Assignment 3: Exploring NLTK

Overview:

This assignment practices using features of NLTK and and allows for the examination of a professional-level-NLP API

Instruction 2:

As per instruction 2, the following code chunk downloads nltk book and installs additional items needed to use the Google Colab environment properly.

In [ ]:
```
import nltk
nltk.download('book')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
```

Instruction 3:

Two things that I learned from looking at the code for the Text object on NLTK's website is that Text objects aren't used for analysis. They are used to store information like counts and concordance. Analysis must be done using various functions. I also learned that Text objects contain an attribute that is a list called tokens. These tokens are index addressable.

The following code imports everything from nltk.book and extracts the first 20 tokens from text1.

In [5]:
```
from nltk.book import *
print(text1.tokens[:20])
```

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

Instruction 4:

The following code chunk examines the concordance() method when called against text1. The concordance() method locates a specified word within the text and returns the line containing that word. In this case, the method takes 3 arguments: the specified word, which is "sea", the width of the line returned, which is set to 79, and the number of lines desires, which is 5.

In [8]:
```
print(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 '
None
```

Instruction 5:

Here, NLTK's count() method is explored.

When compared to Python's count() method, it looks to be very similar. Both take an argument and return an integer that represents the number of times that argument occurs in a list. The main difference between the 2 is that Python's count() can take a lot of different data types; it also has an optional start and end index that NLTK's count() does not appear to have. The error: "TypeError: count() takes 2 positional arguments but 4 were given" is thrown.

In [32]:
```
word1 = 'sea'
sentence1 = 'The sea sea sea sea sea the sea sea'
# use NLTK count()
print("The word {} appears {} times in text1.".format(word1, text1.count(word1)))
# use Python count()
print("The word {} appears {} times in sentence1.".format(word1, sentence1.count("sea")))
print("The word {} appears {} times in sentence1 from indexes 0 to 20.".format(word1, sentence1.
# try NLTK count() with indices
print("The word {} appears {} times in text1 from index {} to {}".format(word1, text1.count(word
```

```
The word sea appears 433 times in text1.
The word sea appears 7 times in sentence1.
The word sea appears 4 times in sentence1 from indexes 0 to 20.
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-a7a82b1465b7> in <module>
      7 print("The word {} appears {} times in sentence1 from indexes 0 to 20.".format(word1, se
ntence1.count("sea", 0, 20)))
      8 # try NLTK count() with indices
----> 9 print("The word {} appears {} times in text1 from index {} to {}".format(word1, text1.co
unt(word1, 0,1000)))

TypeError: count() takes 2 positional arguments but 4 were given
```

Instruction 6:

This code chunk takes the intro of chapter 4 in our Natural Language Processing text book. The intro is stored in a variable called "raw_text" and tokenized into words using NLTK's word tokenizer.

Book citation in IEEE:

K. Mazidi, "4. Linguistics 101," in Exploring NLP with Python: Building Understanding Through Code, First., 2019, p. 49.

In [33]:
```
from nltk.corpus.reader.tagged import word_tokenize
raw_text = """ Linguistics and NLP are closely bound together. In fact, NLP is sometimes called C
Linguistics. Linguistics is the study of human language, and is a fascinating field of study. Ma
universities offer advanced degrees in Linguistics. The goal of this chapter is more modest: To
familiarize the reader with terminology and concepts that are frequently used in NLP. Language is
complex. Therefore, it is not surprising that linguists study language from so many aspects, fro
letters that make up written words, the sounds that make up spoken words, sentence structure, mea
culture, and more. To further linguistics literacy, the next few paragraphs explore terminology t
every NLP practitioner should understand."""

tokens = word_tokenize(raw_text)
print(tokens[:10])
```

```
['Linguistics', 'and', 'NLP', 'are', 'closely', 'bound', 'together', '.', 'In', 'fact']
```

Instruction 7:

In the next code chunk, the same raw text from above is tokenized using NLTK's sentence tokenizer and then displayed.

In [35]:
```
sentences = nltk.sent_tokenize(raw_text)
for sentence in sentences:
        print(sentence)
```

```
 Linguistics and NLP are closely bound together.
In fact, NLP is sometimes called Computational
Linguistics.
Linguistics is the study of human language, and is a fascinating field of study.
Many
universities offer advanced degrees in Linguistics.
The goal of this chapter is more modest: To
familiarize the reader with terminology and concepts that are frequently used in NLP.
Language is
complex.
Therefore, it is not surprising that linguists study language from so many aspects, from the
letters that make up written words, the sounds that make up spoken words, sentence structure, me
aning,
culture, and more.
To further linguistics literacy, the next few paragraphs explore terminology that
every NLP practitioner should understand.
```

Instruction 8:

This area uses NLTK's PorterStemmer() to stem the raw text from above.

In [36]:
```
from nltk import PorterStemmer
stemmer = PorterStemmer()
stemmed = [stemmer.stem(word) for word in tokens]
print(stemmed)
```

```
['linguist', 'and', 'nlp', 'are', 'close', 'bound', 'togeth', '.', 'in', 'fact', ',', 'nlp', 'i
s', 'sometim', 'call', 'comput', 'linguist', '.', 'linguist', 'is', 'the', 'studi', 'of', 'huma
n', 'languag', ',', 'and', 'is', 'a', 'fascin', 'field', 'of', 'studi', '.', 'mani', 'univers',
'offer', 'advanc', 'degre', 'in', 'linguist', '.', 'the', 'goal', 'of', 'thi', 'chapter', 'is',
'more', 'modest', ':', 'to', 'familiar', 'the', 'reader', 'with', 'terminolog', 'and', 'concep
t', 'that', 'are', 'frequent', 'use', 'in', 'nlp', '.', 'languag', 'is', 'complex', '.', 'theref
or', ',', 'it', 'is', 'not', 'surpris', 'that', 'linguist', 'studi', 'languag', 'from', 'so', 'm
ani', 'aspect', ',', 'from', 'the', 'letter', 'that', 'make', 'up', 'written', 'word', ',', 'th
e', 'sound', 'that', 'make', 'up', 'spoken', 'word', ',', 'sentenc', 'structur', ',', 'mean',
',', 'cultur', ',', 'and', 'more', '.', 'to', 'further', 'linguist', 'literaci', ',', 'the', 'ne
xt', 'few', 'paragraph', 'explor', 'terminolog', 'that', 'everi', 'nlp', 'practition', 'should',
'understand', '.']
```

Instruction 9:

This area uses NLTK's WordNetLemmatizer() to lemmatize the raw text from above.

When the PorterStemmer() and WordNetLemmatizer() are compared, there are quite a few differences. A lot of the stemmed words dont make sense. Here are a few noticable differences:

Stem --------- Lem

togeth ------- together

sometim ----sometimes

studi----------study

languag------language

fascin---------fascinating

In [39]:
```
from nltk import WordNetLemmatizer
lemmer = WordNetLemmatizer()
lemmed = [lemmer.lemmatize(word) for word in tokens]
print(lemmed)
```

```
['Linguistics', 'and', 'NLP', 'are', 'closely', 'bound', 'together', '.', 'In', 'fact', ',', 'NL
P', 'is', 'sometimes', 'called', 'Computational', 'Linguistics', '.', 'Linguistics', 'is', 'th
e', 'study', 'of', 'human', 'language', ',', 'and', 'is', 'a', 'fascinating', 'field', 'of', 'st
udy', '.', 'Many', 'university', 'offer', 'advanced', 'degree', 'in', 'Linguistics', '.', 'The',
'goal', 'of', 'this', 'chapter', 'is', 'more', 'modest', ':', 'To', 'familiarize', 'the', 'reade
r', 'with', 'terminology', 'and', 'concept', 'that', 'are', 'frequently', 'used', 'in', 'NLP',
'.', 'Language', 'is', 'complex', '.', 'Therefore', ',', 'it', 'is', 'not', 'surprising', 'tha
t', 'linguist', 'study', 'language', 'from', 'so', 'many', 'aspect', ',', 'from', 'the', 'lette
r', 'that', 'make', 'up', 'written', 'word', ',', 'the', 'sound', 'that', 'make', 'up', 'spoke
n', 'word', ',', 'sentence', 'structure', ',', 'meaning', ',', 'culture', ',', 'and', 'more',
'.', 'To', 'further', 'linguistics', 'literacy', ',', 'the', 'next', 'few', 'paragraph', 'explor
e', 'terminology', 'that', 'every', 'NLP', 'practitioner', 'should', 'understand', '.']
```

Instruction 10:

Since a computer can't read and work with raw text, having a set of tools to easily convert that text into a format that machines can use is very handy. NLTK is a library for just that. NLTK quickly and efficiently tokenizes both words and sentences. It was strange that new lines were included in the sentence tokenization:

"Linguistics. Linguistics is the study of human language, and is a fascinating field of study. Many"

Both the stemmer and lemmatizer are effective as well, though stemming seems to lose a lot of the original meaning of the text. In the future, the NLTK library can help break up the text into usable chunks. It can assign parts of speech to those chunks. It can be used to create word clouds and more! The NLTK library will be very helpful in moving forward because it can preprocess text data so that the computer can use it for Natural Language Processing.