Jesse Truong JTT190006 9/11/2022

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

```
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Package conll2000 is already up-to-date!
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Package conll2002 is already up-to-date!
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package dependency_treebank is already up-to-date!
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package ieer to /root/nltk_data...
[nltk_data]    |   Package ieer is already up-to-date!
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package nps_chat to /root/nltk_data...
[nltk_data]    |   Package nps_chat is already up-to-date!
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package ppattach to /root/nltk_data...
[nltk_data]    |   Package ppattach is already up-to-date!
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    |   Package reuters is already up-to-date!
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Package senseval is already up-to-date!
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Package state_union is already up-to-date!
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
```

✓ 0s completed at 3:09 PM

```
[nltk_data]    |   Package swadesh is already up-to-date!
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Package timit is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Package toolbox is already up-to-date!
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Package udhr is already up-to-date!
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package unicode_samples is already up-to-date!
[nltk_data]    | Downloading package webtext to /root/nltk_data...
```

```
print(text1)
print(text1.tokens[0:20])
```

```
<Text: Moby Dick by Herman Melville 1851>
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.', '(',
```

Text Token is held within a list, so we can access the data using list functions

The Text is a object which tokens are stored within the object

```
text1.concordance('sea',79,5)
```

```
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 '
```

```
print(text1.count('sea'))
print(text1.tokens.count('sea'))
```

```
433
433
```

The count method in the API is exaclty the same as the count in python, it takes a word and counts the number of occuerances in the list. The count in the API uses the python count call in the method

```
from nltk import word_tokenize
```

```
raw_text = "Despite never having been on particularly close terms with Liyue's citizens in
tokens = word_tokenize(raw_text)
print(tokens[0:10])
```

```
['Despite', 'never', 'having', 'been', 'on', 'particularly', 'close', 'terms', 'with'
```

Cite “Ganyu/Lore,” Genshin Impact Wiki. [Online]. Available: https://genshin-impact.fandom.com/wiki/Ganyu/Lore#Character_Story_1. [Accessed: 11-Sep-2022].

```
from nltk import sent_tokenize
sentances = sent_tokenize(raw_text)
for s in sentances:
  print(s)
```

```
Despite never having been on particularly close terms with Liyue's citizens in the fi
Her more important reason for hiding the truth is to prevent curious onlookers from t
After all, a qilin's horns are a sensitive part of their body, both physically and me
Another secret that Ganyu is anxious to keep hidden is the fact that she is watching
The qilin are strict vegetarians, but Liyue is a veritable powerhouse of gastronomica
Ganyu, who has grown accustomed to city life, is thus very vigilant in matters concer
Whenever she finds herself being drawn towards some delicious dish, she will attempt
For Ganyu, the difficulty of such a challenge is second only to finding a Flaming Flo
But she is not one to give up half-way through the ascent.
During the Archon War, she once choked a giant monster to death with ease
```

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
alist = list()
for w in tokens:
  alist.append(stemmer.stem(w))
print(alist)
```

```
['despit', 'never', 'have', 'been', 'on', 'particularli', 'close', 'term', 'with', 'l
```

```
from nltk.stem import WordNetLemmatizer
lemmat = WordNetLemmatizer()
blist = list()
for w in tokens:
  blist.append(lemmat.lemmatize(w))
print(blist)
```

```
['Despite', 'never', 'having', 'been', 'on', 'particularly', 'close', 'term', 'with',
```

Stem-lemma despit-Despite have-having particularli-particularly liyu-Liyue

a: The NLTK library seems pretty powerful in terms of data processing/ text processing, I'm more

of a fan of the sent_tokenize and word_tokenize as its a simple way to split the sentaces and more accurate b: The code quality seem really simply, almost like its very replicaible. Though it save time, i seems like i could make it myself c: the proterstemmer seems to lessen down the data from the rawdata so that may be useful in future project, word_tokenize is amazing so i will def use that so i don't have to make for loop all the time