```
import nltk
nltk.download("stopwords")
nltk.download("all")

from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import *
```

```
[nltk_data]    | Downloading package subjectivity to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package subjectivity 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 switchboard to /root/nltk_data...
[nltk_data]    |   Package switchboard is already up-to-date!
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Package tagsets 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 toolbox to /root/nltk_data...
[nltk_data]    |   Package toolbox 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 twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Package udhr is already up-to-date!
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package unicode_samples is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package universal_tagset is already up-to-date!
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package universal_treebanks_v20 is already up-to-
[nltk_data]    |       date!
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package vader_lexicon is already up-to-date!
[nltk_data]    | Downloading package verbnet to /root/nltk_data...
[nltk_data]    |   Package verbnet is already up-to-date!
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]    |   Package verbnet3 is already up-to-date!
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Package webtext is already up-to-date!
[nltk_data]    | Downloading package wmt15_eval to /root/nltk_data...
[nltk_data]    |   Package wmt15_eval is already up-to-date!
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data...
```

```
[nltk_data]    |   Package word2vec_sample 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 wordnet2021 to /root/nltk_data...
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Package ycoe is already up-to-date!
[nltk_data]    |
```

```
from nltk.book import *
```

Tokenize takes text and divides into substrings where each word, number, and special character is its own token.

Tokenizing a string is useful to generate statistics on the text such as the frequency of certain words, remove plurality from individual words, remove punctuation from text, ect.

```
text1.tokens[0:20]
```

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

Calls concordance method for "sea" and lists first 5 lines

```
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 '
```

Both count methods count the number of times a specific substring occurs in a string. NLTK's count differs in that it is counting tokens which match with the list of tokens produced by tokenizing the string. So matching fails if the NLTK count is provided a string that is not a token, such as "sea ", which is not a token due to the space at the end. Pythons count is simply matching substrings, so this stipulation does not exist. In the below example, I use nltk's count for "sea" and "sea ", adding a space such that the count search is not a token. Notice how the nltk count returns 433 for the token, and 0 when a space is added. Then for pythons count, I created some text and searched for both "them" and "them ", and the same result was returned regardless of the space.

```
print(text1.count("sea"))
print(text1.count("sea "))
x = "Voldemort himself created his worst enemy, just as tyrants everywhere do! Have you any i
print(x.count("them"))
print(x.count("them "))
```

```
433
0
2
2
```

Raw_text is tokenizes and the results are saved into x, and the first ten results are printed.

```
raw_text = 'Voldemort himself created his worst enemy, just as tyrants everywhere do! Have yo
x = word_tokenize(raw_text)
print(x[:10])
```

```
['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'as', 'tyrants
```

used nltk.sent_tokenize on the previously defined raw_text

list comprehension using nltk's porter stemmer

```
stemmer = PorterStemmer()
stems = [stemmer.stem(z) for z in x]
```

```
stems


     ['voldemort',
      'himself',
      'creat',
      'hi',
      'worst',
      'enemi',
      ',',
      'just',
      'as',
      'tyrant',
      'everywher',
      'do',
      '!',
      'have',
      'you',
      'ani',
      'idea',
      'how',
      'much',
      'tyrant',
      'fear',
      'the',
      'peopl',
      'they',
      'oppress',
      '?',
      'all',
      'of',
      'them',
      'realiz',
      'that',
      ',',
      'one',
      'day',
      ',',
      'amongst',
      'their',
      'mani',
      'victim',
      ',',
      'there',
      'is',
      'sure',
      'to',
      'be',
      'one',
      'who',
      'rise',
      'against',
      'them',
      'and',
      'strike',
```

```
      'back',
      '''
sent_tokenize(raw_text)
```

```
['Voldemort himself created his worst enemy, just as tyrants everywhere do!',
 'Have you any idea how much tyrants fear the people they oppress?',
 'All of them realize that, one day, amongst their many victims, there is sure to be
one who rises against them and strikes back!']
```

Used word net lemmatizer. First defined and instance of wordnetlemmatizer as wnl

```
wnl = WordNetLemmatizer()
lems = [wnl.lemmatize(z) for z in x]
print(lems)
```

```
['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'a', 'tyrant',
```

I found the NLTK library to be highly functional in regards to string manipulation, generating statistical information from strings, and overall ease of use. The library is very simple to understand and the code quality and documentation in the NLTK library facilitates smooth operation. Going forward, I hope to use NLTK to convert text data into a more easily processable form, elimate irellevant bits, and implement NLP techniques


0s
completed at 4:55 PM