NLTK Exploration

This is a brief exploration into some of the common methods and topics used in the nltk tool kit that we'll be using in this NLP course (CS 4395)

Imports and Downloads

Importing and installing necessary libraries for nltk

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

```
[nltk_data] Downloading package stopwords to /Users/josh/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/josh/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /Users/josh/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /Users/josh/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

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

Tokens Method and Text objects

1. The tokens method will return a list of the tokens in the text.
2. Text objects can return specified tokens or the whole text in the form of a list of tokens by calling them or using the [:n] format where n is the number of tokens to return

This code uses the concept described in 2. to print the first 10 tokens of the token list for the Text object text1

```
text1[:20]
```

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

Concordance Method

This code prints a concordance of text1 using the word sea only showing 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 soeve
 S PLUTARCH ' S MORALS . " The Indian Sea breedeth the most and the biggest f
cely had we proceeded two days on the sea , when about sunrise a great many W
many Whales and other monsters of the sea , appeared . Among the former , one
 waves on all sides , and beating the sea before him into a foam ." -- TOOKE
```

Count Method

The count() method counts the number of times the parameter word appears in the text. Looking at the API it uses the tokens as a list to apply the python count method looking for the parameter word in that list and then returns the count from that method. So In essence the count() method uses the python count method

As shown below the method will return the count of a specific word in the Text object

```
text1.count(word = 'dragon')
```

```
5
```

```
text1.count(word = 'the')
```

```
13721
```

```
text1.count(word = 'applesauce')
```

```
0
```

Tokenization of Raw Text

This code uses the word_tokenize method from nltk to tokenize some raw text (source = "Bee Movie") and then prints the first 10 tokens of the generated token list

```
from nltk import word_tokenize
raw_text = "According to all known laws of aviation, there is no way a bee should b
tokens = word_tokenize(raw_text)
tokens[:10]
```

```
['According',
 'to',
 'all',
 'known',
 'laws',
 'of',
 'aviation',
 ',',
 'there',
 'is']
```

Sentence Segmentation of Raw Text

This code uses the sent_tokenize method from nltk to segement some raw text (source = "Bee Movie") into sentences and then prints the generated sentence list

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

```
['According to all known laws of aviation, there is no way a bee should be
able to fly.',
 'Its wings are too small to get its fat little body off the ground.',
 "The bee, of course, flies anyway because bees don't care what humans think
is impossible.",
 'Yellow, black.',
 'Yellow, black.',
 'Yellow, black.',
 'Yellow, black.',
 'Ooh, black and yellow!',
 "Let's shake it up a little."]
```

Porter Stemmer

This code uses the Porter Stemmer object from nltk and the stem() method to to generate a list of stemmed tokens from the previously generated tokens list using the raw_text and prints it out using a list comprehension

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

```
['accord',
 'to',
 'all',
 'known',
 'law',
 'of',
 'aviat',
 ',',
 'there',
 'is',
 'no',
 'way',
 'a',
 'bee',
 'should',
 'be',
 'abl',
 'to',
 'fli',
 '.',
 'it',
 'wing',
 'are',
 'too',
 'small',
 'to',
 'get',
 'it',
 'fat',
 'littl',
 'bodi',
 'off',
 'the',
 'ground',
 '.',
 'the',
 'bee',
 ',',
 'of',
 'cours',
 ',',
 'fli',
 'anyway',
 'becaus',
 'bee',
 'do',
 "n't",
 'care',
 'what',
 'human',
 'think',
 'is',
 'imposs',
 '.',
 'yellow',
 ',',
 'black',
 '.',
 'yellow',
 ' '
```

Lemmatizer

This code uses the Lemmatizer object from nltk to to generate a list of lemmatized tokens from the previously generated tokens list using the raw_text and prints it out using a list comprehension

Differences Between Stemmer and Lemmatizer

Usinng the format of stem-lemma here are some difference in the stemmer and lemmatizer for this text:

1. accord-According
2. aviat-aviation
3. abl-able
4. fli-fly
5. it-Its
6. littl-little

```
from nltk.stem import WordNetLemmatizer
from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()
lemmas = [wnl.lemmatize(t) for t in tokens]
lemmas
```

```
     'to',
     'get',
     'it',
     'fat',
     'little',
     'body',
     'off',
     'the',
     'ground',
     '.',
     'The',
     'bee',
     ',',
     'of',
     'course',
     ',',
     'fly',
     'anyway',
     'because',
     'bee',
     'do',
     "n't",
     'care',
     'what',
     'human',
     'think',
     'is',
     'impossible',
     '.',
     'Yellow',
     ',',
     'black',
     '.',
     'Yellow',
     ',',
     'black',
     '.',
     'Yellow',
     ',',
     'black',
     '.',
     'Yellow',
     ',',
     'black',
     '.',
     'Ooh'
```

```
      ',',
      'black',
      'and',
      'yellow',
      '!',
      'Let',
      "'s",
      'shake',
      'it',
      'up',
      'a',
      'little',
      '.']
```

Conclusion

The nltk library is definitely workable in terms of functionality, all of the methods that are present function as they should as have readable and understadnable documentation. Once someone has a grasp on the basics of the library it seems like it would be a pretty straightforward library to use in projects. In terms of the code quality of the library, it seems pretty robust overall and covers a wide breadth of the functions and concepts used in NLP. Looking at it like this, coupled with the functinality previously described, the code quality of the library seems pretty high. The nltk library seems like a really great foundation for a lot of NLP based projects some that I would consider would be something like a fantasy language translator, a twitter or reddit sentiment analysis bot, or a custom social media hashtag identifier.