Assignment 2

Alexis Jennings

aej190000

CS 4395.001

```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('book')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Package conll2000 is already up-to-date!
```

```
[nltk_data]     | Downloading package conll2002 to /root/nltk_data...
[nltk_data]     |   Package conll2002 is already up-to-date!
[nltk_data]     | Downloading package dependency_treebank to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Package dependency_treebank is already up-to-date!
[nltk_data]     | Downloading package genesis to /root/nltk_data...
[nltk_data]     |   Package genesis is already up-to-date!
[nltk_data]     | Downloading package gutenberg to /root/nltk_data...
[nltk_data]     |   Package gutenberg is already up-to-date!
[nltk_data]     | Downloading package ieer to /root/nltk_data...
[nltk_data]     |   Package ieer is already up-to-date!
[nltk_data]     | Downloading package inaugural to /root/nltk_data...
[nltk_data]     |   Package inaugural is already up-to-date!
[nltk_data]     | Downloading package movie_reviews to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Package movie_reviews is already up-to-date!
[nltk_data]     | Downloading package nps_chat to /root/nltk_data...
[nltk_data]     |   Package nps_chat is already up-to-date!
[nltk_data]     | Downloading package names to /root/nltk_data...
[nltk_data]     |   Package names is already up-to-date!
[nltk_data]     | Downloading package ppattach to /root/nltk_data...
[nltk_data]     |   Package ppattach is already up-to-date!
[nltk_data]     | Downloading package reuters to /root/nltk_data...
[nltk_data]     |   Package reuters is already up-to-date!
[nltk_data]     | Downloading package senseval to /root/nltk_data...
[nltk_data]     |   Package senseval is already up-to-date!
[nltk_data]     | Downloading package state_union to /root/nltk_data...
[nltk_data]     |   Package state_union is already up-to-date!
[nltk_data]     | Downloading package stopwords to /root/nltk_data...
[nltk_data]     |   Package stopwords is already up-to-date!
[nltk_data]     | Downloading package swadesh to /root/nltk_data...
[nltk_data]     |   Package swadesh is already up-to-date!
[nltk_data]     | Downloading package timit to /root/nltk_data...
[nltk_data]     |   Package timit is already up-to-date!
[nltk_data]     | Downloading package treebank to /root/nltk_data...
[nltk_data]     |   Package treebank is already up-to-date!
[nltk_data]     | Downloading package toolbox to /root/nltk_data...
[nltk_data]     |   Package toolbox is already up-to-date!
```

The `tokens()` method extracts tokens from text. To get the first 20 tokens, I replaced `()` with `[:20]`. The output is a list of the first 20 tokens from text1. Text objects have various methods besides `tokens()` to aid with processing, including `concordance()` and

`collocations()`. Text objects are also indexed, and the `index()` method can be used to find the index of the first occurrence of a given word.

```
from nltk.book import *
text1.tokens[:20]
```

```
['[',
 'Moby',
 'Dick',
 'by',
 'Herman',
 'Melville',
 '1851',
 ']',
 'ETYMOLOGY',
 '.',
 '(',
 'Supplied',
 'by',
 'a',
 'Late',
 'Consumptive',
 'Usher',
 'to',
 'a',
 'Grammar']
```

The `concordance()` method returns a list of occurrences of a given word in a given text. This concordance looks for the word "sea" in text1.

```
text1.concordance("sea",79,5)
```

```
Displaying 5 of 455 matches:
 shall slay the dragon that is in the sea ." -- ISAIAH " And what thing soever
 S PLUTARCH ' S MORALS . " The Indian Sea breedeth the most and the biggest fis
cely had we proceeded two days on the sea , when about sunrise a great many Wha
```

```
many Whales and other monsters of the sea , appeared . Among the former , one w
 waves on all sides , and beating the sea before him into a foam ." -- TOOKE '
```

The `count()` method tallies up the number of occurrences of some value in a given list of tokens. It only counts the exact and full value provided, and not parts of tokens. Python's `count()` method works the same way, counting the number of occurrences of some value in a given list. They give the same result for "sea" below.

```
print(text1.count('sea')) # using Text.count()
words = text1.tokens
print(words.count('sea')) # using Python's count()
```

```
433
433
```

`word_tokenize()` is a method that takes some text and turns its words into tokens. Here, I took a passage from Shakespeare's *Hamlet*, tokenized it, and printed the first 10 tokens.

References

[1] W. Shakespeare, P. Edwards, and W. Shakespeare, Hamlet, Prince of Denmark. Cambridge: Cambridge University Press, 2019.

```
from nltk import word_tokenize
raw_text = 'To be, or not to be, that is the question: Whether 'tis nobler in the mind to suffer The slings and ar
tokens = word_tokenize(raw_text)
tokens[:10]
```

```
['To', 'be', ',', 'or', 'not', 'to', 'be', ',', 'that', 'is']
```

`sent_tokenize()` is a method that takes some text and turns its sentences into tokens. Here, I took a passage from Shakespeare's *Hamlet*, tokenized it, and printed the tokenized sentences.

```
from nltk import sent_tokenize
sentences = sent_tokenize(raw_text)
sentences
```

```
['To be, or not to be, that is the question: Whether 'tis nobler in the mind to suffer The slings and arrows
of outrageous fortune, Or to take arms against a sea of troubles And by opposing end them.',
 'To die—to sleep, No more; and by a sleep to say we end The heart-ache and the thousand natural shocks That
flesh is heir to: 'tis a consummation Devoutly to be wish'd.',
 'To die, to sleep; To sleep, perchance to dream—ay, there's the rub: For in that sleep of death what dreams
may come, When we have shuffled off this mortal coil, Must give us pause—there's the respect That makes
calamity of so long life']
```

`stem()` gets the stems of a list of word tokens. The list comprehension performs the stemming.

```
porter = nltk.PorterStemmer()
[porter.stem(t) for t in tokens]
```

```
 ',',
 'd',
 '.',
 'to',
 'die',
 ',',
 'to',
 'sleep',
 ';',
 'to',
 'sleep',
 ',',
 'perchanc',
 'to',
 'dream—ay',
 ',',
 'there',
 ''',
 's',
 'the',
 'rub',
 ':',
 'for',
 'in',
 'that',
 'sleep',
 'of',
```

```
 'death',
 'what',

 'dream',
 'may',
 'come',
 ',',
 'when',
 'we',
 'have',
 'shuffl',
 'off',
 'thi',
 'mortal',
 'coil',
 ',',
 'must',
 'give',
 'us',
 'pause—ther',
 '',
 's',
 'the',
 'respect',
 'that',
 'make',
 'calam',
 'of',
 'so',
 'long',
 'life']
```

`lemmatize()` gets the lemmatization of a list of word tokens. The list comprehension performs the lemmatizing.

Differences between `stem()` and `lemmatize()`:

outrag-outrage

fortun-fortune

troubl-trouble

oppos-opposing

natur-natural

```
wnl = nltk.WordNetLemmatizer()
[wnl.lemmatize(t) for t in tokens]
```

```
['To',
 'be',
 ',',
 'or',
 'not',
 'to',
 'be',
 ',',
 'that',
 'is',
 'the',
 'question',
 ':',
 'Whether',
 '',
 'ti',
 'nobler',
 'in',
 'the',
 'mind',
 'to',
 'suffer',
 'The',
 'sling',
 'and',
 'arrow',
 'of',
 'outrageous',
 'fortune',
 ',',
 'Or',
 'to',
 'take',
```

```
'arm',
'against',
'a',
'sea',
'of',
'trouble',
'And',
'by',
'opposing',
'end',
'them',
'.',
'To',
'die—to',
'sleep',
',',
'No',
'more',
';',
'and',
'by',
'a',
'sleep',
'to',
'say',
```

The NLTK library is useful for separating and processing text. However, the stemmer and lemmanizer are not super useful, as the stemmer is too harsh, and the lemmanizer is too lax. The code is well documented and easy to follow. The names of the methods also make it easy to remember what the method does. NLTK would make text processing easier in future projects.