Portfolio Assignment2-NLTK-Prachi Patel

September 11, 2022

```
In [1]: import nltk
        nltk.download('stopwords')
        nltk.download('wordnet')
        nltk.download('punkt')
        nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Prachi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Prachi\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Prachi\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Prachi\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
Out[1]: True
```

import the texts from nltk.book

```
In [2]: from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
```

```
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Create an empty list and remove the parantheses and square parantheses. Get the first 20 tokens and print them.

I learned that the tokens method results in a list. It is not a callable method. I also learned that Text objects have many inbuilt method like concordance and token searcher.

```
In [3]: texts = []
        counter = 0
        while len(texts) != 20:
            if (text1.tokens[counter] != '[') and (text1.tokens[counter] != ']') and (text1.to
                texts.append(text1.tokens[counter])
            counter = counter + 1

        print(texts)
```

```
['Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', 'ETYMOLOGY', 'Supplied', 'by', 'a', 'Late
```

Print the concordance, and the first five instances of the word sea with the concordance width being 79 characters.

```
In [4]: print(text1.concordance('sea', width=79, 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 '
None
```

Compare the two count methods

The count method in the API counts the number of times a word occurs in the text object. Python's count method also counts the number of times a word occurs in the string, but it also has an option to narrow the search to certain indices. They both count the number of occurrences of a particular word. The Python count method has more use because it can check the number of occurrences of substrings as well and not just singular words.

```
In [5]: print(text1.count('sea'))

        tex = 'hello, sea. I am the beach.'

        print(tex.count('sea', 0, 10))
```

```
433
1
```

Raw Text taken from Harry Potter and the Chamber of Secrets, Chapter 1 by J.K. Rowling.

```
In [6]: raw_text = "Not for the first time, an argument had broken out over breakfast at numbe
```

import word tokenizer from nltk and tokenize the raw text. Print the first ten tokens

```
In [7]: from nltk import word_tokenize

        tokens = word_tokenize(raw_text)
        print(tokens[0:10])
```

```
['Not', 'for', 'the', 'first', 'time', ',', 'an', 'argument', 'had', 'broken']
```

import the sentence tokenizer from nltk and tokenize the raw text by sentences. Print the sentences.

```
In [8]: from nltk import sent_tokenize

        sent = sent_tokenize(raw_text)
        print(sent)
```

```
['Not for the first time, an argument had broken out over breakfast at number four, Privet Dri
```

import the porter stemmer and used a list comprehension to stem each token in tokens. Printed results

```
In [9]: from nltk.stem.porter import *

        stemmer = PorterStemmer()
        stemmed = [stemmer.stem(tok) for tok in tokens]
        print(stemmed)
```

```
['not', 'for', 'the', 'first', 'time', ',', 'an', 'argument', 'had', 'broken', 'out', 'over',
```

import the word net lemmatizer and used a list comprehension to lemmatize each token in tokens. Printed results

Stemm-Lemmatizer earli-early morn-morning hoot-hooting nois-noise hi-his

```
In [10]: from nltk.stem import WordNetLemmatizer

         wordNetLem = WordNetLemmatizer()
         lemmatized = [wordNetLem.lemmatize(tok) for tok in tokens]
         print(lemmatized)
```

```
['Not', 'for', 'the', 'first', 'time', ',', 'an', 'argument', 'had', 'broken', 'out', 'over',
```

Opinion on functionality of NLTK library: I believe that the NLTK library is quite functional when thinking about text objects. It's many algorithms such as the Porter Stemmer and the Word Net Lemmatizer, allows the user to easily preprocess the text data. It can be used to tokenize the data into either words or sentences, and this allows the user to use the NLTK library to preprocess the data more easily and perform functions on the data more easily.

Opinion of the code quality of the NLTK library: The code quality of the NLTK library is high. Although the code itself is confusing to understand sometimes, the code in the NLTK library is of high quality and has each and every function given with comments explaining the code and it's purpose. Thus, this gives the code a high quality as it is easier to understand with the given comments.

List of ways I can use NLTK in future projects: –Preprocess/tokenize data –Filter out stop words –Stemming –Part of Speech tagging –Lemmatizing