Portfolio Assingment 3: Exploring NLTK

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


nltk.download('book')
```

1. Extract the first 20 tokens from text1

Two things I learned about tokens() or Text objects:

1. Text objects are initialized with tokens and an optional name as parameters
2. Using the TokenSearcher, we can use regex to search over tokenized strings

```
import nltk.book


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

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

2. Print a concordance for text1 word 'sea', selecting only 5 lines.

```
nltk.book.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 '
```

3. Experiment with Python's count() and nltk's count()

The count method in the Text object API accepts a word parameter anc counts the number of times the word appears in the text object callin it. There is no difference between this method and the built in str.count(substring) method for strings in python

```
string_1 = 'Hello! My name is Savi. Hello!'
string_1.count('Hello')
```

```
2
```

```
text_1 = nltk.Text(nltk.word_tokenize(string_1))
text_1.count('Hello')
```

```
2
```

4. Using raw text of at least 5 sentences of your choice from any source (cite the source), save the text into a variable called raw_text. Using NLTK's word tokenizer, tokenize the text into variable 'tokens'. Print the first 10 tokens.

```
from nltk import word_tokenize
```

```
# Text source: Lelouch vi Britannia from Code Geass
#   -- https://tvtropes.org/pmwiki/pmwiki.php/Quotes/CodeGeass
raw_text = '''
    Attention, entire world! Hear my proclamation: I am Lelouch vi Britannia, Emperor
'''
```

```
tokens = word_tokenize(raw_text)
tokens[:5]
```

```
['Attention', ',', 'entire', 'world', '!']
```

5. Using the same raw text, and NLTK's sentence tokenizer sent_tokenize(), perform sentence segmentation and display the sentences.

```
from nltk import sent_tokenize


sentences = sent_tokenize(raw_text)
sentences
```

```
['\n    Attention, entire world!',
 'Hear my proclamation: I am Lelouch vi Britannia, Emperor of the Holy Britannian Empire and your only ruler!',
 'Schneizel has surrendered to me: as a result of this, I am now in control of both the Damocles and the FLEIJA weapons, and even the Black Knights no longer possess the strength to oppose me now!',
 'If anyone dares to oppose my supreme authority, they shall know the devastating power of the FLEIJAs.',
 'Those who could oppose my military rule no longer exist!',
 'Yes, from this day, from this moment forward, the world belongs to me!',
 'Lelouch vi Britannia commands you: Obey me, subjects!',
 'OBEY ME, WORLD!']
```

6. Using NLTK's PorterStemmer(), write a list comprehension to stem the text. Display the list.

```
from nltk.stem.porter import *


stemmer = PorterStemmer()


[stemmer.stem(t) for t in tokens]
```

```
 'oppos',
 'my',
 'suprem',
 'author',
 ',',
 'they',
 'shall',
 'know',
 'the',
 'devast',
```

```
'power',
'of',
'the',
'fleija',
'.',
'those',
'who',
'could',
'oppos',
'my',
'militari',
'rule',
'no',
'longer',
'exist',
'!',
'ye',
',',
'from',
'thi',
'day',
',',
'from',
'thi',
'moment',
'forward',
',',
'the',
'world',
'belong',
'to',
'me',
'!',
'lelouch',
'vi',
'britannia',
'command',
'you',
':',
'obey',
'me',
',',
'subject',
'!',
'obey',
'me',
',',
'world',
```

7. Using NLTK's WordNetLemmatizer, write a list comprehension to lemmatize the text. Display the list.

Differences between stems and lemmas (stem - lemma):

1. suprem - supreme
2. author - authority
3. devast - devastating
4. militari - military
5. thi - this

```
from nltk.stem.wordnet import WordNetLemmatizer


wn = WordNetLemmatizer()


[wn.lemmatize(t) for t in tokens]
```

```
      'even',
      'the',
      'Black',
      'Knights',
      'no',
      'longer',
      'posse',
      'the',
      'strength',
      'to',
      'oppose',
      'me',
      'now',
      '!',
      'If',
      'anyone',
      'dare',
      'to',
      'oppose',
      'my',
      'supreme',
      'authority',
      ',',
      'they',
      'shall',
      'know',
      'the',
      'devastating',
      'power',
      'of',
      'the',
      'FLEIJAs',
      '.',
      'Those',
      'who',
      'could',
      'oppose',
      'my',
```

```
        'military',
        'rule',
        'no',
        'longer',
        'exist',
        '!',
        'Yes',
        ',',
        'from',
        'this',
        'day',
        ',',
        'from',
        'this',
        'moment',
        'forward',
        ',',
        'the',
        'world',
        'belongs',
```

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

```
# I think that the NLTK is very functional. It can easily extract words,
# sentences and other text features with just a few lines of code.
# Using other programming languages or even just base Python, these processes
# would require loops and complex idnex finding for spaces or
# quotations, but NLTK makes it easy by just calling a function that only
# requires one line of code on out end.

# I think the code quality of the NLTK library is very nice. Looking through
# the API and documentation it seems like the code is well
# organized and the comments in are informative, making it easy to understand
# the back end work of the NLTK functions

# I can see NLTK being very useful in recommender systems for finding a list of
# similar words between reviews or descriptions to then
# provide recommendations based on. Also NLTK can be used in tone detection by
# considering only the puncutation returned from a list of
# tokens. Lastly, NLTK can be used to perform cleaning on a text dataset, say
# maybe removing all punctuation or reducing words to lemmas.
```

✓ 0s completed at 10:21 PM