Natural_Language_Processing

December 26, 2023

0.0.1 Task 01

```
[ ]:
```

```
[1]: import nltk
     from nltk.book import *

     # It will list up all the names
```

```
*** 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
```

```
[ ]:
```

For length of each corpus

```
[6]: data = [text1, text2, text3, text4, text5, text6, text7, text8, text9]

     for i in data:
         print(len(i))
```

```
260819
141576
44764
152901
45010
16967
100676
```

```
4867
69213
```

```
[ ]:
```

For unique words

```
[7]: data = [text1, text2, text3, text4, text5, text6, text7, text8, text9]

     for i in data:

         print(len(set(i)))
```

```
19317
6833
2789
10025
6066
2166
12408
1108
6807
```

```
[ ]:
```

For lexical richness

```
[8]: import nltk
     from nltk.book import *

     data = [text1, text2, text3, text4, text5, text6, text7, text8, text9]

     for i in data:
         tmp = len(set(i)) / len(i)
         print(tmp)
```

```
0.07406285585022564
0.04826383002768831
0.06230453042623537
0.06556530042314962
0.13477005109975562
0.1276595744680851
0.12324685128531129
0.22765564002465585
0.0983485761345412
```

```
[ ]:
```

0.0.2 Task 2

Term Frequency

[11]:
```
def TF(word):
    tmp = "TF of word",word,"is", text1.count(word) / len(text1) * 100
    return tmp

print(TF("monster"))
print(TF("evil"))
print(TF("devil"))
print(TF("the"))
```

```
('TF of word', 'monster', 'is', 0.018786974875296663)
('TF of word', 'evil', 'is', 0.004217484155678842)
('TF of word', 'devil', 'is', 0.019553790176329l8)
('TF of word', 'the', 'is', 5.260736372733581)
```

[ ]:
```

```

[12]:
```
fdist1 = FreqDist(text1)
fdist1.most_common(3)
```

[12]:
```
[(',', 18713), ('the', 13721), ('.', 6862)]
```

[13]:
```
print(TF(","))
print(TF("the"))
print(TF("."))
```

```
('TF of word', ',', 'is', 7.174707364110744)
('TF of word', 'the', 'is', 5.260736372733581)
('TF of word', '.', 'is', 2.630943297842565)
```

[ ]:
```

```

For logTF

[20]:
```
import math

def logTF(word):
    tmp = "logTF of word",word,"is", math.log(text1.count(word), 10)
    return tmp


print(logTF("monster"))
print(logTF("evil"))
print(logTF("devil"))
print(logTF("the"))
print(logTF(","))
print(logTF("the"))
print(logTF("."))
```

```
('logTF of word', 'monster', 'is', 1.6901960800285134)
('logTF of word', 'evil', 'is', 1.041392685158225)
('logTF of word', 'devil', 'is', 1.7075701760979363)
('logTF of word', 'the', 'is', 4.1373857643339695)
('logTF of word', ',', 'is', 4.2721434175910495)
('logTF of word', 'the', 'is', 4.1373857643339695)
('logTF of word', '.', 'is', 3.8364507137201547)
```

[ ]:

```
```

For IDF

[21]:

```
import math

def logTF(word):
    tmp = "logTF of word",word,"is", math.log(9 / text1.count(word), 10)
    return tmp


print(logTF("monster"))
print(logTF("evil"))
print(logTF("devil"))
print(logTF("the"))
print(logTF(","))
print(logTF("the"))
print(logTF("."))
```

```
('logTF of word', 'monster', 'is', -0.7359535705891886)
('logTF of word', 'evil', 'is', -0.08715017571890013)
('logTF of word', 'devil', 'is', -0.7533276666586114)
('logTF of word', 'the', 'is', -3.1831432548946452)
('logTF of word', ',', 'is', -3.3179009081517252)
('logTF of word', 'the', 'is', -3.1831432548946452)
('logTF of word', '.', 'is', -2.8822082042808295)
```

[ ]:

```
```

[ ]:

```
```

0.0.3 Task 3

Tokenization and POS

[22]:

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/hamza/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[22]:
```
True
```

[23]:
```python
text = "NLTK is a powerful library for natural language processing."
```

[26]:
```python
words = nltk.word_tokenize(text)
sentences = nltk.sent_tokenize(text)

print(words)
print(sentences)
```

```
['NLTK', 'is', 'a', 'powerful', 'library', 'for', 'natural', 'language',
'processing', '.']
['NLTK is a powerful library for natural language processing.']
```

[ ]:
```python

```

[27]:
```python
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/hamza/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[27]:
```
True
```

[29]:
```python
tags = nltk.pos_tag(words)
print(tags)
```

```
[('NLTK', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('powerful', 'JJ'), ('library',
'NN'), ('for', 'IN'), ('natural', 'JJ'), ('language', 'NN'), ('processing',
'NN'), ('.', '.')]
```

[ ]:
```python

```

[30]:
```python
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /home/hamza/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[30]:
```
True
```

[31]:
```python
from nltk.corpus import stopwords

filtered_words = [word for word in words if word.lower() not in stopwords.
 ↪words('english')]
```

```
print(filtered_words)
```

```
['NLTK', 'powerful', 'library', 'natural', 'language', 'processing', '.']
```

[ ]:

```
```

[ ]:

```
```