Exploring_NLTK

September 11, 2022

1 Exploring NLTK

Bridgette Bryant CS 4395.001, Human Language Technologies, Assignment 2

Imports the NLTK library and then imports nltk.book and prints some introductory examples.

[1]:
```python
import nltk
```

[2]:
```python
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
```

Sets the contents of nltk.book to a Text object variable named text1

This code section utilized the .tokens() method from the NLTK Text object. I learned that tokens is a required list in the Text Class given when creating a Text object. All text objects have a tokenized list already which you can get using the text.tokens command. I also learned that because it is already created, you can used the text.tokens command as a list utilizing indexing in orer to avoid doing uneccessary space with a variable (as shown below)

[3]:
```python
for i in range(0,20): print(text1.tokens[i])
```

```
[
Moby
Dick
by
Herman
```

```
Melville
1851
]
ETYMOLOGY
.
(
Supplied
by
a
Late
Consumptive
Usher
to
a
Grammar
```

The command below utilized the concordance() method from the NLTK Text Object, it prints the concordance for text1 word 'sea', selecting only the first 5 lines, with 79 being the default line width.

```
[4]: 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 '
```

The Text object count() method workds the same as the List object count() method. It will return the number of times an object appears in the list. I know this because the way the count function works in this Text API is by calling text.tokens function (which returns a list of already processed tokens) and using the List object count() function on that list. In other words, text1.count() is the same as text1.tokens.count().

```
[5]: text1.count('sea')
```

```
[5]: 433
```

```
[6]: text1.tokens.count('sea')
```

```
[6]: 433
```

```
[7]: text1.count('whale')
```

```
[7]: 906
```

```
[8]: text1.tokens.count('whale')
```

```
[8]: 906
```

My Choosen Raw Text for the next example: "Techkla Minnau. (Pause) Teckla is one of my aides. Like so many the people we tell ourselves we're here to serve, Teckla lives in a district that rarely has electricity and running water as a result of the war. Her childen can now only bathe every two weeks, and they have no light in which to read or study at night. The Republic has always funded these basic services, but now, there are those who would divert the money to the war with no thought for what the people need to survive. If not for people like Teckla and her children, who are we fighting for? My people, your people, all of our people. This war is meant to save them from suffering, not increase it. I support our brave soldiers, whether they come from the clone factories or from any of the thousands of systems loyal to the Republic, but if we continue to impoverish our people, it is not on the battlefield where Dooku will defeat us, but in our own homes. Therefore, it is our duty and responsibility to preserve the lives of those around us by defeating this bill." - Senator Amidala, Padme (Star Wars: The Clone Wars - Season 3 Episode 11)

The code below simply sets the raw_text variable to the choosen text above.

[9]:
```
raw_text = "Techkla Minnau. (Pause) Teckla is one of my aides. Like so many the␣
  ↪people we tell ourselves we're here to serve, Teckla lives in a district␣
  ↪that rarely has electricity and running water as a result of the war. Her␣
  ↪childen can now only bathe every two weeks, and they have no light in which␣
  ↪to read or study at night. The Republic has always funded these basic␣
  ↪services, but now, there are those who would divert the money to the war␣
  ↪with no thought for what the people need to survive. If not for people like␣
  ↪Teckla and her children, who are we fighting for? My people, your people,␣
  ↪all of our people. This war is meant to save them from suffering, not␣
  ↪increase it. I support our brave soldiers, whether they come from the clone␣
  ↪factories or from any of the thousands of systems loyal to the Republic, but␣
  ↪if we continue to impoverish our people, it is not on the battlefield where␣
  ↪Dooku will defeat us, but in our own homes. Therefore, it is our duty and␣
  ↪responsibility to preserve the lives of those around us by defeating this␣
  ↪bill."
```

Import the word tokenizer from nltk

[10]:
```
from nltk.tokenize import word_tokenize
```

The following code line simply tokenizes the raw_text by words using the NLTK word_tokenize and saves it into a variable named tokens.

[11]:
```
tokens = word_tokenize(raw_text)
```

The following code line prints the first 10 tokens from the code above, aka the first 10 tokenized words of raw_text.

[12]:
```
print(tokens[:10])
```

```
['Techkla', 'Minnau', '.', '(', 'Pause', ')', 'Teckla', 'is', 'one', 'of']
```

Import the sentence tokenizer from nltk

```
[13]: from nltk.tokenize import sent_tokenize
```

The following code line simply tokenizes the raw_text by sentences using the NLTK sent_tokenize and prints the tokenized sentences.

```
[14]: print(sent_tokenize(raw_text))
```

```
['Techkla Minnau.', '(Pause) Teckla is one of my aides.', "Like so many the
people we tell ourselves we're here to serve, Teckla lives in a district that
rarely has electricity and running water as a result of the war.", 'Her childen
can now only bathe every two weeks, and they have no light in which to read or
study at night.', 'The Republic has always funded these basic services, but now,
there are those who would divert the money to the war with no thought for what
the people need to survive.', 'If not for people like Teckla and her children,
who are we fighting for?', 'My people, your people, all of our people.', 'This
war is meant to save them from suffering, not increase it.', 'I support our
brave soldiers, whether they come from the clone factories or from any of the
thousands of systems loyal to the Republic, but if we continue to impoverish our
people, it is not on the battlefield where Dooku will defeat us, but in our own
homes.', 'Therefore, it is our duty and responsibility to preserve the lives of
those around us by defeating this bill.']
```

Import the stemmers from NLTK

```
[15]: from nltk.stem import *
```

Create a Porter Stemmer named stemmer

```
[16]: stemmer = PorterStemmer()
```

Stem all the tokens (the tokenized words from raw_text above) and save them into a variable named stemmed. Then print them.

```
[17]: stemmed = [stemmer.stem(t) for t in tokens]
      print(stemmed)
```

```
['techkla', 'minnau', '.', '(', 'paus', ')', 'teckla', 'is', 'one', 'of', 'my',
'aid', '.', 'like', 'so', 'mani', 'the', 'peopl', 'we', 'tell', 'ourselv', 'we',
"'re", 'here', 'to', 'serv', ',', 'teckla', 'live', 'in', 'a', 'district',
'that', 'rare', 'ha', 'electr', 'and', 'run', 'water', 'as', 'a', 'result',
'of', 'the', 'war', '.', 'her', 'childen', 'can', 'now', 'onli', 'bath',
'everi', 'two', 'week', ',', 'and', 'they', 'have', 'no', 'light', 'in',
'which', 'to', 'read', 'or', 'studi', 'at', 'night', '.', 'the', 'republ', 'ha',
'alway', 'fund', 'these', 'basic', 'servic', ',', 'but', 'now', ',', 'there',
'are', 'those', 'who', 'would', 'divert', 'the', 'money', 'to', 'the', 'war',
'with', 'no', 'thought', 'for', 'what', 'the', 'peopl', 'need', 'to', 'surviv',
'.', 'if', 'not', 'for', 'peopl', 'like', 'teckla', 'and', 'her', 'children',
',', 'who', 'are', 'we', 'fight', 'for', '?', 'my', 'peopl', ',', 'your',
'peopl', ',', 'all', 'of', 'our', 'peopl', '.', 'thi', 'war', 'is', 'meant',
'to', 'save', 'them', 'from', 'suffer', ',', 'not', 'increas', 'it', '.', 'i',
'support', 'our', 'brave', 'soldier', ',', 'whether', 'they', 'come', 'from',
'the', 'clone', 'factori', 'or', 'from', 'ani', 'of', 'the', 'thousand', 'of',
'system', 'loyal', 'to', 'the', 'republ', ',', 'but', 'if', 'we', 'continu',
'to', 'impoverish', 'our', 'peopl', ',', 'it', 'is', 'not', 'on', 'the',
'battlefield', 'where', 'dooku', 'will', 'defeat', 'us', ',', 'but', 'in',
'our', 'own', 'home', '.', 'therefor', ',', 'it', 'is', 'our', 'duti', 'and',
'respons', 'to', 'preserv', 'the', 'live', 'of', 'those', 'around', 'us', 'by',
'defeat', 'thi', 'bill', '.']
```

Create a Word Net Lemmatizer named wnl

[18]:
```python
wnl = WordNetLemmatizer()
```

Lemmatize all the tokens (the tokenized words from raw_text above) and save them into a variable named lemmatized. Then print them.

[19]:
```python
lemmatized = [wnl.lemmatize(t) for t in tokens]
print(lemmatized)
```

```
['Techkla', 'Minnau', '.', '(', 'Pause', ')', 'Teckla', 'is', 'one', 'of', 'my',
'aide', '.', 'Like', 'so', 'many', 'the', 'people', 'we', 'tell', 'ourselves',
'we', "'re", 'here', 'to', 'serve', ',', 'Teckla', 'life', 'in', 'a',
'district', 'that', 'rarely', 'ha', 'electricity', 'and', 'running', 'water',
'a', 'a', 'result', 'of', 'the', 'war', '.', 'Her', 'childen', 'can', 'now',
'only', 'bathe', 'every', 'two', 'week', ',', 'and', 'they', 'have', 'no',
'light', 'in', 'which', 'to', 'read', 'or', 'study', 'at', 'night', '.', 'The',
'Republic', 'ha', 'always', 'funded', 'these', 'basic', 'service', ',', 'but',
'now', ',', 'there', 'are', 'those', 'who', 'would', 'divert', 'the', 'money',
'to', 'the', 'war', 'with', 'no', 'thought', 'for', 'what', 'the', 'people',
'need', 'to', 'survive', '.', 'If', 'not', 'for', 'people', 'like', 'Teckla',
'and', 'her', 'child', ',', 'who', 'are', 'we', 'fighting', 'for', '?', 'My',
'people', ',', 'your', 'people', ',', 'all', 'of', 'our', 'people', '.', 'This',
'war', 'is', 'meant', 'to', 'save', 'them', 'from', 'suffering', ',', 'not',
'increase', 'it', '.', 'I', 'support', 'our', 'brave', 'soldier', ',',
'whether', 'they', 'come', 'from', 'the', 'clone', 'factory', 'or', 'from',
'any', 'of', 'the', 'thousand', 'of', 'system', 'loyal', 'to', 'the',
'Republic', ',', 'but', 'if', 'we', 'continue', 'to', 'impoverish', 'our',
'people', ',', 'it', 'is', 'not', 'on', 'the', 'battlefield', 'where', 'Dooku',
'will', 'defeat', 'u', ',', 'but', 'in', 'our', 'own', 'home', '.', 'Therefore',
',', 'it', 'is', 'our', 'duty', 'and', 'responsibility', 'to', 'preserve',
'the', 'life', 'of', 'those', 'around', 'u', 'by', 'defeating', 'this', 'bill',
'.']
```

1.1 Stems vs. Lemmas

1. Stems keep ‘as’ as a word, but lemmas remove the ‘s’ from as making it ‘a’.
2. Stems cut off the ends of words such as ‘e’, ‘ed’, and ‘es’, but lemmas keep endings of words but removes ‘s’.
3. Stems changes ‘y’ to ‘i’, but lemmas keep ‘y’.

4. Stems remove ‘s’ in pronouns such as ‘his’ to ‘hi’, lemmas keep the ‘s’ in pronouns keeping it ‘his’.
5. Stems make everything lowercase, lemmas keeps the given uppercase letters.

2 My opinion of NLTK

NLTK has many functionalities for processing text data such as tokenizing, stemming, lemmatizing, translating, creating your own grammars, and much more. To tokenize data you can use many different tokenizers such as word_tokenize to tokenize by words, sent_tokenize to tokenize by sentences, or WhitespaceTokenizer to tokenize by whitespace. To stem/lemmatize there are also different flavors to choose from. These prebuilt modules ih NLTK are very robust and programmer-friendly to implement. This makes them high quality as they have precise error messages and most of them are very straight forward to use. For the more complex modules NLTK provides extensive documentation of all thier modules with examples. If none of those help somehow it is also a very popular resouce so there are even more examples/implementation online, making it unlikely that you will get stuck on trying to figure out how to use an NLTK module. I will likely use NLTK in future projects to tokenize, stem, and lemmatize data. I may also utilize classification, chunk, parsing, and tagging to analyze data. Possibly might use the probability and toolbox to analyze and predict text patterns.