Module 2 (Python 3)

October 1, 2020

1 Module 2 (Python 3)

1.1 Basic NLP Tasks with NLTK

```
In [27]: import nltk

         nltk.download('gutenberg')

         nltk.download('genesis')

         nltk.download('inaugural')

         nltk.download('nps_chat')

         nltk.download('webtext')

         nltk.download('treebank')

         nltk.download('udhr')

         nltk.download('tagsets')

         nltk.download('averaged_perceptron_tagger')

         from nltk.book import *
```

```
[nltk_data] Downloading package tagsets to /home/jovyan/nltk_data...
[nltk_data]   Unzipping help/tagsets.zip.
```

1.1.1 Counting vocabulary of words

```
In [28]: text7
```

```
Out[28]: <Text: Wall Street Journal>
```

```
In [29]: sents()
```

```
sent1: Call me Ishmael .
sent2: The family of Dashwood had long been settled in Sussex .
sent3: In the beginning God created the heaven and the earth .
sent4: Fellow - Citizens of the Senate and of the House of Representatives :
sent5: I have a problem with people PMing me to lol JOIN
sent6: SCENE 1 : [ wind ] [ clop clop clop ] KING ARTHUR : Whoa there !
sent7: Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 .
sent8: 25 SEXY MALE , seeks attrac older single lady , for discreet encounters .
sent9: THE suburb of Saffron Park lay on the sunset side of London , as red and ragged as a clou
```

```
In [6]: sent7
```

```
Out[6]: ['Pierre',
         'Vinken',
         ',',
         '61',
         'years',
         'old',
         ',',
         'will',
         'join',
         'the',
         'board',
         'as',
         'a',
         'nonexecutive',
         'director',
         'Nov.',
         '29',
         '.']
```

```
In [7]: len(sent7)
```

```
Out[7]: 18
```

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

```
Out[8]: 100676
```

```
In [9]: len(set(text7))
```

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

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

```
Out[10]: ['Educational',
          'file',
          'Knowing',
```

```
 'Cross',
 'halt',
 'DEPOSIT',
 'Tony',
 'transition',
 'Probably',
 'Cristal']
```

1.1.2 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 list
         list(vocab1)[:10]
```

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

```
In [13]: dist['four']
```

```
Out[13]: 20
```

```
In [14]: freqwords = [w for w in vocab1 if len(w) > 5 and dist[w] > 100]
         freqwords
```

```
Out[14]: ['billion',
          'company',
          'president',
          'because',
          'market',
          'million',
          'shares',
          'trading',
          'program']
```

1.1.3 Normalization and stemming

```
In [15]: input1 = "List listed lists listing listings"
         words1 = input1.lower().split(' ')
         words1
```

```
Out[15]: ['list', 'listed', 'lists', 'listing', 'listings']
```

```
In [16]: porter = nltk.PorterStemmer()
         [porter.stem(t) for t in words1]
```

```
Out[16]: ['list', 'list', 'list', 'list', 'list']
```

1.1.4 Lemmatization

```
In [19]: udhr = nltk.corpus.udhr.words('English-Latin1')
         udhr[:20]

Out[19]: ['Universal',
          'Declaration',
          'of',
          'Human',
          'Rights',
          'Preamble',
          'Whereas',
          'recognition',
          'of',
          'the',
          'inherent',
          'dignity',
          'and',
          'of',
          'the',
          'equal',
          'and',
          'inalienable',
          'rights',
          'of']

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

Out[20]: ['univers',
          'declar',
          'of',
          'human',
          'right',
          'preambl',
          'wherea',
          'recognit',
          'of',
          'the',
          'inher',
          'digniti',
          'and',
          'of',
          'the',
          'equal',
          'and',
          'inalien',
          'right',
          'of']
```

```
In [21]: WNlemma = nltk.WordNetLemmatizer()
         [WNlemma.lemmatize(t) for t in udhr[:20]]
```

```
Out[21]: ['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 [22]: text11 = "Children shouldn't drink a sugary drink before bed."
         text11.split(' ')
```

```
Out[22]: ['Children', "shouldn't", 'drink', 'a', 'sugary', 'drink', 'before', 'bed.']
```

```
In [23]: nltk.word_tokenize(text11)
```

```
Out[23]: ['Children',
          'should',
          "n't",
          'drink',
          'a',
          'sugary',
          'drink',
          'before',
          'bed',
          '.']
```

```
In [24]: 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[24]: 4
```

```
In [25]: sentences
```

```
Out[25]: ['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 [30]: 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 [32]: text13 = nltk.word_tokenize(text11)
         nltk.pos_tag(text13)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/jovyan/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

```
Out[32]: [('Children', 'NNP'),
          ('should', 'MD'),
          ("n't", 'RB'),
          ('drink', 'VB'),
          ('a', 'DT'),
          ('sugary', 'JJ'),
          ('drink', 'NN'),
          ('before', 'IN'),
          ('bed', 'NN'),
          ('.', '.')]
```

```
In [33]: text14 = nltk.word_tokenize("Visiting aunts can be a nuisance")
         nltk.pos_tag(text14)
```

```
Out[33]: [('Visiting', 'VBG'),
          ('aunts', 'NNS'),
          ('can', 'MD'),
          ('be', 'VB'),
          ('a', 'DT'),
          ('nuisance', 'NN')]
```

```
In [34]: # 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 [35]: text16 = nltk.word_tokenize("I saw the man with a telescope")
         grammar1 = nltk.data.load('mygrammar.cfg')
         grammar1
```

```
Out[35]: <Grammar with 13 productions>
```

```
In [36]: 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 [37]: 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 [38]: text18 = nltk.word_tokenize("The old man the boat")
         nltk.pos_tag(text18)

Out[38]: [('The', 'DT'), ('old', 'JJ'), ('man', 'NN'), ('the', 'DT'), ('boat', 'NN')]

In [39]: text19 = nltk.word_tokenize("Colorless green ideas sleep furiously")
         nltk.pos_tag(text19)

Out[39]: [('Colorless', 'NNP'),
          ('green', 'JJ'),
          ('ideas', 'NNS'),
          ('sleep', 'VBP'),
          ('furiously', 'RB')]

In [ ]:
```