*You are currently looking at **version 1.0** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ (https://www.coursera.org/learn/python-text-mining/resources/d9pwm) course resource.*

Assignment 2 - Introduction to NLTK

In part 1 of this assignment you will use nltk to explore the Herman Melville novel Moby Dick. Then in part 2 you will create a spelling recommender function that uses nltk to find words similar to the misspelling.

Part 1 - Analyzing Moby Dick

In [1]:

```
import nltk
from nltk.probability import FreqDist
import pandas as pd
import numpy as np

# If you would like to work with the raw text you can use 'moby_raw'
with open('moby.txt', 'r') as f:
    moby_raw = f.read()

# If you would like to work with the novel in nltk.Text format you can use 'text1'
moby_tokens = nltk.word_tokenize(moby_raw)
text1 = nltk.Text(moby_tokens)
```

In [2]:

```
dist = FreqDist(text1)
vocab1 = dist.keys()
unq = set(text1)
```

Example 1

How many tokens (words and punctuation symbols) are in text1?

This function should return an integer.

In [3]:

```
def example_one():

    return len(nltk.word_tokenize(moby_raw)) # or alternatively len(text1)

example_one()
```

Out[3]:

```
254989
```

Example 2

How many unique tokens (unique words and punctuation) does text1 have?

This function should return an integer.

In [4]:

```
def example_two():

    return len(set(nltk.word_tokenize(moby_raw))) # or alternatively len(set(text1))

example_two()
```

Out[4]:

```
20755
```

Example 3

After lemmatizing the verbs, how many unique tokens does text1 have?

This function should return an integer.

In [5]:

```
from nltk.stem import WordNetLemmatizer

def example_three():

    lemmatizer = WordNetLemmatizer()
    lemmatized = [lemmatizer.lemmatize(w, 'v') for w in text1]

    return len(set(lemmatized))

example_three()
```

Out[5]:

```
16900
```

Question 1

What is the lexical diversity of the given text input? (i.e. ratio of unique tokens to the total number of tokens)

This function should return a float.

In [6]:

```
def answer_one():

    ans = 20755/254989   # ans = example_two()/example_one()
    return ans           # Your answer here

answer_one()
```

Out[6]:

```
0.08139566804842562
```

Question 2

What percentage of tokens is 'whale'or 'Whale'?

This function should return a float.

In [7]:

```
def answer_two():
    import re
    ans = len([w for w in moby_tokens if len(w)==5 and re.match(r'[Ww]hale',w)])/254989
    return ans * 100 # Your answer here

answer_two()
```

Out[7]:

```
0.4125668166077752
```

Question 3

What are the 20 most frequently occurring (unique) tokens in the text? What is their frequency?

This function should return a list of 20 tuples where each tuple is of the form `(token, frequency)`. The list should be sorted in descending order of frequency.

In [8]:

```
def answer_three():

    freqwords = [w for w in vocab1 if dist[w] > 1500]
    df = pd.DataFrame(freqwords, columns = ['word'])
    df['freq'] = 0
    for i in range(len(df)):
        df['freq'][i] = dist[df['word'][i]]
    df = df.sort_values(['freq'], ascending = False)
    df.reset_index(inplace = True, drop = True)
    ans = []
    for i in range(20):
        ans.append((df['word'][i],df['freq'][i]))

    return ans  # Your answer here

answer_three()
```

Out[8]:

```
[(',', 19204),
 ('the', 13715),
 ('.', 7308),
 ('of', 6513),
 ('and', 6010),
 ('a', 4545),
 ('to', 4515),
 (';', 4173),
 ('in', 3908),
 ('that', 2978),
 ('his', 2459),
 ('it', 2196),
 ('I', 2097),
 ('!', 1767),
 ('is', 1722),
 ('--', 1713),
 ('with', 1659),
 ('he', 1658),
 ('was', 1639),
 ('as', 1620)]
```

Question 4

What tokens have a length of greater than 5 and frequency of more than 150?

This function should return a sorted list of the tokens that match the above constraints. To sort your list, use `sorted()`

In [9]:

```
def answer_four():

    freqwords = [w for w in vocab1 if len(w) > 5 and dist[w] > 150]

    return sorted(freqwords)  # Your answer here

answer_four()
```

Out[9]:

```
['Captain',
 'Pequod',
 'Queequeg',
 'Starbuck',
 'almost',
 'before',
 'himself',
 'little',
 'seemed',
 'should',
 'though',
 'through',
 'whales',
 'without']
```

Question 5

Find the longest word in text1 and that word's length.

This function should return a tuple `(longest_word, length)`.

In [10]:

```
def answer_five():

    longest_word =  max(unq, key=len)

    return (longest_word, len(longest_word))  # Your answer here

answer_five()
```

Out[10]:

```
("twelve-o'clock-at-night", 23)
```

Question 6

What unique words have a frequency of more than 2000? What is their frequency?

"Hint: you may want to use `isalpha()` to check if the token is a word and not punctuation."

This function should return a list of tuples of the form `(frequency, word)` sorted in descending order of frequency.

In [11]:

```
def answer_six():

    freqwords = [w for w in vocab1 if dist[w] > 2000]
    df = pd.DataFrame(freqwords, columns = ['word'])
    df['check'] = False
    df['freq'] = 0
    for i in range(len(df)):
        df['check'][i] = df['word'][i].isalpha()
        df['freq'][i] = dist[df['word'][i]]
    df = df.sort_values(['freq'], ascending = False)
    df = df[df['check']]
    df.reset_index(inplace = True, drop = True)
    ans = []
    for i in range(len(df)):
        ans.append((df['freq'][i], df['word'][i]))

    return ans  # Your answer here

answer_six()
```

Out[11]:

```
[(13715, 'the'),
 (6513, 'of'),
 (6010, 'and'),
 (4545, 'a'),
 (4515, 'to'),
 (3908, 'in'),
 (2978, 'that'),
 (2459, 'his'),
 (2196, 'it'),
 (2097, 'I')]
```

Question 7

What is the average number of tokens per sentence?

This function should return a float.

In [12]:

```
def answer_seven():
    n_token = 254989
    n_sent = len(nltk.sent_tokenize(moby_raw))
    ans = n_token/n_sent

    return ans  # Your answer here

answer_seven()
```

Out[12]:

```
25.881952902963864
```

Question 8

What are the 5 most frequent parts of speech in this text? What is their frequency?

This function should return a list of tuples of the form `(part_of_speech, frequency)` sorted in descending order of frequency.

In [13]:

```
def answer_eight():

    pstg = nltk.pos_tag(text1)
    pos = [x[1] for x in pstg]
    dist8 = FreqDist(pos)
    df8 = pd.DataFrame(np.unique(pos), columns = ['pos'])
    df8['freq'] = 0
    for i in range(len(df8)):
        df8['freq'][i] = dist8[df8['pos'][i]]
    df8 = df8.sort_values(['freq'], ascending = False)
    df8.reset_index(inplace = True, drop = True)
    df8 = df8[0:5]
    ans = [tuple(x) for x in df8.to_records(index = False)]

    return ans  # Your answer here

answer_eight()
```

Out[13]:

```
[('NN', 32730), ('IN', 28657), ('DT', 25867), (',', 19204), ('JJ', 17620)]
```

Part 2 - Spelling Recommender

For this part of the assignment you will create three different spelling recommenders, that each take a list of misspelled words and recommends a correctly spelled word for every word in the list.

For every misspelled word, the recommender should find find the word in `correct_spellings` that has the shortest distance*, and starts with the same letter as the misspelled word, and return that word as a recommendation.

*Each of the three different recommenders will use a different distance measure (outlined below).

Each of the recommenders should provide recommendations for the three default words provided: `['cormulent', 'incendenece', 'validrate']`.

In [14]:

```
from nltk.corpus import words

correct_spellings = words.words()
```

In [15]:

```
from nltk.metrics.distance import (
    edit_distance,
    jaccard_distance)
from nltk.util import ngrams
```

Question 9

For this recommender, your function should provide recommendations for the three default words provided above using the following distance metric:

Jaccard distance (https://en.wikipedia.org/wiki/Jaccard_index) on the trigrams of the two words.

This function should return a list of length three: `['cormulent_reccomendation', 'incendenece_reccomendation', 'validrate_reccomendation']`.

In [16]:

```
def answer_nine(entries=['cormulent', 'incendenece', 'validrate']):

    ans = []
    for item in entries:
        c = list(item)[0]
        word_list = [x for x in correct_spellings if x.startswith(c)]
        sim = [nltk.jaccard_distance(set(nltk.ngrams(item, n=3)), set(nltk.ngrams(x, n=3))) for x in word_list]
        min_index = sim.index(min(sim))
        ans.append(word_list[min_index])

    return ans  # Your answer here

answer_nine()
```

Out[16]:

```
['corpulent', 'indecence', 'validate']
```

Question 10

For this recommender, your function should provide recommendations for the three default words provided above using the following distance metric:

Jaccard distance (https://en.wikipedia.org/wiki/Jaccard_index) on the 4-grams of the two words.

This function should return a list of length three: `['cormulent_reccomendation', 'incendenece_reccomendation', 'validrate_reccomendation']`.

In [17]:

```
def answer_ten(entries=['cormulent', 'incendenece', 'validrate']):

    ans = []
    for item in entries:
        c = list(item)[0]
        word_list = [x for x in correct_spellings if x.startswith(c)]
        sim = [nltk.jaccard_distance(set(nltk.ngrams(item, n=4)), set(nltk.ngrams(x, n=4))) for x in word_list]
        min_index = sim.index(min(sim))
        ans.append(word_list[min_index])

    return ans  # Your answer here

answer_ten()
```

Out[17]:

```
['cormus', 'incendiary', 'valid']
```