assignment2

September 11, 2022

import the NLTK library and the needed helper functions, classes

[98]:
```
import nltk
from nltk.book import *
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
```

1. NLTK text objects are already pre-tokenized
2. NLTK text objects come with some nice built-in-functions similar to python std such as count() and len()

Display the first 20 tokens of text1

[99]:
```
text1[:20]
```

[99]:
```
['[',
 'Moby',
 'Dick',
 'by',
 'Herman',
 'Melville',
 '1851',
 ']',
 'ETYMOLOGY',
 '.',
 '(',
 'Supplied',
 'by',
 'a',
 'Late',
 'Consumptive',
 'Usher',
 'to',
 'a',
 'Grammar']
```

Display the first 5 lines with concordance of “sea”

[100]:
```
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 '
```

Counts the instance of "a" in text1 using the NLTK count method and the python std count method found in python lists

[101]:
```
'''
The count method in the NLTK returns the occurences of a given string from any␣
 ↪NLTK text object.
This is the same as the python API method provided in the std library. We can␣
 ↪see this by converting the Text Object into a normal python list and then␣
 ↪using the count method.
The result ends up being the same with both methods
'''
print(text1.count("a"))
print((list(text1)).count("a"))
```

```
4569
4569
```

tokenizes the raw_text into words, and then prints the first 10 words

[102]:
```
#The raw text is from the Misty Mountain song in Lord of The Rings
raw_text = "Far over the Misty Mountains cold. \
To dungeons deep and caverns old. \
We must away, ere break of day. \
To find our long forgotten gold. \
The pines were roaring on the height. \
The winds were moaning in the night. \
The fire was red, it flaming spread. \
The trees like torches blazed with light. \
The wind was on the withered heath. \
But in the forest stirred no leaf. \
There shadows lay be night or day. \
And dark things silent crept beneath. \
"
tokens = word_tokenize(raw_text)
print(tokens[0:10])
```

```
['Far', 'over', 'the', 'Misty', 'Mountains', 'cold', '.', 'To', 'dungeons',
'deep']
```

tokenizes the raw_text into sentences, then displays all the sentences

[103]:
```
sent_tokens = sent_tokenize(raw_text)
sent_tokens
```

[103]:
```
['Far over the Misty Mountains cold.',
 'To dungeons deep and caverns old.',
 'We must away, ere break of day.',
 'To find our long forgotten gold.',
 'The pines were roaring on the height.',
 'The winds were moaning in the night.',
 'The fire was red, it flaming spread.',
 'The trees like torches blazed with light.',
 'The wind was on the withered heath.',
 'But in the forest stirred no leaf.',
 'There shadows lay be night or day.',
 'And dark things silent crept beneath.']
```

creates a stem object using PorterStemmer() then stems each of the tokens using lis comprehension. The stemmed tokens are then printed

[104]:
```
stemmer = PorterStemmer()
stemmed = [stemmer.stem(t) for t in tokens]
print(stemmed)
```

```
['far', 'over', 'the', 'misti', 'mountain', 'cold', '.', 'to', 'dungeon',
'deep', 'and', 'cavern', 'old', '.', 'we', 'must', 'away', ',', 'ere', 'break',
'of', 'day', '.', 'to', 'find', 'our', 'long', 'forgotten', 'gold', '.', 'the',
'pine', 'were', 'roar', 'on', 'the', 'height', '.', 'the', 'wind', 'were',
'moan', 'in', 'the', 'night', '.', 'the', 'fire', 'wa', 'red', ',', 'it',
'flame', 'spread', '.', 'the', 'tree', 'like', 'torch', 'blaze', 'with',
'light', '.', 'the', 'wind', 'wa', 'on', 'the', 'wither', 'heath', '.', 'but',
'in', 'the', 'forest', 'stir', 'no', 'leaf', '.', 'there', 'shadow', 'lay',
'be', 'night', 'or', 'day', '.', 'and', 'dark', 'thing', 'silent', 'crept',
'beneath', '.']
```

1. misti-Misty
2. mountain-Mountains
3. flame-flaming
4. wither-withered
5. stir-stirred

creates a word lemmatize object using WordNetLemmatier(), then we lemmatize each of the tokens from the text using list comprehension. Then the lemmatized tokens are printed

[105]:
```
wnl = WordNetLemmatizer()
lemmas = [wnl.lemmatize(t) for t in tokens]
print(lemmas)
```

```
['Far', 'over', 'the', 'Misty', 'Mountains', 'cold', '.', 'To', 'dungeon',
'deep', 'and', 'cavern', 'old', '.', 'We', 'must', 'away', ',', 'ere', 'break',
```

```
'of', 'day', '.', 'To', 'find', 'our', 'long', 'forgotten', 'gold', '.', 'The',
'pine', 'were', 'roaring', 'on', 'the', 'height', '.', 'The', 'wind', 'were',
'moaning', 'in', 'the', 'night', '.', 'The', 'fire', 'wa', 'red', ',', 'it',
'flaming', 'spread', '.', 'The', 'tree', 'like', 'torch', 'blazed', 'with',
'light', '.', 'The', 'wind', 'wa', 'on', 'the', 'withered', 'heath', '.', 'But',
'in', 'the', 'forest', 'stirred', 'no', 'leaf', '.', 'There', 'shadow', 'lay',
'be', 'night', 'or', 'day', '.', 'And', 'dark', 'thing', 'silent', 'crept',
'beneath', '.']
```

[106]:
```
'''
A. The NLTK library has very strong functionalty in order to learn and␣
 ↪experiment with the basics of NLP. There is access to a vast amount of␣
 ↪example texts that already have
   many built in convenience functions. There are also a vast amount of built␣
 ↪in implementation of common NLP algorithms such as lemmatizing and stemming.

B. The code quality of the NLTK library is also very nice. The API is easily␣
 ↪readable and understandable just by looking at the code itself. Each code␣
 ↪has comments that make it
   clear what the function output and input is, as well as how it works. The␣
 ↪API is also design in a way where it builds on top of each other and␣
 ↪practices strong OOP concepts.
   The written documentation itself is also very easy to read and understand.␣
 ↪It also provides numerous code examples that make it easy to read and␣
 ↪immediately start coding

C. I may use NLTK for future projects in anything that relates to NLP or␣
 ↪parsing and understanding text-based data.
'''
```

[106]:
```
' \nA. The NLTK library has very strong functionalty in order to learn and
experiment with the basics of NLP. There is access to a vast amount of example
texts that already have\n   many built in convenience functions. There are also
a vast amount of built in implementation of common NLP algorithms such as
lemmatizing and stemming.\n\nB. The code quality of the NLTK library is also
very nice. The API is easily readable and understandable just by looking at the
code itself. Each code has comments that make it\n   clear what the function
output and input is, as well as how it works. The API is also design in a way
where it builds on top of each other and practices strong OOP concepts.\n   The
written documentation itself is also very easy to read and understand. It also
provides numerous code examples that make it easy to read and immediately start
coding\n   \nC. I may use NLTK for future projects in anything that relates to
NLP or parsing and understanding text-based data.\n'
```