```

1.2.2 POS tagging and parsing ambiguity

```
In [43]: text18 = nltk.word_tokenize("The old man the boat")
         nltk.pos_tag(text18)
```

```
Out[43]: [('The', 'DT'), ('old', 'JJ'), ('man', 'NN'), ('the', 'DT'), ('boat', 'NN')]
```

```
In [44]: text19 = nltk.word_tokenize("Colorless green ideas sleep furiously")
         nltk.pos_tag(text19)
```

```
Out[44]: [('Colorless', 'NNP'),
          ('green', 'JJ'),
          ('ideas', 'NNS'),
          ('sleep', 'VBP'),
          ('furiously', 'RB')]
```