Natural_Language_Processing

December 26, 2023

0.1 Lab 2

0.1.1 Task 01

[ ]:

10 frequently occuring Bigrams

[6]:
```
import nltk
from nltk.book import *

list(bigrams(text1))[:10]
```

[6]:
```
[('[', 'Moby'),
 ('Moby', 'Dick'),
 ('Dick', 'by'),
 ('by', 'Herman'),
 ('Herman', 'Melville'),
 ('Melville', '1851'),
 ('1851', ']'),
 (']', 'ETYMOLOGY'),
 ('ETYMOLOGY', '.'),
 ('.', '(')]
```

[7]:
```
list(bigrams(text2))[:10]
```

[7]:
```
[('[', 'Sense'),
 ('Sense', 'and'),
 ('and', 'Sensibility'),
 ('Sensibility', 'by'),
 ('by', 'Jane'),
 ('Jane', 'Austen'),
 ('Austen', '1811'),
 ('1811', ']'),
 (']', 'CHAPTER'),
 ('CHAPTER', '1')]
```

[8]:
```
list(bigrams(text3))[:10]
```

[8]:
```
[('In', 'the'),
 ('the', 'beginning'),
 ('beginning', 'God'),
 ('God', 'created'),
 ('created', 'the'),
 ('the', 'heaven'),
 ('heaven', 'and'),
 ('and', 'the'),
 ('the', 'earth'),
 ('earth', '.')]
```

[ ]:

5 frequently occuring Trigrams

[12]:
```
from nltk.collocations import *

from nltk.collocations import TrigramAssocMeasures

TrigramCollocationFinder.from_words(text1).nbest(TrigramAssocMeasures().pmi, 5)
```

[12]:
```
[('AFTER', 'EXCHANGING', 'HAILS'),
 ('Anacharsis', 'Clootz', 'deputation'),
 ('CAULKING', 'ITS', 'SEAMS'),
 ('ELIZABETH', 'OAKES', 'SMITH'),
 ('Et', 'tu', 'Brute')]
```

[13]:
```
TrigramCollocationFinder.from_words(text2).nbest(TrigramAssocMeasures().pmi, 5)
```

[13]:
```
[('Austen', '1811', ']'),
 ('Jane', 'Austen', '1811'),
 ('200', 'L', 'per'),
 ('Drury', 'Lane', 'lobby'),
 ('L', 'per', 'annum')]
```

[14]:
```
TrigramCollocationFinder.from_words(text3).nbest(TrigramAssocMeasures().pmi, 5)
```

[14]:
```
[('olive', 'leaf', 'pluckt'),
 ('sewed', 'fig', 'leaves'),
 ('yield', 'royal', 'dainties'),
 ('Fifteen', 'cubits', 'upward'),
 ('leaf', 'pluckt', 'o')]
```

[ ]:

Number of words with length > 16

[29]:
```
count = 0
set1 = set(text1)
for i in set1:
    if len(i) > 16:
        count+=1
print(count)
```

```
11
```

[30]:
```
count = 0
set1 = set(text2)
for i in set1:
    if len(i) > 16:
        count+=1
print(count)
```

```
3
```

[31]:
```
count = 0
set1 = set(text3)
for i in set1:
    if len(i) > 16:
        count+=1
print(count)
```

```
0
```

[ ]:
```

```

Number of words with frequency > 500

[35]:
```
count = 0
set1 = set(text1)
for i in set1:
    if fdist1[i] > 500:
        count+=1
print(count)
```

```
67
```

[36]:
```
count = 0
set1 = set(text2)
for i in set1:
    if fdist1[i] > 500:
        count+=1
print(count)
```

```
64
```

[37]:
```
count = 0
set1 = set(text3)
for i in set1:
    if fdist1[i] > 500:
        count+=1
print(count)
```

```
61
```

[ ]:

Number of ending in “ed” words

[38]:
```
count = 0
set1 = set(text1)
for i in set1:
    if i.endswith('ed'):
        count+=1
print(count)
```

```
2196
```

[39]:
```
count = 0
set1 = set(text2)
for i in set1:
    if i.endswith('ed'):
        count+=1
print(count)
```

```
902
```

[40]:
```
count = 0
set1 = set(text3)
for i in set1:
    if i.endswith('ed'):
        count+=1
print(count)
```

```
281
```

[ ]:

0.1.2 Task 2

[ ]:

[41]:
```
nltk.corpus.gutenberg.fileids()
```

[41]:
```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

[42]:
```
gutenberg_sc = nltk.corpus.gutenberg.words('shakespeare-caesar.txt')
```

[43]:
```
from nltk.corpus import brown
brown.words()
```

[43]:
```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', …]
```

[46]:
```
psh_raw = nltk.data.load('/home/hamza/nltk_data/corpora/hamza/hamza.txt',␣
 ↪format ='raw')
```

[51]:
```
psh_text = nltk.data.load('/home/hamza/nltk_data/corpora/hamza/hamza.txt',␣
 ↪format ='text')
```

[56]:
```
from nltk.util import ngrams
words = nltk.word_tokenize(psh_text)
psh_bigrams = list(ngrams(words, 2))
psh_trigrams = list(ngrams(words, 3))

print(psh_bigrams)
print(psh_trigrams)
```

```
[('Peshawar', 'is'), ('is', 'a'), ('a', 'vibrant'), ('vibrant', 'city'),
('city', 'in'), ('in', 'Pakistan'), ('Pakistan', '.'), ('.', 'It'), ('It',
'has'), ('has', 'a'), ('a', 'rich'), ('rich', 'cultural'), ('cultural',
'heritage'), ('heritage', 'and'), ('and', 'is'), ('is', 'known'), ('known',
'for'), ('for', 'its'), ('its', 'historical'), ('historical', 'significance'),
('significance', '.'), ('.', 'The'), ('The', 'city'), ('city', 'is'), ('is',
'located'), ('located', 'in'), ('in', 'the'), ('the', 'Khyber'), ('Khyber',
```

```
'Pakhtunkhwa'), ('Pakhtunkhwa', 'province'), ('province', '.'), ('.',
'Peshawar'), ('Peshawar', 'has'), ('has', 'a'), ('a', 'diverse'), ('diverse',
'population'), ('population', 'and'), ('and', 'is'), ('is', 'a'), ('a',
'melting'), ('melting', 'pot'), ('pot', 'of'), ('of', 'various'), ('various',
'ethnicities'), ('ethnicities', '.'), ('.', 'The'), ('The', 'city'), ('city',
'has'), ('has', 'a'), ('a', 'unique'), ('unique', 'blend'), ('blend', 'of'),
('of', 'modernity'), ('modernity', 'and'), ('and', 'tradition'), ('tradition',
'.')]
[('Peshawar', 'is', 'a'), ('is', 'a', 'vibrant'), ('a', 'vibrant', 'city'),
('vibrant', 'city', 'in'), ('city', 'in', 'Pakistan'), ('in', 'Pakistan', '.'),
('Pakistan', '.', 'It'), ('.', 'It', 'has'), ('It', 'has', 'a'), ('has', 'a',
'rich'), ('a', 'rich', 'cultural'), ('rich', 'cultural', 'heritage'),
('cultural', 'heritage', 'and'), ('heritage', 'and', 'is'), ('and', 'is',
'known'), ('is', 'known', 'for'), ('known', 'for', 'its'), ('for', 'its',
'historical'), ('its', 'historical', 'significance'), ('historical',
'significance', '.'), ('significance', '.', 'The'), ('.', 'The', 'city'),
('The', 'city', 'is'), ('city', 'is', 'located'), ('is', 'located', 'in'),
('located', 'in', 'the'), ('in', 'the', 'Khyber'), ('the', 'Khyber',
'Pakhtunkhwa'), ('Khyber', 'Pakhtunkhwa', 'province'), ('Pakhtunkhwa',
'province', '.'), ('province', '.', 'Peshawar'), ('.', 'Peshawar', 'has'),
('Peshawar', 'has', 'a'), ('has', 'a', 'diverse'), ('a', 'diverse',
'population'), ('diverse', 'population', 'and'), ('population', 'and', 'is'),
('and', 'is', 'a'), ('is', 'a', 'melting'), ('a', 'melting', 'pot'), ('melting',
'pot', 'of'), ('pot', 'of', 'various'), ('of', 'various', 'ethnicities'),
('various', 'ethnicities', '.'), ('ethnicities', '.', 'The'), ('.', 'The',
'city'), ('The', 'city', 'has'), ('city', 'has', 'a'), ('has', 'a', 'unique'),
('a', 'unique', 'blend'), ('unique', 'blend', 'of'), ('blend', 'of',
'modernity'), ('of', 'modernity', 'and'), ('modernity', 'and', 'tradition'),
('and', 'tradition', '.')]
```

[57]:
```
from nltk.corpus import PlaintextCorpusReader
import os
corpus_root = os.path.expanduser('/home/hamza/nltk_data/corpora/hamza/')
corpus = PlaintextCorpusReader(corpus_root, '.*', encoding='latin1')
wordlist = corpus.words()
bigramlist = list(ngrams(wordlist,2))
print(bigramlist)
```

```
[('Peshawar', 'is'), ('is', 'a'), ('a', 'vibrant'), ('vibrant', 'city'),
('city', 'in'), ('in', 'Pakistan'), ('Pakistan', '.'), ('.', 'It'), ('It',
'has'), ('has', 'a'), ('a', 'rich'), ('rich', 'cultural'), ('cultural',
'heritage'), ('heritage', 'and'), ('and', 'is'), ('is', 'known'), ('known',
'for'), ('for', 'its'), ('its', 'historical'), ('historical', 'significance'),
('significance', '.'), ('.', 'The'), ('The', 'city'), ('city', 'is'), ('is',
'located'), ('located', 'in'), ('in', 'the'), ('the', 'Khyber'), ('Khyber',
'Pakhtunkhwa'), ('Pakhtunkhwa', 'province'), ('province', '.'), ('.',
'Peshawar'), ('Peshawar', 'has'), ('has', 'a'), ('a', 'diverse'), ('diverse',
'population'), ('population', 'and'), ('and', 'is'), ('is', 'a'), ('a',
```

```
'melting'), ('melting', 'pot'), ('pot', 'of'), ('of', 'various'), ('various',
'ethnicities'), ('ethnicities', '.'), ('.', 'The'), ('The', 'city'), ('city',
'has'), ('has', 'a'), ('a', 'unique'), ('unique', 'blend'), ('blend', 'of'),
('of', 'modernity'), ('modernity', 'and'), ('and', 'tradition'), ('tradition',
'.')]
```

[ ]:

[ ]:

0.1.3 Task 3

[ ]:

[58]:
```
cfd = nltk.ConditionalFreqDist(bigramlist)
```

[59]:
```
def generate_model(cfdist, word, num):
    for i in range(num):
        print(word, end=' ')
        word = cfdist[word].max()
```

[62]:
```
generate_model(cfd, 'various', 30)
```

```
various ethnicities . The city in Pakistan . The city in Pakistan . The city in
Pakistan . The city in Pakistan . The city in Pakistan . The city
```

[ ]: