Amanuel Shiferaw CS 4395.001 Professor Karen Mazidi September 11, 2022 Exploring NLTK: Portfolio Assignment

```
import nltk
from nltk.book import text1
```

```
    *** Introductory Examples for the NLTK Book ***
    Loading text1, ..., text9 and sent1, ..., sent9
    Type the name of the text or sentence to view it.
    Type: 'texts()' or 'sents()' to list the materials.
    text1: Moby Dick by Herman Melville 1851
    text2: Sense and Sensibility by Jane Austen 1811
    text3: The Book of Genesis
    text4: Inaugural Address Corpus
    text5: Chat Corpus
    text6: Monty Python and the Holy Grail
    text7: Wall Street Journal
    text8: Personals Corpus
    text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Download Dependencies

```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('treebank')
nltk.download('webtext')
```

```
    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Unzipping corpora/stopwords.zip.
    [nltk_data] Downloading package wordnet to /root/nltk_data...
    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Unzipping tokenizers/punkt.zip.
    [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] Downloading package gutenberg to /root/nltk_data...
    [nltk_data]   Unzipping corpora/gutenberg.zip.
    [nltk_data] Downloading package genesis to /root/nltk_data...
    [nltk_data]   Unzipping corpora/genesis.zip.
    [nltk_data] Downloading package inaugural to /root/nltk_data...
    [nltk_data]   Unzipping corpora/inaugural.zip.
    [nltk_data] Downloading package nps_chat to /root/nltk_data...
```

```
[nltk_data]   Unzipping corpora/nps_chat.zip.
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data]   Unzipping corpora/webtext.zip.
True
```

Extract the first 20 tokens Two things learned about the tokens API:

1) The list of tokens for the document are already stored in “_token” variable as a list of string. Therefore, the function call simply returns the already stored list of strings and does not parse/process the document when the function is called.

2) The text objects contains api calls that allow us to get the frequency of a word in a text and an the first occurrence of a word in a document by making use of python default methods.

```
text1.tokens[:20]
```

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

Concordance method

```
print(text1.concordance("sea", lines = 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
```

Count API and python built in method. Comparison: The count() API makes use of the python count method for lists to count the frequency of a word in the text by first tokenizing it.

```
print(text1.count('Moby'))
print(text1.tokens.count('Moby'))
```

```
    84
    84
```

Write short sentences. Home. (n.d.). Retrieved September 11, 2022, from https://www.plainlanguage.gov/guidelines/concise/write-short-sentences/

```
raw_text = "Express only one idea in each sentence. Long, complicated sentences often
```

```
from nltk.tokenize import word_tokenize
original_tokens = word_tokenize(raw_text)
tokens = original_tokens[:10]
print(tokens)
```

```
    ['Express', 'only', 'one', 'idea', 'in', 'each', 'sentence', '.', 'Long', ',']
```

```
from nltk.corpus.reader.tagged import sent_tokenize
sentences = sent_tokenize(raw_text)
print(sentences)
```

```
    ['Express only one idea in each sentence.', 'Long, complicated sentences often m
```

Differences between stemming and lemmatizing:

1) Stemming may result in words that do not have meaning while lemma maintains the meaning of words after being altered

2) Stemming removes all ‘s’ at the end of a sentence while lemma conditionally removes the `s` at the end of a word if it maintains its meaning after alteration

3) Stemming may result with words with incorrect spelling, but lemmas understand the context of words and converts them to base form

4) Stemming depends on heuristics to alter the words in a document but lemma depends on the analysis of the meaning on words to alter the words

5) Stemming cannot decipher between a word that can have multiple meaning depending on the

```
from nltk.stem.porter import *
stemmer = PorterStemmer()
stemmedList = [stemmer.stem(word) for word in original_tokens]
print(stemmedList)
```

```
['express', 'onli', 'one', 'idea', 'in', 'each', 'sentenc', '.', 'long', ',', 'c
```

```
from nltk.stem import WordNetLemmatizer
lemmer = WordNetLemmatizer()
lemmatizedList = [lemmer.lemmatize(word) for word in original_tokens]
print(lemmatizedList)
```

```
['Express', 'only', 'one', 'idea', 'in', 'each', 'sentence', '.', 'Long', ',', '
```

Write a paragraph outlining: a. your opinion of the functionality of the NLTK library b. your opinion of the code quality of the NLTK library c. a list of ways you may use NLTK in future projects

In my opinion, the NLTK library has extensive functionality to process large texts and is straigtforward to use. It contains functionality to parse, tokenize, classify, stem, and other functionalities for users. A lot of the methods abstract processes to make them easy and convenient to use by the user. For instance, the count API helps the user to retrieve the tokens of the text object and then call the python count method on the list to count the frequency of a word in a text. Instead, it bundles that process together into a count API that retrieves and calls the count python method. Some of the things I was thinking of using NLTK for were sentiment analysis, removing stop words, stemming a body of text, lemmatizing a body of text, finding the frequency of a word in a text, and so on.