Assign3_Exploring_NLTK

September 8, 2022

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

- Extract the first 20 tokens from text1

The tokens() method returns the document that this concordance index was created froml retun type is list(str).

Text objects provide methods performing a variety of analyses on the text's contexts and displaying the results.

[16]:
```
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
```

[18]:
```
text1.tokens[:20]
```

[18]:
```
['[',
 'Moby',
 'Dick',
 'by',
 'Herman',
 'Melville',
```

```
 '1851',
 ']',
 'ETYMOLOGY',
 '.',
 '(',
 'Supplied',
 'by',
 'a',
 'Late',
 'Consumptive',
 'Usher',
 'to',
 'a',
 'Grammar']
```

- Concordance for text1 word 'sea', selecting only 5 lines

[29]:
```
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 '
```

- Experiment with the nltk count method and Python's count method

NLTK's count method returns self.tokens.count(word), so when we call it, we can simply do 'text.count(word)'.

While when we use Python's count method, we need to use a list object to call the method, such as 'text.tokens.count(word)'.

[26]:
```
# nltk count()
text1.count('the')
```

[26]: 13721

[27]:
```
#python count()
text1.tokens.count('the')
```

[27]: 13721

- Use raw text of at least 5 sentences and save the text into a variable called raw_text

```
raw_text=
'What doesn't kill you makes you stronger.
Stand a little taller.
Doesn't mean I'm lonely when I'm alone.
```

```
What doesn't kill you makes a fighter.
Footsteps even lighter.
Doesn't mean I'm over 'cause you're gone'
```

source: https://www.azlyrics.com/lyrics/kellyclarkson/whatdoesntkillyoustronger.html

- Use NLTK's word tokenizer, tokenize the text into variable 'tokens'
- Print the first 10 tokens

[32]:
```
raw_text= 'What doesn\'t kill you makes you stronger. Stand a little taller.␣
↪Doesn\'t mean I\'m lonely when I\'m alone. What doesn\'t kill you makes a␣
↪fighter. Footsteps even lighter. Doesn\'t mean I\'m over \'cause you\'re␣
↪gone'

from nltk import word_tokenize
print(word_tokenize(raw_text)[:10])
```

```
['What', 'does', "n't", 'kill', 'you', 'makes', 'you', 'stronger', '.', 'Stand']
```

- Perform sentence segmentation using NLTK's sentence tokenizer and display the sentences

[33]:
```
from nltk import sent_tokenize
print(sent_tokenize(raw_text))
```

```
["What doesn't kill you makes you stronger.", 'Stand a little taller.', "Doesn't
mean I'm lonely when I'm alone.", "What doesn't kill you makes a fighter.",
'Footsteps even lighter.', "Doesn't mean I'm over 'cause you're gone"]
```

- Using NLTK's PorterStemmer()
- Write a list comprehension to stem the text. Display the list.

[35]:
```
from nltk.stem.porter import *
stemmer= PorterStemmer()
stemmed= [stemmer.stem(t) for t in word_tokenize(raw_text)]
print(stemmed)
```

```
['what', 'doe', "n't", 'kill', 'you', 'make', 'you', 'stronger', '.', 'stand',
'a', 'littl', 'taller', '.', 'doe', "n't", 'mean', 'i', "'m", 'lone', 'when',
'i', "'m", 'alon', '.', 'what', 'doe', "n't", 'kill', 'you', 'make', 'a',
'fighter', '.', 'footstep', 'even', 'lighter', '.', 'doe', "n't", 'mean', 'i',
"'m", 'over', "'caus", 'you', "'re", 'gone']
```

- Use NLTK's WordNetLemmatizer
- Write a list comprehension to lemmatize the text. Display the list.
- Some differences I see in the stems verses the lemmas:
- little - little
- doe - does
- lone - lonely
- alon - alone
- footstep - footsteps
- caus - cause

[36]:
```
from nltk.stem import WordNetLemmatizer
wnl= WordNetLemmatizer()
lemmatized= [wnl.lemmatize(t) for t in word_tokenize(raw_text)]
print(lemmatized)
```

```
['What', 'doe', "n't", 'kill', 'you', 'make', 'you', 'stronger', '.', 'Stand',
'a', 'little', 'taller', '.', 'Does', "n't", 'mean', 'I', "'m", 'lonely',
'when', 'I', "'m", 'alone', '.', 'What', 'doe', "n't", 'kill', 'you', 'make',
'a', 'fighter', '.', 'Footsteps', 'even', 'lighter', '.', 'Does', "n't", 'mean',
'I', "'m", 'over', "'cause", 'you', "'re", 'gone']
```

The NLTK library is quite capable and offers several text-manipulating functionalities, including stemming, tokenization, filtering stop words, and POS tagging, to name a few. With NLTK, text processing is rather simple. The library's code quality is top notch. Every method has a documentation string, therefore it is well documented. Also, there is relatively little redundant code, which makes it pretty tidy. I'll certainly make use of this powerful library for future projects to carry out text normalization, filter text documents, and use POS tagging to further analyze incoming content.