

(1 & 2) Import NLTK and relevant libraries

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

from nltk.book import *
```

(3) Extract first 20 tokens from text1

- Text1 is a *Text* object which contains various useful methods for text analysis.
- The tokens object contains a list of tokens for each book. The tokens appear to be words and individual symbols

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

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



(4) Utilize the concordance method of the Text object to print the concordance for 'sea'

```
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 '
```

(5) The count method in the nltk Text object is functionally equivalent to the default python list count implementation. Looking into the actual implementation, the NLTK method simply calls the default list count method and passes the list of tokens.

```
print(len(text1))
print(len(list(text1)))

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

260819
433
433

(6) Tokenize a piece of text by words using the NLTK word_tokenize method

Source: [Bee Movie Script](#)

```
raw_text = """According to all known laws of aviation, there is no way a bee should be able to fly. Its wings are too small to get its fat little body off the ground. The bee, of course, flies anyway because bees don't care what humans think is impossible. Yellow, black. Yellow, black. Yellow, black. Yellow, black. Ooh, black and yellow! Let's shake it up a little."""
```

```
tokens = nltk.word_tokenize(raw_text)
print(tokens[:10])
```

```
['According', 'to', 'all', 'known', 'laws', 'of', 'aviation', ',', 'there', 'is']
```

(7) Tokenize a piece of text by sentences using the NLTK sent_tokenize method

```
sentences = nltk.sent_tokenize(raw_text)
print(sentences)
```

```
['According to all known laws of aviation, there is no way a bee should be able to fly.
```

```
']
```

(8) Stem a piece of text using the NLTK PorterStemmer() object

```
stemmer = nltk.stem.PorterStemmer()
print([stemmer.stem(t) for t in tokens])
```

```
['it', 'wing', 'are', 'too', 'small', 'to', 'get', 'it', 'fat', 'littl', 'bodi', 'off',
```

```
']
```

(9) Lemmatize a piece of text using the NLTK WordNetLemmatizer() object. The stemmer and lemmatizer are similar, yet have some minor differences in the results Some of the differences include:

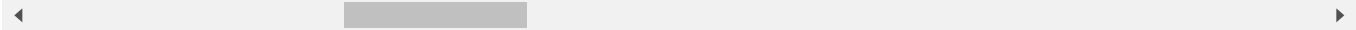
- accord vs According
- aviat vs aviation
- abl vs able
- fli vs fly
- it vs Its

- littl vs little
- bodi vs body

There are many more examples of where they differ.

```
lemmer=.nltk.stem.WordNetLemmatizer()
print([lemmer.lemmatize(t) for t in tokens])

'Its', 'wing', 'are', 'too', 'small', 'to', 'get', 'it', 'fat', 'little', 'body', 'off',
```



(10) The NLTK library appears to provide some useful functionality for dealing with text analysis. The word and sentence tokenizers are convenient for splitting a given piece of text into the desired format. The stemmer and lemmatizer are useful for converting a set of tokens into a potentially more useful format while additionally decreasing the overall size of the data.

Overall, the NLTK library seems to be well made. I do, however, wish that the documentation was more fleshed out in terms of object attributes/data types/implementation/information. It appears that the only provide minor commentary with the object source code which makes it somewhat tedious to learn to use. Though, I realize that I only looked at a very small portion of the library and am nowhere near utilizing it to its full potential.

In the future I would like to look into text/speech sentiment analysis. I've seen other projects on sentiment analysis and it seems really interesting. NLTK would also be useful for data mining and deriving the most useful information from websites.