Downloading nltk assets

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

```
[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 package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg 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 inaugural to /root/nltk_data...
[nltk_data]   Package inaugural 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 webtext to /root/nltk_data...
[nltk_data]   Package webtext is already up-to-date!
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
True
```

Token extraction

Extracting tokens from text1 and printing first 20

Things I learned:

1. A Text object doesn’t have a tokens() method, but instead a tokens attribute
2. A Text object is actually a wrapper for the tokens/text and provides methods for some initial analysis of the text, however it only displays the results of these methods. If a project needs to use the analysis methods in more depth, it’s recommended to skip the Text object and work with the analysis function directly.

```
from nltk.book import *
tokens = [token for token in text1.tokens]
tokens[:20]
```

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

Concordance method

Prints a concordance (every occurence of a given word, plus some context) of 'sea' in text1

```
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 '
```

Count method

Using the count method to retrieve the number of times the token "sea" occurs within text1

Observations of count method in nltk vs default python:

- The method uses Python’s count method for lists on self.token to determine how many times a certain word exists in the text. This makes it basically identical to Python’s count method for lists, however unlike the count method for strings, the user has no flexibility in terms of start and end position of the string that will be checked for the word. Additionally, in the string version of count(), any substring of a word that matches the given argument will also count (see the string version of count() cell below)

```
text1.count('sea')
```

```
433
```

```
# String version of count()
sample_text = "Man I sure love seattle, right next to the ocean, or was it a sea? I don't remember"
sample_text.count('sea')
```

```
2
```

Word Tokenizing Harry Potter and the Half-Blood Prince

Tokenizing an excerpt from Harry Potter and the Half-Blood Prince using word tokenization and then printing the first ten tokens

```
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!'

tokens = nltk.word_tokenize(raw_text)

tokens[:10]
```

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

Sentence Tokenizing Harry Potter and the Half-Blood Prince

Tokenizing an excerpt from Harry Potter and the Half-Blood Prince using sentence tokenization and then printing all tokens

```
tokens = nltk.sent_tokenize(raw_text)

tokens
```

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

Using NLTK's PorterStemmer()

Stemming the same raw text using NLTK's PorterStemmer() object

```
from nltk import stem

stemmer = stem.PorterStemmer()
stemmed = [stemmer.stem(word) for word in nltk.word_tokenize(raw_text)]

stemmed
```

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

Using NLTK's WordNetLemmatizer

Lemmatizing the same raw text using NLTK's WordNetLemmatizer() Object

5 differences between stems vs lemmas (given as stem-lemma):

- creat-created
- hi-his
- enemi-enemy
- everywher-everywhere
- ani-any

```
lemmatizer = stem.WordNetLemmatizer()

lemmatized = [lemmatizer.lemmatize(word) for word in nltk.word_tokenize(raw_text)]

lemmatized
```

```
     ['Voldemort',
      'himself',
      'created',
      'his',
      'worst',
      'enemy',
      ',',
      'just',
      'a',
      'tyrant',
      'everywhere',
      'do',
      '!',
      'Have',
      'you',
      'any',
      'ideahow',
      'much',
      'tyrant',
      'fear',
      'the',
      'people',
      'they',
      'oppress',
      '?',
      'All',
      'of',
      'them',
      'realize',
      'that',
      ',',
      'one',
      'day',
      ',',
      'amongsttheir',
      'many',
      'victim',
      ',',
      'there',
      'is',
      'sure',
      'to',
      'be',
      'one',
      'who',
      'rise',
      'against',
      'them',
      'and',
      'strike',
      'back',
      '!']
```

Discussing NLTK

It seems that the NLTK library so far can function well as a text pre-processor, and the functions are granular enough that it won't interfere in any future projects. The functions and documentation is all very clean, and the outputs are expected, so I would say overall that the code quality of the library is quite high. I can see myself using this to process and tokenize text as a pre-processing step before encoding the tokens for input into a model, or to gain insight on certain quirks about a text using things like the concordance() method.