Assignment2 (Score: 10.0 / 11.0)

1. Test cell (Score: 1.0 / 1.0)
2. Test cell (Score: 1.0 / 1.0)
3. Test cell (Score: 1.0 / 1.0)
4. Test cell (Score: 1.0 / 1.0)
5. Test cell (Score: 1.0 / 1.0)
6. Test cell (Score: 1.0 / 1.0)
7. Test cell (Score: 0.0 / 1.0)
8. Test cell (Score: 1.0 / 1.0)
9. Test cell (Score: 1.0 / 1.0)
10. Test cell (Score: 1.0 / 1.0)
11. Test cell (Score: 1.0 / 1.0)

Assignment 2 - Introduction to NLTK¶

In part 1 of this assignment you will use nltk to explore the CMU Movie Summary Corpus (http://www.cs.cmu.edu/~ark/personas/). All data is released a Creative Commons Attribution-ShareAlike License (https://creativecommons.org/licenses/by-sa/3.0/us/legalcode). Then in part 2 you will create a sp recommender function that uses nltk to find words similar to the misspelling.

Part 1 - Analyzing Plots Summary Text¶

In [1]:

```
import nltk
import pandas as pd
import numpy as np

nltk.data.path.append("assets/")

# If you would like to work with the raw text you can use 'plots_raw'
with open('assets/plots.txt', 'rt', encoding="utf8") as f:
    plots_raw = f.read()

# If you would like to work with the plot summaries in nltk.Text format you can use 'text1'.
plots_tokens = nltk.word_tokenize(plots_raw)
text1 = nltk.Text(plots_tokens)
```

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(plots_raw)) # or alternatively len(text1)

example_one()
```

Out[2]:

```
374441
```

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(plots_raw))) # or alternatively len(set(text1))

example_two()
```

Out[3]:

```
25933
```

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

def example_three():

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

    return len(set(lemmatized))

example_three()
```

Out[4]:

```
21760
```

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]:

Student's answer

```
def answer_one():


    return example_two()/example_one()

answer_one()
```

Out[5]:

```
0.06925790712021386
```

In [6]:

Grade cell: `cell-f652837013625f3e` Score: 1.0 / 1.0

Question 2¶

What percentage of tokens is 'love'or 'Love'?

This function should return a float.

In [7]:

Student's answer

```
def answer_two():


    return (text1.vocab()['love'] + text1.vocab()['Love']) / len(nltk.word_tokenize(plots_raw)) * 100
```

In [8]:

```
answer_two()
```

Out[8]:

```
0.12391805384559917
```

In [9]:

Grade cell: `cell-27092be2e8e84d2e` Score: 1.0 / 1.0

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 [10]:

Student's answer

```
def answer_three():
    import operator

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

answer_three()
```

Out[10]:

```
[(',', 19420),
 ('the', 18698),
 ('.', 16624),
 ('to', 12149),
 ('and', 11400),
 ('a', 8979),
 ('of', 6510),
 ('is', 5699),
 ('in', 5109),
 ('his', 4693),
 ("'s", 3682),
 ('her', 3674),
 ('he', 3556),
 ('that', 3517),
 ('with', 3293),
 ('him', 2570),
 ('for', 2433),
 ('by', 2321),
 ('The', 2234),
 ('on', 1925)]
```

In [11]:

Grade cell: `cell-507a8849d081756b` Score: 1.0 / 1.0

Question 4¶

What tokens have a length of greater than 5 and frequency of more than 200?

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

In [12]:

Student's answer

```
def answer_four():

    return sorted([token for token, freq in text1.vocab().items() if len(token) > 5 and freq > 200]) # Your answer he

answer_four()
```

Out[12]:

```
['However',
 'Meanwhile',
 'another',
 'because',
 'becomes',
 'before',
 'begins',
 'daughter',
 'decides',
 'escape',
 'family',
 'father',
 'friend',
 'friends',
 'himself',
 'killed',
 'leaves',
 'mother',
 'people',
 'police',
 'returns',
 'school',
 'through']
```

In [13]:

Grade cell: `cell-dfac54d8588a79bb` Score: 1.0 / 1.0

Question 5¶

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

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

In [14]:

Student's answer

```
def answer_five():
    import operator

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

answer_five()
```

Out[14]:

```
('live-for-today-for-tomorrow-we-die', 34)
```

In [15]:

Grade cell: `cell-8f427855b0328b18` Score: 1.0 / 1.0

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 [16]:

Student's answer

```
def answer_six():
    import operator

    return sorted([(freq, token) for token, freq in text1.vocab().items() if freq > 2000 and token.isalpha()], key=op

answer_six()
```

Out[16]:

```
[(18698, 'the'),
 (12149, 'to'),
 (11400, 'and'),
 (8979, 'a'),
 (6510, 'of'),
 (5699, 'is'),
 (5109, 'in'),
 (4693, 'his'),
 (3674, 'her'),
 (3556, 'he'),
 (3517, 'that'),
 (3293, 'with'),
 (2570, 'him'),
 (2433, 'for'),
 (2321, 'by'),
 (2234, 'The')]
```

In [17]:

Grade cell: `cell-e6629ec7a557e797` Score: 1.0 / 1.0

Question 7¶

`text1` is in `nltk.Text` format that has been constructed using tokens output by `nltk.word_tokenize(plots_raw)`.

Now, use `nltk.sent_tokenize` on the tokens in `text1` by joining them using whitespace to output a sentence-tokenized copy of `text1`. Report t
average number of whitespace separated tokens per sentence in the sentence-tokenized copy of `text1`.

This function should return a float.

In [18]:

Student's answer

```
def answer_seven():

    return np.mean([len(nltk.word_tokenize(sent)) for sent in nltk.sent_tokenize(plots_raw)]) # Your answer here

answer_seven()
```

Out[18]:

```
22.317439504112528
```

In [19]:

Grade cell: `cell-11a1faa1d07cef4c` Score: 0.0 / 1.0

```
You have failed this test due to an error. The traceback has been removed because it may contain hidden tests. This is

AssertionError: This is an incorrect solution
```

Question 8¶

What are the 5 most frequent parts of speech in `text1`? 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 [20]:

Student's answer

```
def answer_eight():
    from collections import Counter
    import operator

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

answer_eight()
```

Out[20]:

```
[('NN', 51452), ('IN', 39225), ('NNP', 38361), ('DT', 34471), ('VBZ', 23799)]
```

In [21]:

Grade cell: `cell-1ea3284952623d77` Score: 1.0 / 1.0

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 corr 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 sa 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 [22]:

```
from nltk.corpus import words

correct_spellings = words.words()
```

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.

Refer to:

- NLTK Jaccard distance (https://www.nltk.org/api/nltk.metrics.distance.html?highlight=jaccard_distance#nltk.metrics.distance.jaccard_distance)
- NLTK ngrams (https://www.nltk.org/api/nltk.util.html?highlight=ngrams#nltk.util.ngrams)

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

In [23]:

Student's answer

```
def answer_nine(entries=['cormulent', 'incendenece', 'validrate']):
    from nltk.metrics.distance import (
    jaccard_distance,
    )
    from nltk.util import ngrams
    spellings_series = pd.Series(correct_spellings)
    correct = []
    for entry in entries :
        spellings = spellings_series[spellings_series.str.startswith(entry[0])]
        distances = ((jaccard_distance(set(ngrams(entry, 3)),set(ngrams(word, 3))), word) for word in spellings)
        closet = min(distances)
        correct.append(closet[1])

    return correct

answer_nine()
```

Out[23]:

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

In [24]:

Grade cell: `cell-2a2de5b65d33eeeb` Score: 1.0 / 1.0

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.

Refer to:

- NLTK Jaccard distance (https://www.nltk.org/api/nltk.metrics.distance.html?highlight=jaccard_distance#nltk.metrics.distance.jaccard_distance)
- NLTK ngrams (https://www.nltk.org/api/nltk.util.html?highlight=ngrams#nltk.util.ngrams)

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

In [25]:

Student's answer

```
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)))) f

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

    return result # Your answer here

answer_ten()
```

Out[25]:

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

In [26]:

Grade cell: `cell-7a14f4e02e15bfa2` Score: 1.0 / 1.0

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. (https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance)

Refer to:

- NLTK edit distance (https://www.nltk.org/api/nltk.metrics.distance.html?highlight=edit_distance#nltk.metrics.distance.edit_distance)

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

In [27]:

Student's answer

```
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[27]:

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

In [28]:

Grade cell: `cell-152ee7cd1d36928c` Score: 1.0 / 1.0

This assignment was graded by mooc_adswpy:5a1483384bca, v1.47.103123