Exploring NLTK

Assignment 2
CS 4395.001: Human Language Technologies
Hannah Valena - HCV180000

Imports

The imports and downloads below install Python's Natural Language Toolkit (NLTK), which we can use to perform natural language processing.

```
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
```

```
nltk.download('book')
```

```
[nltk_data]    |   Unzipping corpora/ppattach.zip.
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/senseval.zip.
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
```

```
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection book
True
```

```
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
```

Tokens()

1. The return type for the tokens() method for Text objects is a list.
2. Tokens() returns the tokens in a Text object.

The code below extracts and prints the first 20 tokens from text1 (Moby Dick).

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

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

Concordance()

The code below uses the concordance() method of NLTK Text objects to find 5 occurrences of the word "sea"

```
text1.concordance(word="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 '
```

Count()

The **count()** method in the Text API counts the number of a times a word appears in a Text. It gets all of the tokens in the Text using tokens(), then it counts how many times the word, ie token, appears in the tokens list.

This is different from Python's **count** method. Python's count() method can check for the number of occurrences of a substring of multiple words, while Text's count() method checks for the occurrence of one token.

This difference is shown in the code blocks below.

```
story_tokens = ['hi', 'this', 'is', 'my', 'story', '.', 'my', 'story', 'is', 'a', 'sto
story_text = Text(story_tokens)
story_str = 'hi this is my story. my story is a story.'
```

```
story_text.count('story')
```

```
    3
```

```
story_str.count('story')
```

```
    3
```

```
story_text.count('my story')
```

```
    0
```

```
story_str.count('my story')
```

```
    2
```

Word_tokenize()

NLTK's word tokenizer separates text into individual tokens.

The code below shows how word_tokenize() works using raw text input from the trasncript of Shrek, taken from the following website: https://shrek.fandom.com/wiki/Shrek_(film)/Transcript.

```
from nltk import word_tokenize
```

```
raw_text = 'A masked man is pouring a glass of milk. Another man is shown walking dowr
```

```
tokens = nltk.word_tokenize(raw_text)
```

```
# display the first 10 elements in tokens
tokens[:10]
```

```
['A', 'masked', 'man', 'is', 'pouring', 'a', 'glass', 'of', 'milk', '.']
```

Sent_tokenize()

Similar to word_tokenize, sent_tokenize() separates text into sentences.

The code below shows how sent_tokenize() works using the same raw_text input as above.

```
from nltk import sent_tokenize


sentences = sent_tokenize(raw_text)
for sent in sentences:
  print(sent)
```

```
A masked man is pouring a glass of milk.
Another man is shown walking down the hallway towards a set of doors.
As he is let into the room by two guards, we can see that the man is abnormally
The masked man is dunking what looks to be a small person into the glass of milk
The Gingerbread Man is pulled out of the milk by Thelonious and is slammed down
Farquaad manically laughs as he walks over to the table.
When he reaches the table we see that he is too short to see above it.
He clears his throat and the table is lowered.
```

PorterStemmer()

NTLK's PorterStemmer removes affixes from tokens to help normalize text.

The below code demonstrates PorterStemmer() using the same raw_input text as above.

```
from nltk.stem.porter import *
stemmer = PorterStemmer()

# stem each token using a list comprehension
stemmed = [stemmer.stem(t) for t in tokens]
print(stemmed)
```

```
oni', 'and', 'is', 'slam', 'down', 'onto', 'a', 'cooki', 'sheet', '.', 'farquaad'
```

WordNetLemmatizer()

Similar to PorterStemmer(), NTLK's WordNetLemmatizer() aims to normalize text. WordNetLemmatizer() tries to group different variations of a word to make analysis more straightforward. Lemmatization is less aggressive than stemming.

The code below demonstrates WordNetLemmatizer() using the same raw_input text as above.

Five differences between the stemmed vs. lemmatized (in the format stemmed-lemmatized) list are:

1. mask-masked
2. pour-pouring
3. anoth-Another
4. walk-walking
5. abnorm-abnormally

```
from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()

# stem each token using a list comprehension
lemmatize = [wnl.lemmatize(t) for t in tokens]
print(lemmatize)
```

```
:', 'of', 'the', 'milk', 'by', 'Thelonious', 'and', 'is', 'slammed', 'down', 'ontc
```

NLTK Library

Python's NLTK library seems to have many text processing functionalities, which makes Python a good choice of language for natural language processing. The code quality of Python's NLTK seems to be very high, which accounts for its popularity in both research and in education. In future projects, NLTK could be used to create a chatbot with predictive chat suggestions, a resume screening software, and even a sentiment analysis of various online forums/news.

✓ 0s completed at 10:15 PM