

Reg Gonzalez | CS 4395.001

Assignment 2: Exploring NLTK

▼ **Steps 1 & 2: Download necessary libraries for this assignment**

First, import NLTK library and download specific libraries for this assignment.

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

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

Download these specific libraries in order to use the texts that we need for this assignment.

```
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
from nltk.book import *

[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data] Unzipping corpora/gutenberg.zip.
[nltk_data] Downloading package genesis to /root/nltk_data...
[nltk_data] Unzipping corpora/genesis.zip.
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data] Unzipping corpora/inaugural.zip.
[nltk_data] Downloading package nps_chat to /root/nltk_data...
[nltk_data] Unzipping corpora/nps_chat.zip.
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data] Unzipping corpora/webtext.zip.
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data] Unzipping corpora/treebank.zip.
*** 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
```

▼ **Step 3: Using the token() method**

The first thing I learned about Text objects/the tokens() method was that the tokens() method returns a list of strings from the text. Each element in the list represents a token. The second thing I learned was the tokens don't just include individual words, but different kinds of punctuation as well. This means that the the tokens include commas, periods, parenthesis, etc.—essentially everything that isn't whitespace.

For this first code block, we are getting the tokens from text1, making a list of the first 20 tokens, and then displaying the list.

```
# Get tokens from text 1
tokens_from_text1 = text1.tokens
print(tokens_from_text1)

# Make a list of the first 20 tokens from text 1
text1_first_20_tokens = []
for counter in range(0, 21):
    text1_first_20_tokens.append(tokens_from_text1[counter])

# Display the first 20 tokens
print(text1_first_20_tokens)
```

```
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.', '(', 's',
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.', '(', 's'
```

▼ **Step 4: Using the concordance() method**

In this code block, we are using the concordance() method in order to find the concordance lines given a particular word (in this case, "sea"). Because we are specifying the parameter "lines = 5", this means that we're choosing to display only 5 lines. The default value for "lines" is 25.

```
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 '
```

▼ Step 5: Comparing & Contrasting count() methods in NLTK Library and Python

Here, we are using the count() method commonly used with Text objects and Python's own count() method. The two count() methods work virtually the same way.

count() for the Text object returns the number of times a particular word appears in the text. In this case, we are counting the number of times "sea" and "Sea" appear in the text. As you can see, the count() method is case-sensitive.

Python's count() method works in a very similar way. It also counts the number of times a particular element appears in a list; it is also case sensitive.

One thing to note about these different count() methods, though, is that while they are very similar, the count() method for Text objects works *specifically* with/for the Text objects, whereas Python's count() method is more general. It doesn't just work with Text objects.

```
# Experiment with count() methods

# Use NLTK's count() method
print("Number of times the word 'sea' is used in text 1 using NLTK's count() method: ", text1
print("Number of times the word 'Sea' is used in text 1 using NLTK's count() method: ", text1

# Create an arbitrary list
all_stars_7 = ["Raja", "Jinkx", "Monet", "Trinity", "Yvie", "Vivienne", "Jaida", "Shea"]
jinkx_upper = all_stars_7.count("Jinkx")
jinkx_lower = all_stars_7.count("jinkx")
print("\nNumber of times the name 'Jinkx' is in list of all stars: ", jinkx_upper)
print("Number of times the name 'jinkx' is in list of all stars: ", jinkx_lower)

Number of times the word 'sea' is used in text 1 using NLTK's count() method: 433
Number of times the word 'Sea' is used in text 1 using NLTK's count() method: 22

Number of times the name 'Jinkx' is in list of all stars: 1
Number of times the name 'jinkx' is in list of all stars: 0
```

▼ Step 6: Word Tokenization

The raw text I'm using for this portion of the assignment is from Alice Oseman's book, [Nick and Charlie](#). After I get the first 5 sentences of the book into a variable called "raw_text", I then use

NLTK's word tokenizer to get the tokens of the text. I use a list to obtain the first 10 tokens and then display that out onto the screen.

```
# Use raw text from Alice Oseman's book, 'Nick and Charlie'
raw_text = "As Head Boy of Truham Grammar School, I've done many things.\n
            I got drunk on the parents' wine at parents' evening.\n
            I've been photographed with the mayor three times.\n
            I once accidentally made a Year 7 cry.\n
            But none of that was quite as bad as having to stop everyone in Year 13 from enjo

from nltk.tokenize import word_tokenize

# Tokenize the raw text
tokens = word_tokenize(raw_text)

# Print the first 10 tokens
raw_text_first_10_tokens = []
for counter in range(0, 10):
    raw_text_first_10_tokens.append(tokens[counter])

print(raw_text_first_10_tokens)
```

['As', 'Head', 'Boy', 'of', 'Truham', 'Grammar', 'School', ',', 'I', '']

▼ Step 7: Sentence Tokenization

For this code chunk, we're simply using the sentence tokenizer in order to print out the sentences of the text on screen. First, I put the sentences into a variable called "sentences_from_raw_text" and then I print them out using a for loop for each sentence.

```
from nltk.tokenize import sent_tokenize

# Use sent_tokenize on the raw text to print out the sentences
sentences_from_raw_text = sent_tokenize(raw_text)

for sentence in sentences_from_raw_text:
    print(sentence)
```

As Head Boy of Truham Grammar School, I've done many things.
 I got drunk on the parents' wine at parents' evening.
 I've been photographed with the mayor three times.
 I once accidentally made a Year 7 cry.
 But none of that was quite as bad as having to stop everyone in Year 13 from enjoying th

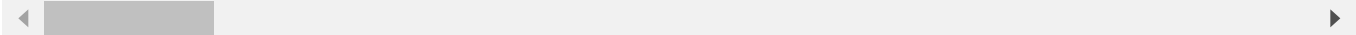
▼ Step 8: Stemming

Here, I'm using NLTK's `PorterStemmer()` method in order to print out the list of stemmed text from the variable "tokens" (the list of tokens from the variable "raw_text"). Stemmed words are the result of removing the affix from the word, often resulting in the base form of the word.

```
from nltk import PorterStemmer

# Use PorterStemmer() and a list comprehension to print out the stemmed text
stemmer = PorterStemmer()
stemmed = [stemmer.stem(token) for token in tokens]
print(stemmed)
```

['as', 'head', 'boy', 'of', 'truham', 'grammar', 'school', ',', 'i', '', 've', 'done',



▼ Step 9: Lemmatization / Comparing & Contrasting Lemmas and Stems

In this code block, I'm using NLTK's `WordNetLemmatizer()` method in order to print out a list of lemmatized text from the variable "tokens" (the list of tokens from variable "raw_text"). Lemmatized words are words cut down to the base form (given the context).

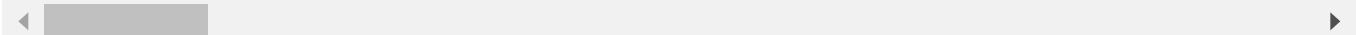
5 differences between lemmas and stems:

1. Lemmas consider the context of a sentence, stems do not.
2. Lemmas convert the word to its base form, stems remove affixes specifically.
3. Lemmatizing applies less aggressive changes than stemming.
4. Stemming can sometimes change the meaning of a word, which can be problematic. For example, "his" can be changed to "hi" if you stem it. Lemmatizing doesn't have that issue.
5. Stemming makes all the words lowercase while lemmatizing keeps the capitalized words.

```
from nltk import WordNetLemmatizer

# Use WordNetLemmatizer() and a list comprehension to print out the lemmatized text
wnl = WordNetLemmatizer()
lemmatized = [wnl.lemmatize(token) for token in tokens]
print(lemmatized)
```

['As', 'Head', 'Boy', 'of', 'Truham', 'Grammar', 'School', ',', 'I', '', 've', 'done',



Step 10: *Opinions on the NLTK Library*

I think that there's a lot of great uses and functionality in the NLTK library. Through this assignment, we've only explored a little bit of the library, but I think so far I can see there's a lot of different uses with NLTK. There's functionality like tokenization, lemmatization, stemming, etc. which are all fundamental know-hows for natural language processing.

In terms of the code quality, I think the NLTK library seems pretty easy to use. The syntax is not complicated and you can perform different kinds of processing (like the ones I mentioned above) with only a few lines of code. The documentation was easy to follow was well, which is great as someone who has had no prior experience to NLTK libraries prior to this class.

Lastly, I think there are a variety of things I can use NLTK libraries for in future projects.

Tokenization, for example, seems like a fundamental process in NLP, and NLTK libraries make it easy to use word *and* sentence tokenizers. Other NLTK properties like the `concordance()` and `count()` methods also seem helpful when analyzing different kinds of text.

[Colab paid products](#) - [Cancel contracts here](#)

✓ 1s completed at 3:12 PM

