*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 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 [13]:
```python
#nltk.download()


import nltk
import pandas as pd
import numpy as np




nltk.download('gutenberg')

nltk.download('genesis')

nltk.download('inaugural')

nltk.download('nps_chat')

nltk.download('treebank')

nltk.download('udhr')

nltk.download('webtext')

nltk.download('wordnet')

nltk.download('punkt')

nltk.download('wordnet')
#nltk.download()
from nltk.book import *

# 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)
```

```
[nltk_data] Downloading package gutenberg to /home/jovyan/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package genesis to /home/jovyan/nltk_data...
[nltk_data]   Package genesis is already up-to-date!
[nltk_data] Downloading package inaugural to /home/jovyan/nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
[nltk_data] Downloading package nps_chat to /home/jovyan/nltk_data...
[nltk_data]   Package nps_chat is already up-to-date!
[nltk_data] Downloading package treebank to /home/jovyan/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
[nltk_data] Downloading package udhr to /home/jovyan/nltk_data...
[nltk_data]   Unzipping corpora/udhr.zip.
[nltk_data] Downloading package webtext to /home/jovyan/nltk_data...
[nltk_data]   Package webtext is already up-to-date!
[nltk_data] Downloading package wordnet to /home/jovyan/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package punkt to /home/jovyan/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /home/jovyan/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Example 1

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

This function should return an integer.

In [14]:
```python
def example_one():

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

example_one()
```

Out[14]: `254989`

Example 2

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

This function should return an integer.

In [15]:
```python
def example_two():

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

example_two()
```

Out[15]: `20755`

Example 3

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

This function should return an integer.

In [16]:
```python
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[16]: `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 [17]:
```python
def answer_one():


    return example_two()/example_one()

answer_one()
```

Out[17]: `0.08139566804842562`

Question 2

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

This function should return a float.

In [18]:
```python
def answer_two():


    return (text1.vocab()['whale'] + text1.vocab()['Whale']) / len(nltk.word_tokenize(moby_raw)) * 100

answer_two()
```

Out[18]: `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 [20]:
```python
def answer_three():

    import operator
    return sorted(text1.vocab().items(), key=operator.itemgetter(1), reverse=True)[:20]

answer_three()
```

Out[20]:
```
[(',', 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 an alphabetically sorted list of the tokens that match the above constraints. To sort your list, use `sorted()`

In [21]:
```python
def answer_four():


    return sorted([token for token, freq in text1.vocab().items() if len(token) > 5 and freq > 150])

answer_four()
```

Out[21]:
```
['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 [22]:
```python
def answer_five():
    import operator

    return sorted([(token, len(token))for token, freq in text1.vocab().items()], key=operator.itemgetter(1), reverse=T
rue)[0]

answer_five()
```

Out[22]: `("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 [23]:
```python
def answer_six():

    import operator

    return sorted([(freq, token) for token, freq in text1.vocab().items() if freq > 2000 and token.isalpha()], key=ope
rator.itemgetter(0), reverse=True)

answer_six()
```

Out[23]:
```
[(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 [30]:
```python
def answer_seven():


   # use the built-in package to split the text into sentences

    sentences = nltk.sent_tokenize(moby_raw)

#     print(len(sentences))

    countWordsSum = 0

    # count all the words in each sentences

    for i in range(len(sentences)):

        words = nltk.word_tokenize(sentences[i])

        countWordsSum = countWordsSum + len(words)

    return (countWordsSum / len(sentences))

answer_seven()
```

Out[30]: `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 [38]:
```python
moby_frequencies = nltk.FreqDist(moby_tokens)
    # set up the dataframe
df = pd.DataFrame(moby_frequencies.most_common(),columns=["token", "frequency"])
# find the valid words in moby
moby_words = df[df.token.str.isalpha()]
print(moby_words)
```

```
                token  frequency
1                 the      13715
3                  of       6513
4                 and       6010
5                   a       4545
6                  to       4515
8                  in       3908
9                that       2978
10                his       2459
11                 it       2196
12                  I       2097
14                 is       1722
16               with       1659
17                 he       1658
18                was       1639
19                 as       1620
23                all       1444
24                for       1413
25               this       1280
26                 at       1230
27                not       1170
28                 by       1135
29                but       1110
30                him       1058
31               from       1052
32                 be       1027
34                 on       1003
35                 so        914
36                one        880
37                you        841
38              whale        782
...               ...        ...
20720        lookouts          1
20722         animate          1
20723       inanimate          1
20724       whelmings          1
20725  intermixingly          1
20726         ironical          1
20727     coincidings          1
20728      destroying          1
20729      backwardly          1
20731      tauntingly          1
20732     incommoding          1
20733       intercept          1
20734        etherial          1
20735          thrill          1
20737        Epilogue          1
20738            ONLY          1
20739         ESCAPED          1
20740            THEE          1
20741       halfspent          1
20742          suction          1
20743         closing          1
20745           Ixion          1
20746            Till          1
20747       liberated          1
20748          Buoyed          1
20749       dirgelike          1
20750        padlocks          1
20751        sheathed          1
20753       retracing          1
20754          orphan          1

[18464 rows x 2 columns]
```

In [42]:
```python
def answer_eight():
    import collections
    import nltk
    nltk.download('averaged_perceptron_tagger')


    from collections import Counter
    import operator

    return sorted(Counter([tag for token, tag in nltk.pos_tag(text1)]).items(), key=operator.itemgetter(1), reverse=Tr
ue)[:5]

answer_eight()
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/jovyan/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[42]: `[('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 [44]:
```python
from nltk.corpus import words
import nltk
nltk.download('words')
correct_spellings = words.words()
```

```
[nltk_data] Downloading package words to /home/jovyan/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
```

Question 9

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

Jaccard distance on the trigrams of the two words.

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

In [45]:
```python
def answer_nine(entries=['cormulent', 'incendenece', 'validrate']):
    result = []
    import operator
    for entry in entries:
        spell_list = [spell for spell in correct_spellings if spell.startswith(entry[0]) and len(spell) > 2]
        distance_list = [(spell, nltk.jaccard_distance(set(nltk.ngrams(entry, n=3)), set(nltk.ngrams(spell, n=3)))) fo
r spell in spell_list]

        result.append(sorted(distance_list, key=operator.itemgetter(1))[0][0])

    return result # Your answer here

answer_nine()
```

Out[45]: `['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 on the 4-grams of the two words.

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

In [46]:
```python
def answer_ten(entries=['cormulent', 'incendenece', 'validrate']):
    result = []
    import operator
    for entry in entries:
        spell_list = [spell for spell in correct_spellings if spell.startswith(entry[0]) and len(spell) > 2]
        distance_list = [(spell, nltk.jaccard_distance(set(nltk.ngrams(entry, n=4)), set(nltk.ngrams(spell, n=4)))) fo
r spell in spell_list]

        result.append(sorted(distance_list, key=operator.itemgetter(1))[0][0])

    return result # Your answer here

answer_ten()
```

Out[46]: `['cormus', 'incendiary', 'valid']`

Question 11

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

Edit distance on the two words with transpositions.

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

In [47]:
```python
def answer_eleven(entries=['cormulent', 'incendenece', 'validrate']):
    result = []
    import operator
    for entry in entries:
        spell_list = [spell for spell in correct_spellings if spell.startswith(entry[0]) and len(spell) > 2]
        distance_list = [(spell, nltk.edit_distance(entry, spell, transpositions=True)) for spell in spell_list]

        result.append(sorted(distance_list, key=operator.itemgetter(1))[0][0])

    return result# Your answer here

answer_eleven()
```

Out[47]: `['corpulent', 'intendence', 'validate']`

In [ ]: