Programming-Assignment-2

July 1, 2020

*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.*

1 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.

1.1 Part 1 - Analyzing Moby Dick

[1]:
```
import nltk
import pandas as pd
import numpy as np
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('words')

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

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

1.1.1 Example 1

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

This function should return an integer.

[2]:
```python
def example_one():

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

example_one()
```

[2]: 254989

1.1.2 Example 2

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

This function should return an integer.

[3]:
```python
def example_two():

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

example_two()
```

[3]: 20755

1.1.3 Example 3

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

This function should return an integer.

[4]:
```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()
```

[4]: 16900

1.1.4 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.

[5]:
```python
def answer_one():

    unique_tokens = len(set(nltk.word_tokenize(moby_raw)))
    total_tokens = len(nltk.word_tokenize(moby_raw))
    ratio = unique_tokens / total_tokens

    return ratio

answer_one()
```

[5]:
```
0.08139566804842562
```

1.1.5 Question 2

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

This function should return a float.

[6]:
```python
def answer_two():

    dist = nltk.FreqDist(text1)
    total_tokens = len(text1)
    whale = dist['whale']
    Whale = dist['Whale']
    answer = (whale + Whale) / total_tokens * 100
    return answer

answer_two()
```

[6]:
```
0.4125668166077752
```

1.1.6 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.

[7]:
```python
def answer_three():


    return nltk.FreqDist(text1).most_common(20)

answer_three()
```

[7]:
```
[(',', 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)]
```

1.1.7 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()`

[8]:
```
def answer_four():

    dist = nltk.FreqDist(text1)
    vocab = dist.keys()
    return sorted([word for word in vocab if len(word) > 5 and dist[word] >␣
 ↪150])

answer_four()
```

[8]:
```
['Captain',
 'Pequod',
 'Queequeg',
 'Starbuck',
 'almost',
 'before',
 'himself',
 'little',
 'seemed',
 'should',
 'though',
 'through',
 'whales',
 'without']
```

1.1.8 Question 5

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

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

[9]:
```
def answer_five():

    dist = nltk.FreqDist(text1)
    vocab = dist.keys()

    # Find a word with max length
    max_length = 0
    longest_word = None
    for word in vocab:
        if len(word) > max_length:
            max_length = len(word)
            longest_word = word

    return (longest_word, max_length)

answer_five()
```

[9]:
```
("twelve-o'clock-at-night", 23)
```

1.1.9 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.*

[10]:
```
def answer_six():

    dist = nltk.FreqDist(text1)
    result = []
    for word, freq in dist.items():
        if word.isalpha() and freq > 2000:
            result.append((freq, word))

    return sorted(result, reverse=True)

answer_six()
```

[10]:
```
[(13715, 'the'),
 (6513, 'of'),
 (6010, 'and'),
 (4545, 'a'),
 (4515, 'to'),
 (3908, 'in'),
 (2978, 'that'),
 (2459, 'his'),
 (2196, 'it'),
 (2097, 'I')]
```

1.1.10 Question 7

What is the average number of tokens per sentence?

This function should return a float.

[11]:
```python
def answer_seven():

    sentences = nltk.word_tokenize(moby_raw)
    tokens = nltk.sent_tokenize(moby_raw)

    return len(sentences) / len(tokens)

answer_seven()
```

[11]:
```
25.881952902963864
```

1.1.11 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.

[12]:
```python
def answer_eight():

    import collections

    pos_list = nltk.pos_tag(text1)
    pos_counts = collections.Counter((a[1] for a in pos_list))

    return pos_counts.most_common(5)

answer_eight()
```

[12]:
```
[('NN', 32730), ('IN', 28657), ('DT', 25867), (',', 19204), ('JJ', 17620)]
```

1.2 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']`.

```
[13]: from nltk.corpus import words

      correct_spellings = words.words()
```

1.2.1 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']`*.*

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

          recommend = []
          for entry in entries:

              input_spell = [x for x in correct_spellings if x[0] == entry[0]]
              jaccard_dist = [nltk.jaccard_distance(set(nltk.ngrams(entry,n=3)),␣
       ↪set(nltk.ngrams(x,n=3))) for x in input_spell]
              recommend.append(input_spell[np.argmin(jaccard_dist)])

          return recommend

      answer_nine()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:7:
DeprecationWarning: generator 'ngrams' raised StopIteration
  import sys
```

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

1.2.2 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']`*.*

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

          recommend = []
          for entry in entries:

              input_spell = [x for x in correct_spellings if x[0] == entry[0]]
```

```
        jaccard_dist = [nltk.jaccard_distance(set(nltk.ngrams(entry,n=4)),␣
↪set(nltk.ngrams(x,n=4))) for x in input_spell]
        recommend.append(input_spell[np.argmin(jaccard_dist)])

    return recommend

answer_ten()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:7:
DeprecationWarning: generator 'ngrams' raised StopIteration
  import sys
```

[15]:
```
['cormus', 'incendiary', 'valid']
```

1.2.3 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']`.

[16]:
```
def answer_eleven(entries=['cormulent', 'incendenece', 'validrate']):

    recommend = []
    for entry in entries:

        input_spell = [x for x in correct_spellings if x[0] == entry[0]]
        DL_dist = [nltk.edit_distance(x, entry, transpositions=True) for x in␣
↪input_spell]
        recommend.append(input_spell[np.argmin(DL_dist)])

    return recommend

answer_eleven()
```

[16]:
```
['corpulent', 'intendence', 'validate']
```