Assignment 2: Exploring NLTK

Downloading NLTK and its book collection

In [ ]:
```
import nltk
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

Out[ ]:
```
True
```

In [ ]:
```
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
```

Extracting first 20 tokens from text1

The code below extracts and displays the first 20 tokens from text1, an NLTK Text object, using the tokens() method.

The tokens() method tokenizes strings by extracting words and ignoring whitespace. Text objects can be used to list each time a word occurs.

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

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

Displaying 5 word occurrences from text1

The code below lists 5 occurrences of the word "sea" with its surrounding context from text1, an NLTK Text object, using the concordance() method.

```
In [ ]: 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 '
```

Python vs API: count() method

The code below shows that the Python count() method for a string of text returns the count of the number of times a given object occurs in the given text, while the count() method in the API (https://www.nltk.org/_modules/nltk/text.html) uses tokens to return the count of the number of times a specific word appears in a given text. Unlike the count() method in the API, the Python count() method can be used to return a count of the number of times a particular value occurs between positions in the string of text.

```
In [ ]: print("Testing Python count() method:")
        txt = "Count this, this, this, and this. Count this, too."
        x = txt.count("this", 10, 30)
        print(x)
```

```
Testing Python count() method:
2
```

```
In [ ]: print("Testing API count() method:")
```

```
text1.count("sea")
```

```
Testing API count() method:
```

Out[ ]:
```
433
```

Tokenizing raw text into words

The code below uses raw text of 6 sentences from https://fitnessgram.net/pacertest/, saves the text into the variable 'raw_text', uses NLTK's word tokenizer to tokenize the text into the variable 'tokens', and prints out the first 10 tokens of the text.

In [ ]:
```
from nltk import word_tokenize
raw_text = "The FitnessGram PACER Test is a multistage aerobic capacity test that prog
tokens = word_tokenize(raw_text)
print(tokens[:10])
```

```
['The', 'FitnessGram', 'PACER', 'Test', 'is', 'a', 'multistage', 'aerobic', 'capacit
y', 'test']
```

Tokenizing raw text into sentences

The code below uses the raw text from above and NLTK's sentence tokenizer sent_tokenize() to perform sentence segmentation and prints out the sentences.

In [ ]:
```
from nltk import sent_tokenize
sentences = sent_tokenize(raw_text)
print(sentences)
```

```
['The FitnessGram PACER Test is a multistage aerobic capacity test that progressively
gets more difficult as it continues.', "The test is used to measure a student's aerob
ic capacity as part of the FitnessGram assessment.", 'Students run back and forth as
many times as they can, each lap signaled by a beep sound.', 'The test get progressiv
ely faster as it continues until the student reaches their max lap score.', 'The icon
ic voice of the original test has been replaced by both male and female voices in Eng
lish and Spanish to motivate and encourage participants throughout each of the 22-min
ute tracks.', 'The combination of diverse voices and culturally relevant beats is des
igned to inspire a new generation of students to get active and stay healthy now and
well into adulthood.']
```

Stemming raw text

The code below uses NLTK's PorterStemmer() to create a list comprehension to stem the raw text and displays the resulting list.

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

```
['the', 'fitnessgram', 'pacer', 'test', 'is', 'a', 'multistag', 'aerob', 'capac', 'te
st', 'that', 'progress', 'get', 'more', 'difficult', 'as', 'it', 'continu', '.', 'th
e', 'test', 'is', 'use', 'to', 'measur', 'a', 'student', "'s", 'aerob', 'capac', 'a
s', 'part', 'of', 'the', 'fitnessgram', 'assess', '.', 'student', 'run', 'back', 'an
d', 'forth', 'as', 'mani', 'time', 'as', 'they', 'can', ',', 'each', 'lap', 'signal',
'by', 'a', 'beep', 'sound', '.', 'the', 'test', 'get', 'progress', 'faster', 'as', 'i
t', 'continu', 'until', 'the', 'student', 'reach', 'their', 'max', 'lap', 'score',
'.', 'the', 'icon', 'voic', 'of', 'the', 'origin', 'test', 'ha', 'been', 'replac', 'b
y', 'both', 'male', 'and', 'femal', 'voic', 'in', 'english', 'and', 'spanish', 'to',
'motiv', 'and', 'encourag', 'particip', 'throughout', 'each', 'of', 'the', '22-minu
t', 'track', '.', 'the', 'combin', 'of', 'divers', 'voic', 'and', 'cultur', 'relev',
'beat', 'is', 'design', 'to', 'inspir', 'a', 'new', 'gener', 'of', 'student', 'to',
'get', 'activ', 'and', 'stay', 'healthi', 'now', 'and', 'well', 'into', 'adulthood',
'.']
```

Lemmatizing raw text

The code below uses NLTK's WordNetLemmatizer to create a list comprehension to lemmatize the raw text and displays the resulting list.

The output from stemming and lemmatizing the raw text shows that stems remove affixes like prefixes/suffixes from words, while lemmas convert text to the lexical form by searching for root words. While stems converted all text to lower case, lemmas maintained the capitalization of the raw text. Stems can result in tokens that do not form exact words from the text, so they may have different meanings when compared to the raw text. Stems help process text using the NLTK library's PorterStemmer() method, and lemmas use the library's WordNetLemmatizer() method. Stems and lemmas involve text normalization as they change related words into a standardized form for counting their number of occurrences.

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

```
['The', 'FitnessGram', 'PACER', 'Test', 'is', 'a', 'multistage', 'aerobic', 'capacit
y', 'test', 'that', 'progressively', 'get', 'more', 'difficult', 'a', 'it', 'continue
s', '.', 'The', 'test', 'is', 'used', 'to', 'measure', 'a', 'student', "'s", 'aerobi
c', 'capacity', 'a', 'part', 'of', 'the', 'FitnessGram', 'assessment', '.', 'Student
s', 'run', 'back', 'and', 'forth', 'a', 'many', 'time', 'a', 'they', 'can', ',', 'eac
h', 'lap', 'signaled', 'by', 'a', 'beep', 'sound', '.', 'The', 'test', 'get', 'progre
ssively', 'faster', 'a', 'it', 'continues', 'until', 'the', 'student', 'reach', 'thei
r', 'max', 'lap', 'score', '.', 'The', 'iconic', 'voice', 'of', 'the', 'original', 't
est', 'ha', 'been', 'replaced', 'by', 'both', 'male', 'and', 'female', 'voice', 'in',
'English', 'and', 'Spanish', 'to', 'motivate', 'and', 'encourage', 'participant', 'th
roughout', 'each', 'of', 'the', '22-minute', 'track', '.', 'The', 'combination', 'o
f', 'diverse', 'voice', 'and', 'culturally', 'relevant', 'beat', 'is', 'designed', 't
o', 'inspire', 'a', 'new', 'generation', 'of', 'student', 'to', 'get', 'active', 'an
d', 'stay', 'healthy', 'now', 'and', 'well', 'into', 'adulthood', '.']
```

Takeaways

The functionality and code quality of the NLTK library is useful, especially for text processing, as NLTK is an open-source library for Python and can be used for NLP functions such as tokenizing and text normalization such as stemming and lemmatizing. By experimenting with the different functions of the NLTK library, I can use NLTK in future projects such as translating written text to voice text, filtering spam emails, or spellcheckers for text processing.