Double-click (or enter) to edit

Portfolio Assignment 3: Exploring NLTK

```
#Because we are using Google Colab,
#Here are the imports needed for the program
import nltk
nltk.download('all')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
```

```
[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]    |
[nltk_data]  Done downloading collection all
[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!
True
```

Here, we will extract the 1st twenty tokens from the file name 'text1' of the NLTK handbook.

```
from nltk.book import *

text1[ : 20]
```

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

Upon my second look of the Text object, I found that the 'findall' method can also find words that begin with certain letters. This achieved by typing the prefix or sufix portions of the word along with with '.*' to indicate there is room to fill. All of that inside the <> operator.

I have also discovered in tokens that strings that contain letters and integers count as one token.

In this code sample, I will show the first five instandnces of the word 'sea' in the text file text1. By using the concordance method, the 1st 5 limits the maxinum width each word can be adjacent to the speicified word. The 2nd 5 represent the number of outputs this word appears.

```
text1.concordance("sea", 5, 5,)
```

```
     Displaying 5 of 455 matches:
     the sea
     Indian Sea
     the sea
     the sea
     the sea
```

---

5. The count() method in NLTK's API works very different that the count method that Python normally has. For one, NLTK's count() method only increments the counter when the targeted string word appear in the text. The parameter for NLTK's count will accept any acceptable string and can result in a 0 return.

   Python's traditional count method is used to provide the lenght of a dataset container such as lists, dictionaries, tuples, and arrays. The count method can also be used in string; however, Python (and many other langauges) consiter strings leters as individual elements.

In this code block, the variable 'raw_text' is a snippit of one of my childhood book series, Gregor the Overlander. This partical text can be found in the first book of Gregor the Overlander, and its the backbone of the narrative of the series. This is the prophecy quote.

```
from nltk.tokenize import word_tokenize
raw_text = """ Beware, Underlanders, time hangs by a thread.
               The hunters are hunted, white water runs red.
               The Gnawers will strike to extinguish the rest.
               The hope of the hopeless resides in a quest.
```

```
                An Overland warrior, a son of the sun,
                May bring us back light, he may bring us back none.
                But gather your neighbors and follow his call
                Or rats will most surely devour us all.
                Two over, two under, of royal descent,
                Two flyers, two crawlers, two spinners assent.
                One gnawer beside and one lost up ahead.
                And eight will be left when we count up the dead.
                The last who will die must decide where he stands.
                The fate of the eight is contained in his hands.
                So bid him take care, bid him look where he leaps,
                As life may be death and death life again reaps."""

#Tokenizing raw_text
tokens = word_tokenize(raw_text)

#Printing the 1st 10 tokens
print("Here are the first 10 tokens of the prophecy:\n", tokens[:10])
```

```
     Here are the first 10 tokens of the prophecy:
      ['Beware', ',', 'Underlanders', ',', 'time', 'hangs', 'by', 'a', 'thread', '.']
```

```
from nltk.tokenize import sent_tokenize

#Making sentence tokens of raw_text
sentences = sent_tokenize(raw_text)

#printing the sentences
print("Here is each individual sentences of the prophecy:\n")

#Line segmenting each sentences
index = 0
for index  in range( 0 ,len(sentences) ):
  print(sentences[index], "\n")
  index += 1
```

```
     Here is each individual sentences of the prophecy:

      Beware, Underlanders, time hangs by a thread.

     The hunters are hunted, white water runs red.

     The Gnawers will strike to extinguish the rest.

     The hope of the hopeless resides in a quest.

     An Overland warrior, a son of the sun,
                    May bring us back light, he may bring us back none.

     But gather your neighbors and follow his call
                    Or rats will most surely devour us all.
```

```
Two over, two under, of royal descent,
                Two flyers, two crawlers, two spinners assent.

One gnawer beside and one lost up ahead.

And eight will be left when we count up the dead.

The last who will die must decide where he stands.

The fate of the eight is contained in his hands.

So bid him take care, bid him look where he leaps,
                As life may be death and death life again reaps.
```

In this section, we will be stemming raw_text using PorterStemmer() and display our findings

---

```
#1st import PorterStemmer()
from nltk.stem.snowball import PorterStemmer

#Stemming raw_text
stem = PorterStemmer()
stemmed =[stem.stem(tok) for tok in tokens]

print(stemmed)
```

```
['bewar', ',', 'underland', ',', 'time', 'hang', 'by', 'a', 'thread', '.', 'the', 'hunte
```

Now we must find the lemmatize form of our raw_text list.

```
#1st import WordNetLemmatizer() and making the object
from nltk.stem.wordnet import WordNetLemmatizer
WNL = WordNetLemmatizer()

#lemmatizing our tokend raw_text
lematized = [WNL.lemmatize(tok) for tok in tokens]

print(lematized)
```

```
['Beware', ',', 'Underlanders', ',', 'time', 'hang', 'by', 'a', 'thread', '.', 'The', 'h
```

I will use my increditable programming skills to identify

different text between lemamatize and stem versions of raw_text tokens

```
#Determining which has the smallest length
if len(lematized) == len(stemmed):
  print("We can use either one as max range!\n\n")
elif len(lematized) > len(stemmed):
  print("Use stemmed as max range.")
else:
  print("Use lematized as max range.")

print("Left is Lematized words \t\t Right is stemmed\n\n")
for index in range(0, len(stemmed)):

  if lematized[index] != stemmed[index]:
    print(f"{lematized[index] : <15}" ,f"{'not equal to' : ^30}", f"{stemmed[index] : >10}")
```

```
We can use either one as max range!


Left is Lematized words                 Right is stemmed


Beware                   not equal to              bewar
Underlanders             not equal to          underland
The                      not equal to                the
hunted                   not equal to               hunt
The                      not equal to                the
Gnawers                  not equal to              gnawer
The                      not equal to                the
resides                  not equal to              resid
An                       not equal to                 an
Overland                 not equal to           overland
May                      not equal to                may
u                        not equal to                 us
u                        not equal to                 us
But                      not equal to                but
his                      not equal to                 hi
Or                       not equal to                 or
surely                   not equal to              sure
u                        not equal to                 us
Two                      not equal to                two
Two                      not equal to                two
One                      not equal to                one
beside                   not equal to              besid
And                      not equal to                and
The                      not equal to                the
decide                   not equal to              decid
The                      not equal to                the
contained                not equal to            contain
his                      not equal to                 hi
So                       not equal to                 so
```

```
As                    not equal to                as
reaps                 not equal to              reap
```

The following paragraph is deticated to answer these questions.

Your opinion of the functionality of the NLTK library

Your opinion of the code quality of the NLTK library

A list of ways you may use NLTK in future project

The functionality of NLTK library is niche into understanding and reading human language. With this specialized niche, NLTK library can be utilized in unique ways that usually involve human interaction of some form. I do enjouy myself in exploring NLTK library and while the functionality is focuses in data gathering and analysis, it is the usage of that data that can impact a project.

The code quality of NLTK library is tedious at times, in my opionon. While it can be easily imported to all updated python IDEs, there can be instances that the programmer will need to import more assets of NLTK library that normal. Take Google colab for instantces, I needed to code in the following imports to have the operators function:

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

A good aspect of NLTK formating is that all of the outputs for token, stems, and lemmatizer are lists that are easy to travers through

When it come to my future endevors of programming, NLTK library will provide my code an new branch of user interaction. I can branch my code to voice recongition software to any mobile applications I may want to create or use text analysis of my writting form to artifically automate text paragraphs as concept ideas of my writting or to encourage me to write.

Colab paid products - Cancel contracts here

✓ 0s completed at 12:54 PM