Assignment 2

May 9, 2021

---

*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 [69]: import nltk

         nltk.download('punkt')
         nltk.download('gutenberg')
         nltk.download('genesis')
         nltk.download('inaugural')
         nltk.download('nps_chat')
         nltk.download('webtext')
         nltk.download('treebank')
         nltk.download('averaged_perceptron_tagger')
         nltk.download('words')
         from nltk.book import *
         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)
         #print(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 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 webtext to /home/jovyan/nltk_data...
[nltk_data]   Package webtext 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 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.

```
In [70]: def example_one():

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

         example_one()

Out[70]: 254989
```

1.1.2 Example 2

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

This function should return an integer.

```
In [71]: def example_two():

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

         example_two()

Out[71]: 20755
```

1.1.3 Example 3

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

```
In [72]: 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()


        ---------------------------------------------------------------------------

        LookupError                               Traceback (most recent call last)

        /opt/conda/lib/python3.6/site-packages/nltk/corpus/util.py in __load(self)
         79             except LookupError as e:
    ---> 80                 try: root = nltk.data.find('{}/{}'.format(self.subdir, zip_name))
         81                 except LookupError: raise e


        /opt/conda/lib/python3.6/site-packages/nltk/data.py in find(resource_name, paths)
        674     resource_not_found = '\n%s\n%s\n%s\n' % (sep, msg, sep)
    --> 675     raise LookupError(resource_not_found)
        676


        LookupError:
    **********************************************************************
      Resource wordnet not found.
      Please use the NLTK Downloader to obtain the resource:

      >>> import nltk
      >>> nltk.download('wordnet')

      Searched in:
        - '/home/jovyan/nltk_data'
        - '/usr/share/nltk_data'
        - '/usr/local/share/nltk_data'
        - '/usr/lib/nltk_data'
        - '/usr/local/lib/nltk_data'
        - '/opt/conda/nltk_data'
        - '/opt/conda/share/nltk_data'
    - '/opt/conda/lib/nltk_data'
**********************************************************************



During handling of the above exception, another exception occurred:


    LookupError                               Traceback (most recent call last)

    <ipython-input-72-6097938ac8b3> in <module>()
      8     return len(set(lemmatized))
      9
---> 10 example_three()


    <ipython-input-72-6097938ac8b3> in example_three()
      4
      5     lemmatizer = WordNetLemmatizer()
----> 6     lemmatized = [lemmatizer.lemmatize(w,'v') for w in text1]
      7
      8     return len(set(lemmatized))


    <ipython-input-72-6097938ac8b3> in <listcomp>(.0)
      4
      5     lemmatizer = WordNetLemmatizer()
----> 6     lemmatized = [lemmatizer.lemmatize(w,'v') for w in text1]
      7
      8     return len(set(lemmatized))


    /opt/conda/lib/python3.6/site-packages/nltk/stem/wordnet.py in lemmatize(self, word, pos
     38
     39     def lemmatize(self, word, pos=NOUN):
---> 40         lemmas = wordnet._morphy(word, pos)
     41         return min(lemmas, key=len) if lemmas else word
     42


    /opt/conda/lib/python3.6/site-packages/nltk/corpus/util.py in __getattr__(self, attr)
    114             raise AttributeError("LazyCorpusLoader object has no attribute '__bases_
    115
--> 116         self.__load()
    117         # This looks circular, but its not, since __load() changes our
    118         # __class__ to something new:
```

```
    /opt/conda/lib/python3.6/site-packages/nltk/corpus/util.py in __load(self)
     79             except LookupError as e:
     80                 try: root = nltk.data.find('{}/{}'.format(self.subdir, zip_name))
---> 81                 except LookupError: raise e
     82 
     83         # Load the corpus.


    /opt/conda/lib/python3.6/site-packages/nltk/corpus/util.py in __load(self)
     76         else:
     77             try:
---> 78                 root = nltk.data.find('{}/{}'.format(self.subdir, self.__name))
     79             except LookupError as e:
     80                 try: root = nltk.data.find('{}/{}'.format(self.subdir, zip_name))


    /opt/conda/lib/python3.6/site-packages/nltk/data.py in find(resource_name, paths)
    673     sep = '*' * 70
    674     resource_not_found = '\n%s\n%s\n%s\n' % (sep, msg, sep)
--> 675     raise LookupError(resource_not_found)
    676 
    677 


    LookupError: 
**********************************************************************
  Resource wordnet not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('wordnet')

  Searched in:
    - '/home/jovyan/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - '/opt/conda/nltk_data'
    - '/opt/conda/share/nltk_data'
    - '/opt/conda/lib/nltk_data'
**********************************************************************
```

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 [ ]: def answer_one():


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

        answer_one()
```

1.1.5 Question 2

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

This function should return a float.

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

            text = nltk.word_tokenize(moby_raw)
            count = 0
            for t in text:
                if t ==  'whale' or t == 'Whale':
                    count+=1
            return 100 * float(count/example_one())# Your answer here

        answer_two()
```

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 [ ]: def answer_three():

            dist = FreqDist(moby_tokens)
            import operator
            sorted_d = dict( sorted(dist.items(), key=operator.itemgetter(1),reverse=True))
            list_1 = []
            count = 0
            for k, v in sorted_d.items():
                if count < 20:
                    list_1.append((k,v))
                    count+=1
            return list_1 # Your answer here

        answer_three()
```

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 [ ]: def answer_four():

            dist = FreqDist(moby_tokens)
            words = dist.keys()
            my_words = [w for w in words if len(w)>5 and dist[w]>150]
            return sorted(my_words)# Your answer here

        answer_four()
```

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 [ ]: def answer_five():
            length1 = 0
            longest = ''
            for w in text1:
                if len(w) > length1:
                    length1 = len(w)
                    longest = w

            return longest,length1# Your answer here

        answer_five()
```

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 [ ]: def answer_six():

            dist = FreqDist(moby_tokens)
            words = dist.keys()
            words_1 = [w for w in words if w.isalpha()]
            my_words = [w for w in words_1 if dist[w]>2000]
            my_list = []
            for w in my_words:
                my_list.append((dist[w],w))
            final_list = sorted(my_list, reverse=True)

            return final_list# Your answer here

        answer_six()
```

1.1.10 Question 7

What is the average number of tokens per sentence?
This function should return a float.

```
In [ ]: def answer_seven():

            sentences = nltk.sent_tokenize(moby_raw)
            return float(len(text1)/len(sentences)) #Your answer here

        answer_seven()
```

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 [98]: def answer_eight():

             tags = nltk.pos_tag(moby_tokens)
             dist = FreqDist([tag for (word, tag) in tags])
             #final_list = sorted(dist, reverse=True)

             return dist.most_common(5)# Your answer here

           # Your answer here

         answer_eight()

Out[98]: [('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 [ ]: from nltk.corpus import words
        from nltk.metrics.distance import (
            edit_distance,
            jaccard_distance,
            )
        from nltk.util import ngrams

        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']`.

```
In [75]: spellings_series = pd.Series(correct_spellings)
         def jaccard(entries, gram_number):
             outcomes = []
             for entry in entries:
                 spellings = spellings_series[spellings_series.str.startswith(entry[0])]
                 distances = ((jaccard_distance(set(ngrams(entry, gram_number)),
                                                set(ngrams(word, gram_number))), word)
                              for word in spellings)
                 closest = min(distances)
                 outcomes.append(closest[1])
             return outcomes
         def answer_nine(entries=['cormulent', 'incendenece', 'validrate']):
             #finds the closest word based on jaccard distance
             return jaccard(entries, 3)

         #print(answer_nine())

         answer_nine()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:17: DeprecationWarning: generator '
```

```
Out[75]: ['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 [76]: def answer_ten(entries=['cormulent', 'incendenece', 'validrate']):


             return jaccard(entries, 4)

         answer_ten()

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:17: DeprecationWarning: generator '


Out[76]: ['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 [77]: def answer_eleven(entries=['cormulent', 'incendenece', 'validrate']):
             outcomes = []
             for entry in entries:
                 distances = ((edit_distance(entry,
                                             word), word)
                              for word in correct_spellings)
                 closest = min(distances)
                 outcomes.append(closest[1])
             return outcomes

         answer_eleven()

Out[77]: ['corpulent', 'intendence', 'validate']

In [ ]:
```