Assignment 2

June 8, 2019

---

*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

```
In [1]: import nltk
        import pandas as pd
        import numpy as np
        nltk.download('punkt')
        # 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!
```

1.1.1 Example 1

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

This function should return an integer.

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

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

        example_one()
```

```
Out[2]: 254989
```

1.1.2 Example 2

How many unique tokens (unique words and punctuation) does text1 have?
This function should return an integer.

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

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

        example_two()
```

```
Out[3]: 20755
```

1.1.3 Example 3

After lemmatizing the verbs, how many unique tokens does text1 have?
This function should return an integer.

```
In [4]: from nltk.stem import WordNetLemmatizer
        nltk.download('wordnet')
        def example_three():

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

            return len(set(lemmatized))

        example_three()
```

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

```
Out[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.

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


            return example_two()/example_one()# Your answer here

        answer_one()
```

```
Out[5]: 0.08139566804842562
```

1.1.5 Question 2

What percentage of tokens is 'whale'or 'Whale'?
This function should return a float.

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


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

        answer_two()
```

```
Out[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.

```
In [7]: def answer_three():
            import operator

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

        answer_three()
```

```
Out[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()`

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

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

        answer_four()
```

```
Out[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)`.

```
In [9]: def answer_five():
            import operator

            return sorted([(token, len(token))for token, freq in text1.vocab().items()], key=ope

        answer_five()
```

```
Out[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.

```
In [10]: def answer_six():
             import operator

             return sorted([(freq, token) for token, freq in text1.vocab().items() if freq > 200

         answer_six()
```

```
Out[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.

```
In [11]: def answer_seven():
             return np.mean([len(nltk.word_tokenize(sent)) for sent in nltk.sent_tokenize(moby_r

         answer_seven()
```

```
Out[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.

```
In [19]: def answer_eight():
             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=ope
         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[19]: [('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']`.

```
In [13]: nltk.download('words')
         from nltk.corpus import words

         correct_spellings = words.words()
```

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

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']`.

```
In [14]: def answer_nine(entries=['cormulent', 'incendenece', 'validrate']):
             l= len(entries)
             d = [1]* l
             recom = [w[0] for w in entries]
             for i in correct_spellings:
                 for j in range(l):
                     if i.startswith(entries[j][0]):
                         a= set(nltk.ngrams(entries[j], n=3))
                         b= set(nltk.ngrams(i, n=3))
                         if nltk.jaccard_distance(a,b)< d[j]:
                             d[j] = nltk.jaccard_distance(a,b)
```

```
                            recom[j] = i
            return recom

        answer_nine()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: DeprecationWarning: generator 'n
  if __name__ == '__main__':
```

```
Out[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'].*

```
In [17]: def answer_ten(entries=['cormulent', 'incendenece', 'validrate']):
             l= len(entries)
             d = [1]* l
             recom = [w[0] for w in entries]
             for i in correct_spellings:
                 for j in range(l):
                     if i.startswith(entries[j][0]):
                         a= set(nltk.ngrams(entries[j], n=4))
                         b= set(nltk.ngrams(i, n=4))
                         if nltk.jaccard_distance(a,b)< d[j]:
                             d[j] = nltk.jaccard_distance(a,b)
                             recom[j] = i

             return recom

         answer_ten()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: DeprecationWarning: generator 'n
  if __name__ == '__main__':
```

```
Out[17]: ['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'].*

```
In [18]: def answer_eleven(entries=['cormulent', 'incendenece', 'validrate']):
             l= len(entries)
             recom = [w[0] for w in entries]
             d = [nltk.edit_distance(recom[i],entries[i],transpositions = True) for i in range(l
             for i in correct_spellings:
                 w=i.lower()
                 for j in range(l):
                     if w.startswith(entries[j][0]):
                         if nltk.edit_distance(w,entries[j],transpositions = True)< d[j]:
                             d[j] = nltk.edit_distance(w,entries[j],transpositions = True)
                             recom[j] = w
             return recom

         answer_eleven()

Out[18]: ['corpulent', 'intendence', 'validate']

In [ ]:
```