Arjun Balasubramanian
Axb200075

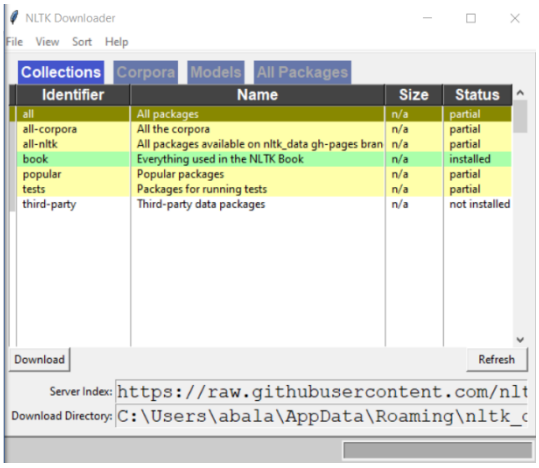
Quiz 2: Exploring NLTK

Instructions: Answer the questions below, save the file, and upload it for the quiz. The quiz has one question which is just a file-upload. The quiz is open notes/computer.

1. Open up a terminal and enter the Python interactive shell. Import NLTK. Then import everything from nltk.book as shown in Chapter 1 of the NLTK book (https://www.nltk.org/book/). Copy and paste your code and output from your interactive shell into the space below.

The code I entered the shell was:
>>import nltk
>>nltk.download()

Then the nltk downloader page showed up and I chose to download the book:



After downloading and exiting the downloader, the following output was produced:
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
True

2. Each of the 9 texts is an NLTK Text object. Look at the code for the Text object at this link: https://www.nltk.org/_modules/nltk/text.html. Look at the tokens() method. At the terminal, extract the first 20 tokens from text1. Copy and paste your code and output into the space below.

To import the text1, which is Moby Dick by Herman Melville in 1851, we imported the text as follows:

>>from nltk.book import text1

Arjun Balasubramanian
Axb200075

After importing the following output is given:

*** 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

Although there is no tokens function in the text class, there is a tokens variable part of the class which stores the text tokenized. To get the first 20 tokens I inputted:

>>> text1.tokens[:20]

I was given the following output after running the line:

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

These are the first 20 tokens in text1

3. Looking at the code for the tokens() method in the API, list two things you learned about the tokens() method or Text objects.

One interesting thing I learned about text objects that I found interesting was that the text in the text objects is stored as tokens rather than as a string, which must be easier for the user as they won't need to go through the process of tokenizing the text themselves. Another interesting thing I learned is that using the text class is much faster than splitting the string as the NLP models can handle it better.

4. Look at the concordance() method in the API. Using the documentation to guide you, in terminal print a concordance for text1 word 'sea', selecting only 5 lines. Copy and paste your code and output in the space below.

Arjun Balasubramanian
Axb200075

To get the concordance for the word sea, I used the following line of code:
>>> text1.concordance('sea',lines = 5)

And got the resulting output:

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. Look at the count() method in the API. How does this work, and how is it different or the same as Python's count method?

The count method in the text class is the same as the python count method. The tokens variable is a list that holds all the tokens from the text in it. The count method in text calls the inbuilt python list function count to get the occurrences and returns it back to the user.

6. Using the raw text below (From Harry Potter and the Half-Blood Prince), save the text into a variable called raw_text. Using NLTK's word tokenizer, tokenize the text into variable 'tokens'. Print the first 10 tokens. Copy and paste your code and the output into the space below.

'Voldemort himself created his worst enemy, just as tyrants everywhere do! Have you any idea how much tyrants fear the people they oppress? All of them realize that, one day, amongst their many victims, there is sure to be one who rises against them and strikes back!'

>>> raw_text = 'Voldemort himself created his worst enemy, just as tyrants everywhere do! Have you any idea how much tyrants fear the people they oppress? All of them realize that, one day, amongst their many victims, there is sure to be one who rises against them and strikes back!' ## In this line we assign the text to the variable raw_text
>>> from nltk.tokenize import word_tokenize ## here we import the tokenizer
>>> tokens = word_tokenize(raw_text) ## Here we tokenize the raw_text and put it in the tokens list
>>> print(tokens[:10]) ## Here we print out the first 10 tokens

Arjun Balasubramanian
Axb200075

After inputting the code, we get the following output:

['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'as', 'tyrants']

7. Using the same raw text, and NLTK's sentence tokenizer sent_tokenize(), perform sentence segmentation and display the sentences. Copy and paste your code and output in the space below.

```
>>> from nltk.tokenize import sent_tokenize ## import the sentence tokenizer
>>> sent_tokens = sent_tokenize(raw_text) ## create the list using the sent tokenizer
>>> print(sent_tokens) ## printed out the list
```

After inputting the code, we get the following output:

['Voldemort himself created his worst enemy, just as tyrants everywhere do!', 'Have you any idea how much tyrants fear the people they oppress?', 'All of them realize that, one day, amongst their many victims, there is sure to be one who rises against them and strikes back!']

8. Using NLTK's PorterStemmer(), write a list comprehension to stem the text and display the list. Copy and paste your code and output in the space below.

```
>>> from nltk.stem import PorterStemmer ## import the porter stemmer
>>> ps = PorterStemmer() ## create the porter stemmer object
>>> stemmed = [ps.stem(word) for word in tokens] ## using list comprehension to stem the tokens list we created earlier
>>> print(stemmed) ## print the stemmed words
```

After inputting the code, we get the following output:

['voldemort', 'himself', 'creat', 'hi', 'worst', 'enemi', ',', 'just', 'as', 'tyrant', 'everywher', 'do', '!', 'have', 'you', 'ani', 'idea', 'how', 'much', 'tyrant', 'fear', 'the', 'peopl', 'they', 'oppress', '?', 'all', 'of', 'them', 'realiz', 'that', ',', 'one', 'day', ',', 'amongst', 'their', 'mani', 'victim', ',', 'there', 'is', 'sure', 'to', 'be', 'one', 'who', 'rise', 'against', 'them', 'and', 'strike', 'back', '!']

9. Using NLTK's WordNetLemmatizer, write a list comprehension to lemmatize the text and display the list. Copy and paste your code and output in the space below.

```
>>> from nltk.stem import WordNetLemmatizer ## import the lemmatizer
>>> lemma = WordNetLemmatizer() ## create the lemmatizer object
```

Arjun Balasubramanian
Axb200075

>>> lemmad = [lemma.lemmatize(word) for word in tokens] ## use list comprehension to lemmatize the words
>>> print(lemmad) ## print the lemmatized words

After running the code, we get the following output:
['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'a', 'tyrant', 'everywhere', 'do', '!', 'Have', 'you', 'any', 'idea', 'how', 'much', 'tyrant', 'fear', 'the', 'people', 'they', 'oppress', '?', 'All', 'of', 'them', 'realize', 'that', ',', 'one', 'day', ',', 'amongst', 'their', 'many', 'victim', ',', 'there', 'is', 'sure', 'to', 'be', 'one', 'who', 'rise', 'against', 'them', 'and', 'strike', 'back', '!']

10. List at least 5 differences you see in the stems verses the lemmas. You can just write them each on a line, like this:

    stem-lemma
    create-created
    hi-his
    everywher-everywhere
    ani-any
    realiz-realize