Natural Language Processing
(IST-664 2022-1005)

Homework Assignment: #1
Student Name: Ryan Tervo

PART 1: CHOOSE THE DATA: For my text selection I choose 'melville-moby_dick.txt' and 'carroll-alice.txt'.

- I would like to see if the texts can be distinquished, story identified, and general theme identified using the natural language processing techniques we have learned so far.

In [1]:
```
#   PART 1 CODE:  CHOOSING THE DATA:

#   Import the Natural Language Toolkit Library and FreqDist
import nltk
from nltk import FreqDist

#   Select File Names out of nltk corpus
fileName1 = nltk.corpus.gutenberg.fileids()[7]
fileName2 = nltk.corpus.gutenberg.fileids()[12]

#   Gather text for those two files.
fileText1 = nltk.corpus.gutenberg.raw(fileName1)
fileText2 = nltk.corpus.gutenberg.raw(fileName2)

#   Determine the number of tokens in each text.
fileTextLength1 = len(nltk.word_tokenize(fileText1))
fileTextLength2 = len(nltk.word_tokenize(fileText2))

#   Print File Names to ensure the write text was selected.
print('For for the first text I selected was "' + fileName1 + '" and has ' +
str(fileTextLength1) + ' tokens prior to any processing.')
print('For for the second text I selected was "' + fileName2 + '" and has ' +
str(fileTextLength2) + ' tokens prior to any processing.')
```

```
For for the first text I selected was "carroll-alice.txt" and has 33494 tokens prior to any
processing.
For for the second text I selected was "melville-moby_dick.txt" and has 255028 tokens prior
to any processing.
```

PART 2: Examine the Text in the Documents Decide how to process the words, tokenization, whether to use all lower case, stopwords, or lemmatization.

- Use processes developed in the lab.

Complete the Following:

- Part A: List Top 50 words by frequency, normalized by the length of the document.
- Part B: List top 50 bigrams by frequency
- Part C: List top 50 bigrams by their Mutual Information Scores (using min frequency 5)

Note: You may wish to modify the stop word list, based on your question in Task 3.

In [2]:
```
#   PART 2A:  TOP 50 FREQUENT WORDS WITHOUT INITIAL FILTERING:
#   Basic Order
#   - Tokenize
#   - lower case all words
#   - Apply alpha filter
#   - Remove stop words
#   - Examine
#   - Remove additional stop words
#   - Repeat last 4 steps until satisfied.

#   Tokenize the text:
fileText1_token = nltk.word_tokenize(fileText1)
fileText2_token = nltk.word_tokenize(fileText2)

#   Lower Case Words:
fileText1_token_lower = [w.lower() for w in fileText1_token]
fileText2_token_lower = [w.lower() for w in fileText2_token]


ndist1_0 = FreqDist(fileText1_token_lower)
ndist2_0 = FreqDist(fileText2_token_lower)

#   PRINT OUT:
print('Initial Top 50 Tokens in FileText1')
print('Word\t\tPercent')
print('-'*50)
ndist1_0.most_common(50)
for item in ndist1_0.most_common(50):
    if len(item[0]) > 6:
        numTab = 1
    else:
        numTab = 2
    print (item[0], '\t'*numTab, str(round(item[1]/fileTextLength1 * 100, 2)) + '%')

print('\nInitial Top 50 Tokens in FileText2')
print('Word\t\tPercent')
print('-'*50)
for item in ndist2_0.most_common(50):
    if len(item[0]) > 6:
        numTab = 1
    else:
        numTab = 2
    print (item[0], '\t'*numTab, str(round(item[1]/fileTextLength2 * 100, 2)) + '%')
```

```
Initial Top 50 Tokens in FileText1
Word            Percent
--------------------------------------------------
,                7.22%
the              4.82%
'                3.91%
.                2.91%
and              2.42%
to               2.15%
a                1.88%
she              1.62%
it               1.61%
i                1.59%
```

```
of                1.49%
said              1.38%
!                 1.34%
alice             1.18%
was               1.09%
in                1.07%
you               1.06%
that              0.85%
--                0.79%
as                0.76%
her               0.74%
:                 0.7%
at                0.62%
n't               0.61%
?                 0.6%
's                0.58%
;                 0.58%
on                0.57%
had               0.55%
with              0.53%
all               0.53%
be                0.44%
for               0.44%
so                0.43%
very              0.41%
not               0.4%
they              0.4%
but               0.39%
this              0.39%
little            0.38%
do                0.37%
he                0.35%
is                0.34%
out               0.34%
what              0.31%
down              0.3%
one               0.3%
up                0.29%
his               0.28%
about             0.28%

Initial Top 50 Tokens in FileText2
Word             Percent
--------------------------------------------------
,                 7.53%
the               5.66%
.                 2.86%
of                2.58%
and               2.52%
a                 1.84%
to                1.8%
;                 1.64%
in                1.63%
that              1.21%
his               0.99%
it                0.98%
i                 0.83%
```

```
he                0.74%
but               0.71%
!                 0.69%
is                0.69%
as                0.68%
with              0.68%
--                0.67%
was               0.65%
's                0.64%
for               0.63%
''                0.63%
all               0.59%
``                0.57%
this              0.55%
at                0.52%
not               0.48%
by                0.47%
from              0.43%
whale             0.43%
him               0.42%
on                0.42%
so                0.42%
be                0.41%
?                 0.39%
one               0.36%
you               0.35%
there             0.34%
now               0.31%
had               0.31%
have              0.3%
or                0.28%
were              0.27%
they              0.26%
which             0.25%
me                0.25%
then              0.25%
their             0.24%
```

In [3]:

```
#   PART 2A:  TOP 50 FREQUENT WORDS WITH INITIAL FILTERING:
#   Remove Symbols from the Text
import re

def alpha_filter(w):
# pattern to match word of non-alphabetical characters
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False

fileText1_token_lower_noSym = [w for w in fileText1_token_lower if not
alpha_filter(w)]
fileText2_token_lower_noSym = [w for w in fileText2_token_lower if not
alpha_filter(w)]

#   Remove Stop Words0:
nltkstopwords = nltk.corpus.stopwords.words('english')
morestopwords =
['could','would','might','must','need','sha','wo','y',"'s","'d","'ll","'t","'m","'re","'
 "n't"]
stopWords0 = nltkstopwords + morestopwords

fileText1_token_lower_noSym_noStop1 = [w for w in fileText1_token_lower_noSym if not w
in stopWords0]
fileText2_token_lower_noSym_noStop1 = [w for w in fileText2_token_lower_noSym if not w
in stopWords0]

#   Examine0
#ndist1_0 = FreqDist(fileText1_token_lower_noSym_noStop1)  #  Commented out after I
examined.
#ndist2_0 = FreqDist(fileText2_token_lower_noSym_noStop1)  #  Commented out after I
examined.

#   PRINT OUT:
print('Intermediate Top 50 Tokens in FileText1')
print('Word\t\tPercent')
print('-'*50)
ndist1_0.most_common(50)
for item in ndist1_0.most_common(50):
    if len(item[0]) > 6:
        numTab = 1
    else:
        numTab = 2
    print (item[0], '\t'*numTab, str(round(item[1]/fileTextLength1 * 100, 2)) + '%')

print('\nIntermediate Top 50 Tokens in FileText2')
print('Word\t\tPercent')
print('-'*50)
for item in ndist2_0.most_common(50):
    if len(item[0]) > 6:
        numTab = 1
    else:
        numTab = 2
    print (item[0], '\t'*numTab, str(round(item[1]/fileTextLength2 * 100, 2)) + '%')
```

```
Intermediate Top 50 Tokens in FileText1
Word             Percent
--------------------------------------------------
,                 7.22%
the               4.82%
'                 3.91%
.                 2.91%
and               2.42%
to                2.15%
a                 1.88%
she               1.62%
it                1.61%
i                 1.59%
of                1.49%
said              1.38%
!                 1.34%
alice             1.18%
was               1.09%
in                1.07%
you               1.06%
```

```
that             0.85%
--               0.79%
as               0.76%
her              0.74%
:                0.7%
at               0.62%
n't              0.61%
?                0.6%
's               0.58%
;                0.58%
on               0.57%
had              0.55%
with             0.53%
all              0.53%
be               0.44%
for              0.44%
so               0.43%
very             0.41%
not              0.4%
they             0.4%
but              0.39%
this             0.39%
little           0.38%
do               0.37%
he               0.35%
is               0.34%
out              0.34%
what             0.31%
down             0.3%
one              0.3%
up               0.29%
his              0.28%
about            0.28%

Intermediate Top 50 Tokens in FileText2
Word            Percent
--------------------------------------------------
,                7.53%
the              5.66%
.                2.86%
of               2.58%
and              2.52%
a                1.84%
to               1.8%
;                1.64%
in               1.63%
that             1.21%
his              0.99%
it               0.98%
i                0.83%
he               0.74%
but              0.71%
!                0.69%
is               0.69%
as               0.68%
with             0.68%
--               0.67%
```

```
was              0.65%
's               0.64%
for              0.63%
''               0.63%
all              0.59%
``               0.57%
this             0.55%
at               0.52%
not              0.48%
by               0.47%
from             0.43%
whale            0.43%
him              0.42%
on               0.42%
so               0.42%
be               0.41%
?                0.39%
one              0.36%
you              0.35%
there            0.34%
now              0.31%
had              0.31%
have             0.3%
or               0.28%
were             0.27%
they             0.26%
which            0.25%
me               0.25%
then             0.25%
their            0.24%
```

In [4]:
```
#   PART 2A:  TOP 50 FREQUENT WORDS WITH ENHANCED FILTERING:


#   Remove Additional Words
additionalStopWords1 = ["n't", "'s", "went", "'and", "'m", "'it", "'ll", "'you",
"'ve", "'but", "'what", "'re", "'that", "'d", "'oh"]
additionalStopWords2 = ["'s", "n't", "'ll"]
stopWords1 = stopWords0 + additionalStopWords1
stopWords2 = stopWords0 + additionalStopWords2

fileText1_token_lower_noSym_noStop2 = [w for w in fileText1_token_lower_noSym if not w
in stopWords1]
fileText2_token_lower_noSym_noStop2 = [w for w in fileText2_token_lower_noSym if not w
in stopWords2]

#   Examine2
ndist1_1 = FreqDist(fileText1_token_lower_noSym_noStop2)
ndist2_1 = FreqDist(fileText2_token_lower_noSym_noStop2)

print('\nFinal Top 50 Words in FileText1')
print('Word\t\tPercent')
print('-'*50)
for item in ndist1_1.most_common(50):
    if len(item[0]) > 6:
        numTab = 1
    else:
        numTab = 2
    print (item[0], '\t'*numTab, str(round(item[1]/fileTextLength1 * 100, 2)) + '%')


print('\nFinal Top 50 Words in FileText2')
print('Word\t\tPercent')
print('-'*50)
for item in ndist2_1.most_common(50):
    if len(item[0]) > 6:
        numTab = 1
    else:
        numTab = 2
    print (item[0], '\t'*numTab, str(round(item[1]/fileTextLength2 * 100, 2)) + '%')
```

```
Final Top 50 Words in FileText1
Word            Percent
--------------------------------------------------
said             1.38%
alice            1.18%
little           0.38%
one              0.3%
know             0.26%
like             0.25%
queen            0.22%
thought          0.22%
time             0.2%
see              0.2%
king             0.19%
began            0.17%
turtle           0.17%
hatter           0.17%
mock             0.17%
quite            0.16%
gryphon          0.16%
think            0.16%
way              0.16%
much             0.15%
say              0.15%
first            0.15%
head             0.15%
thing            0.15%
go               0.14%
voice            0.14%
rabbit           0.14%
looked           0.13%
never            0.13%
got              0.13%
get              0.13%
mouse            0.13%
duchess          0.13%
round            0.12%
came             0.12%
tone             0.12%
dormouse         0.12%
great            0.12%
well             0.11%
back             0.11%
two              0.11%
```

```
cat              0.11%
march            0.1%
large            0.1%
last             0.1%
long             0.1%
moment           0.09%
put              0.09%
come             0.09%
hare             0.09%

Final Top 50 Words in FileText2
Word            Percent
--------------------------------------------------
whale            0.43%
one              0.36%
like             0.23%
upon             0.22%
ahab             0.2%
man              0.19%
ship             0.18%
old              0.17%
ye               0.17%
sea              0.15%
though           0.15%
yet              0.13%
time             0.13%
captain          0.13%
long             0.12%
still            0.12%
said             0.12%
great            0.12%
two              0.11%
boat             0.11%
seemed           0.11%
head             0.11%
last             0.11%
see              0.11%
thou             0.11%
whales           0.1%
way              0.1%
stubb            0.1%
queequeg         0.1%
little           0.1%
white            0.1%
round            0.1%
say              0.09%
sperm            0.09%
three            0.09%
men              0.09%
may              0.09%
first            0.09%
every            0.09%
us               0.09%
much             0.09%
well             0.09%
never            0.08%
hand             0.08%
```

```
good             0.08%
almost           0.08%
starbuck         0.08%
ever             0.08%
go               0.08%
even             0.07%
```

In [5]:
```
#   PART 2B:  TOP 50 BIGRAMS WITHOUT FILTERING
#########################################################################################

#   Bigrams and Bigram frequency distribution
#########################################################################################

#   Perform Imports:
from nltk.collocations import *
bigram_measures1 = nltk.collocations.BigramAssocMeasures()
bigram_measures2 = nltk.collocations.BigramAssocMeasures()

#   Bigrams and Bigram frequency distribution
fileText1_token_lower_bigrams = list(nltk.bigrams(fileText1_token_lower))
fileText2_token_lower_bigrams = list(nltk.bigrams(fileText2_token_lower))

#   Create the bigram finder and score the bigrams by frequency
fileText1_token_lower_bigrams_finder =
BigramCollocationFinder.from_words(fileText1_token_lower)
fileText1_token_lower_bigrams_scored =
fileText1_token_lower_bigrams_finder.score_ngrams(bigram_measures1.raw_freq)

fileText2_token_lower_bigrams_finder =
BigramCollocationFinder.from_words(fileText2_token_lower)
fileText2_token_lower_bigrams_scored =
fileText2_token_lower_bigrams_finder.score_ngrams(bigram_measures2.raw_freq)

#   Scores are Sorted in Decreasing frequency
print('Initial Top 50 Bigram Frequency Score for FileText1\n' + '-' * 50)
for bscore in fileText1_token_lower_bigrams_scored[:50]:
    print (bscore)

print('\nInitial Top 50 Bigram Frequency Score for FileText2\n' + '-' * 50)
for bscore in fileText2_token_lower_bigrams_scored[:50]:
    print (bscore)
```

```
Initial Top 50 Bigram Frequency Score for FileText1
--------------------------------------------------
((',', 'and'), 0.013733803069206425)
((',', "'"), 0.012808264166716427)
(("'", 'said'), 0.009852510897474175)
(('!', "'"), 0.008449274496924822)
(('.', "'"), 0.00782229653072192)
(('said', 'the'), 0.006180211381142891)
(("'", 'i'), 0.005045679823251926)
(('?', "'"), 0.004687406699707410)
(('of', 'the'), 0.003821579984474831 4)
(('said', 'alice'), 0.003433450767301606)
(("'", 'the'), 0.003224458111900639)
(('in', 'a'), 0.002896041081984833 2)
((',', 'i'), 0.002418343583925479)
(('and', 'the'), 0.00235863139666806)
```

```
 (('alice', ','), 0.0023287753030393505)
 (('in', 'the'), 0.0023287753030393505)
 (('it', 'was'), 0.0021794948348958024)
 (('the', 'queen'), 0.0020600704603809636)
 (('to', 'the'), 0.0020600704603809636)
 ((',', 'but'), 0.0018510778049799965)
 ((',', 'as'), 0.0018212217113512867)
 (('as', 'she'), 0.0018212217113512867)
 (('the', 'king'), 0.0018212217113512867)
 (("'", 'she'), 0.0017913656177225771)
 (('at', 'the'), 0.0017913656177225771)
 (('she', 'had'), 0.0017913656177225771)
 (('a', 'little'), 0.0017615095240938676)
 (('it', ','), 0.001731653430465158)
 (('*', '*'), 0.0017017973368364484)
 (("'", 'alice'), 0.0016719412432077386)
 (('i', "'m"), 0.0016719412432077386)
 (('she', 'was'), 0.0016719412432077386)
 (('mock', 'turtle'), 0.001642085149579029)
 ((',', 'you'), 0.0016122290559503194)
 ((';', 'and'), 0.0016122290559503194)
 (('alice', '.'), 0.0016122290559503194)
 ((',', 'she'), 0.0015823729623216098)
 (('and', 'she'), 0.0015823729623216098)
 (('the', 'mock'), 0.0015823729623216098)
 (('--', "'"), 0.0015525168686929003)
 (('do', "n't"), 0.0015525168686929003)
 (('the', 'gryphon'), 0.0015525168686929003)
 (('the', 'hatter'), 0.0015525168686929003)
 (('to', 'be'), 0.0015226607750641907)
 (("'", "'"), 0.001492804681435480 9)
 (('.', 'the'), 0.0014928046814354809)
 ((',', 'that'), 0.0014629485878067713)
 (('went', 'on'), 0.0014330924941780617)
 (('.', 'alice'), 0.0013733803069206425)
 (('to', 'herself'), 0.001343524213291933)

 Initial Top 50 Bigram Frequency Score for FileText2
 ------------------------------------------------
 ((',', 'and'), 0.010355725645811 44)
 (('of', 'the'), 0.007348212745267186)
 (('in', 'the'), 0.004603416095487554)
 ((',', 'the'), 0.003599604749282432)
 ((';', 'and'), 0.003360415326944492)
 (('to', 'the'), 0.0028506673776997034)
 (('.', '``'), 0.002336998290383801)
 (('.', 'but'), 0.002336998290383801)
 ((',', 'that'), 0.002305629185814891)
 ((',', 'as'), 0.002058597487334724)
 (('.', "''"), 0.0019174365167746287)
 (("''", '``'), 0.0018546983076368085)
 ((',', 'i'), 0.0017958812365701 02)
 ((',', 'he'), 0.0017684332700723057)
 (('from', 'the'), 0.0017253007512900544)
 ((',', 'in'), 0.0015802186426588454)
 (('.', 'the'), 0.0014939536050943426)
 (('of', 'his'), 0.0014586633624543187)
```

```
(('and', 'the'), 0.0014429788101698636)
(('the', 'whale'), 0.0014037674294587262)
(('on', 'the'), 0.0013684771868187023)
((',', 'but'), 0.0013449503583920198)
((';', 'but'), 0.001337108082249792)
(('of', 'a'), 0.0013057389776808882)
(('at', 'the'), 0.001290054425396427)
(('to', 'be'), 0.001290054425396427)
(('!', "''"), 0.0012469219066141757)
((',', "''"), 0.0012469219066141757)
(('by', 'the'), 0.0012469219066141757)
(('with', 'the'), 0.001223395078187493)
(('for', 'the'), 0.0011959471116896967)
(('it', 'was'), 0.0011528145929074454)
(('it', 'is'), 0.0011096820741251941)
((',', 'with'), 0.0010351804507740326)
(('.', 'and'), 0.0010273381746318051)
(('in', 'his'), 0.0010273381746318051)
(('.', 'i'), 0.0010155747604184639)
((',', 'or'), 0.00101165362234735)
(('?', "''"), 0.00101165362234735)
(('in', 'a'), 0.00101165362234735)
(('with', 'a'), 0.0009724422416362125)
(('the', 'ship'), 0.0009645999654939850)
(('into', 'the'), 0.0009606788274228712)
(('it', ','), 0.0009489154132095299)
((',', 'when'), 0.0009371519989961886)
(('.', 'it'), 0.0009371519989961886)
(('him', ','), 0.0009371519989961886)
((',', 'it'), 0.0009253885847828473)
((';', 'the'), 0.0009018617563561648)
((',', 'to'), 0.0008861772040717098)
```

In [6]:
```
#   2B:  TOP 50 BIGRAMS WITH FILTERING

#   Remove Stop Words:
fileText1_token_lower_bigrams_finder.apply_word_filter(lambda w: w in stopWords1)
fileText2_token_lower_bigrams_finder.apply_word_filter(lambda w: w in stopWords2)

#   Apply Frequency Filter:
fileText1_token_lower_bigrams_finder.apply_freq_filter(5)
fileText2_token_lower_bigrams_finder.apply_freq_filter(5)

#   Apply Alpha Filter
fileText1_token_lower_bigrams_finder.apply_word_filter(alpha_filter)
fileText2_token_lower_bigrams_finder.apply_word_filter(alpha_filter)

#   Rescore Text File 1:
fileText1_token_lower_bigrams_scored1 =
fileText1_token_lower_bigrams_finder.score_ngrams(bigram_measures1.raw_freq)
print('\nFinal Top 50 Bigram Frequency Score for FileText1\n' + '-' * 50)

for bscore in fileText1_token_lower_bigrams_scored1[:50]:
    print (bscore)

#   Rescore Text File 2:
fileText2_token_lower_bigrams_scored1 =
fileText2_token_lower_bigrams_finder.score_ngrams(bigram_measures2.raw_freq)
```

```
    print('\nFinal Top 50 Bigram Frequency Score for FileText2\n' + '-' * 50)
    for bscore in fileText2_token_lower_bigrams_scored1[:50]:
        print (bscore)
```

```
Final Top 50 Bigram Frequency Score for FileText1
--------------------------------------------------
(('said', 'alice'), 0.003433450767301606)
(('mock', 'turtle'), 0.001642085149579029)
(('march', 'hare'), 0.0009255389024899983)
(('thought', 'alice'), 0.0007762584343464501)
(('white', 'rabbit'), 0.0006568340598316116)
(('alice', 'thought'), 0.00035827312354451543)
(("'of", 'course'), 0.0003284170299158058)
(('alice', 'said'), 0.0003284170299158058)
(('poor', 'alice'), 0.0003284170299158058)
(('alice', 'replied'), 0.00026870484265838657)
(('alice', 'looked'), 0.00023884874902967696)
(('king', 'said'), 0.00023884874902967696)
(('little', 'thing'), 0.00023884874902967696)
(('poor', 'little'), 0.00023884874902967696)
(('alice', 'began'), 0.00020899265540096732)
(('cried', 'alice'), 0.00020899265540096732)
(('good', 'deal'), 0.00020899265540096732)
(('oh', 'dear'), 0.00020899265540096732)
(('beautiful', 'soup'), 0.00017913656177225771)
(('golden', 'key'), 0.00017913656177225771)
(('great', 'hurry'), 0.00017913656177225771)
(('little', 'door'), 0.00017913656177225771)
(('said', 'nothing'), 0.00017913656177225771)
(('three', 'gardeners'), 0.00017913656177225771)
(('alice', 'felt'), 0.0001492804681435481)
(('another', 'moment'), 0.0001492804681435481)
(('came', 'upon'), 0.0001492804681435481)
(('cheshire', 'cat'), 0.0001492804681435481)
(('feet', 'high'), 0.0001492804681435481)
(('kid', 'gloves'), 0.0001492804681435481)
(('little', 'golden'), 0.0001492804681435481)
(('next', 'witness'), 0.0001492804681435481)
(('offended', 'tone'), 0.0001492804681435481)
(('play', 'croquet'), 0.0001492804681435481)
(('right', 'size'), 0.0001492804681435481)
(('trembling', 'voice'), 0.0001492804681435481)
(('white', 'kid'), 0.0001492804681435481)

Final Top 50 Bigram Frequency Score for FileText2
--------------------------------------------------
(('sperm', 'whale'), 0.0006783568863026805)
(('white', 'whale'), 0.0004156406355380586)
(('moby', 'dick'), 0.00031761218376021456)
(('old', 'man'), 0.000294085355333532)
(('captain', 'ahab'), 0.00023918942233793936)
(('right', 'whale'), 0.00020389917969791552)
(('captain', 'peleg'), 0.00012547641827564032)
(('cried', 'ahab'), 0.00012547641827564032)
(('mr.', 'starbuck'), 0.00011371300406229904)
(('one', 'hand'), 0.00010979186599118528)
(('let', 'us'), 0.00010587072792007152)
```

```
(('every', 'one'), 9.410731370673024e-05)
(('cried', 'stubb'), 9.018617563561648e-05)
(('look', 'ye'), 8.626503756450273e-05)
(('never', 'mind'), 8.626503756450273e-05)
(('one', 'side'), 8.626503756450273e-05)
(("'ye", 'see'), 8.234389949338896e-05)
(('thou', 'art'), 8.234389949338896e-05)
(('new', 'bedford'), 7.058048528004768e-05)
(('said', 'stubb'), 7.058048528004768e-05)
(('sperm', 'whales'), 7.058048528004768e-05)
(('years', 'ago'), 7.058048528004768e-05)
(('cried', 'starbuck'), 6.665934720893393e-05)
(('cape', 'horn'), 6.273820913782016e-05)
(('greenland', 'whale'), 6.273820913782016e-05)
(('lower', 'jaw'), 6.273820913782016e-05)
(('old', 'ahab'), 6.273820913782016e-05)
(('something', 'like'), 6.273820913782016e-05)
(('whale', 'fishery'), 6.273820913782016e-05)
(('young', 'man'), 6.273820913782016e-05)
(('ivory', 'leg'), 5.88170710667064e-05)
(('thus', 'far'), 5.88170710667064e-05)
(('well', 'known'), 5.88170710667064e-05)
(('whaling', 'voyage'), 5.88170710667064e-05)
(('captain', 'bildad'), 5.489593299559264e-05)
(('chief', 'mate'), 5.489593299559264e-05)
(('ere', 'long'), 5.489593299559264e-05)
(('ye', 'see'), 5.489593299559264e-05)
(('dost', 'thou'), 5.097479492447888e-05)
(('ever', 'since'), 5.097479492447888e-05)
(('forty', 'years'), 4.705365685336512e-05)
(('much', 'like'), 4.705365685336512e-05)
(('one', 'hundred'), 4.705365685336512e-05)
(('poor', 'queequeg'), 4.705365685336512e-05)
(('three', 'boats'), 4.705365685336512e-05)
(('three', 'years'), 4.705365685336512e-05)
(('ahab', 'stood'), 4.313251878225136e-05)
(('almost', 'every'), 4.313251878225136e-05)
(('art', 'thou'), 4.313251878225136e-05)
(('every', 'way'), 4.313251878225136e-05)
```

In [7]:

```
#   PART 2C:   MUTUAL INFORMATION WITH FILTERING:
fileText1_token_lower_noSym_noStop2_finder3 =
BigramCollocationFinder.from_words(fileText1_token_lower_noSym_noStop2)
fileText2_token_lower_noSym_noStop2_finder3 =
BigramCollocationFinder.from_words(fileText2_token_lower_noSym_noStop2)

#   Remove Stop Words:
fileText1_token_lower_noSym_noStop2_finder3.apply_word_filter(lambda w: w in
stopWords1)
fileText2_token_lower_noSym_noStop2_finder3.apply_word_filter(lambda w: w in
stopWords2)

#   Apply Frequency Filter:
fileText1_token_lower_noSym_noStop2_finder3.apply_freq_filter(5)
fileText2_token_lower_noSym_noStop2_finder3.apply_freq_filter(5)

#   Apply Alpha Filter
fileText1_token_lower_noSym_noStop2_finder3.apply_word_filter(alpha_filter)
```

```
fileText2_token_lower_noSym_noStop2_finder3.apply_word_filter(alpha_filter)

#   Rescore:
fileText1_token_lower_noSym_noStop2_scored3 =
fileText1_token_lower_noSym_noStop2_finder3.score_ngrams(bigram_measures1.pmi)
fileText2_token_lower_noSym_noStop2_scored3 =
fileText2_token_lower_noSym_noStop2_finder3.score_ngrams(bigram_measures2.pmi)

#   Display Results:
print('\nFinal Top 50 Mutual Information Score for FileText1\n' + '-' * 50)
for bscore in fileText1_token_lower_noSym_noStop2_scored3[:50]:
    print (bscore)

#   Display Results:
print('\nFinal Top 50 Mutual Information Score for FileText2\n' + '-' * 50)
for bscore in fileText2_token_lower_noSym_noStop2_scored3[:50]:
    print (bscore)
```

```
Final Top 50 Mutual Information Score for FileText1
--------------------------------------------------
(('soo', 'oop'), 10.765582102959705)
(('beg', 'pardon'), 10.572937025017309)
(('play', 'croquet'), 10.309902619183516)
(('golden', 'key'), 10.180619602238547)
(('evening', 'beautiful'), 10.113505406380012)
(('kid', 'gloves'), 10.113505406380012)
(('join', 'dance'), 9.872497306876218)
(('set', 'work'), 9.350544603680861)
(('dance', 'join'), 9.28753480615506)
(("'of", 'course'), 8.803549953158726)
(('white', 'kid'), 8.66604642940879)
(('beautiful', 'soup'), 8.528542905658856)
(('three', 'gardeners'), 8.514043335963741)
(('march', 'hare'), 8.48547418376697)
(('please', 'majesty'), 8.403012023574998)
(('good', 'deal'), 8.2103669456326)
(('cheshire', 'cat'), 8.139977617741204)
(("'off", 'head'), 7.736435757300187)
(('trembling', 'voice'), 7.72494011846236)
(('mock', 'turtle'), 7.688960821356792)
(('next', 'witness'), 7.666046429408791)
(('feet', 'high'), 7.646937606461087)
(('white', 'rabbit'), 7.5708891963684515)
(('minute', 'two'), 7.556128737330754)
(('oh', 'dear'), 7.476075485764721)
(('great', 'hurry'), 7.41306568823892)
(('right', 'size'), 7.287534806155062)
(('offended', 'tone'), 7.251008930129947)
(('another', 'moment'), 6.481237190880499)
(('little', 'golden'), 6.0875101978470685)
(('came', 'upon'), 5.872497306876218)
(('mouse', 'mouse'), 5.110230274347151)
(('well', 'say'), 5.012986412304226)
(('poor', 'little'), 4.872497306876218)
(('little', 'door'), 4.251008930129947)
(('said', 'caterpillar'), 4.083758062985961)
(('little', 'thing'), 3.958227180902103)
```

```
(('turning', 'alice'), 3.9435804049376983)
(('king', 'queen'), 3.711850119021916)
(('poor', 'alice'), 3.7025723054339057)
(('said', 'king'), 3.6249726683419468)
(('indeed', 'said'), 3.5285429056588544)
(('said', 'pigeon'), 3.458153577767458)
(('thought', 'alice'), 3.4345667574498417)
(('cried', 'alice'), 3.4290072321079403)
(('said', 'hatter'), 3.373264680180945)
(("'no", 'said'), 3.3426763603475216)
(('said', 'dodo'), 3.3426763603475216)
(('alice', 'replied'), 3.2555244112524413)
(('said', 'cat'), 3.251702700300031)

Final Top 50 Mutual Information Score for FileText2
--------------------------------------------------
(('caw', 'caw'), 13.859874455263565)
(('samuel', 'enderby'), 13.026047321844771)
(('warp', 'woof'), 12.763012916010975)
(('latitude', 'longitude'), 12.744397237843629)
(('straits', 'sunda'), 12.707871361818517)
(('mrs.', 'hussey'), 12.620408520568176)
(('um', 'um'), 12.596363046601526)
(('iii', 'duodecimo'), 12.57036783806858)
(('heidelburgh', 'tun'), 12.48547894048207)
(('ii', 'octavo'), 12.315553939039756)
(('st.', 'george'), 12.096436649736168)
(('father', 'mapple'), 12.007431643677423)
(('hither', 'thither'), 11.90051643976091)
(('huzza', 'porpoise'), 11.781871943262292)
(('fiery', 'pit'), 11.637482033927117)
(('beef', 'bread'), 11.398016099231729)
(('steering', 'oar'), 11.122908861097361)
(('hundred', 'seventy-seventh'), 11.064015172043792)
(('wife', 'child'), 10.80098076621)
(('fore', 'aft'), 10.800980766209998)
(('centuries', 'ago'), 10.760338781712653)
(('cape', 'horn'), 10.67996536524863)
(('seven', 'hundred'), 10.648977672764946)
(('blows', 'blows'), 10.345301282433809)
(('moby', 'dick'), 10.332617355466972)
(('new', 'york'), 10.298480425680815)
(('new', 'zealand'), 10.298480425680815)
(('new', 'bedford'), 10.298480425680813)
(('ha', 'ha'), 9.952983859655049)
(('book', 'ii'), 9.874981347653776)
(('harpoons', 'lances'), 9.815075596095785)
(('gave', 'understand'), 9.694409102011953)
(('book', 'folio'), 9.68233626971138)
(('saturday', 'night'), 9.652588926317325)
(('drew', 'nigh'), 9.407137489200277)
(('chief', 'mate'), 9.391104972046934)
(('eight', 'ten'), 9.248439743181219)
(('years', 'ago'), 9.20007672161982)
(('english', 'whalers'), 9.167162099147083)
(('forty', 'years'), 9.141183032566252)
(('brought', 'alongside'), 9.026984441098824)
```

```
(('lower', 'jaw'), 8.952983859655047)
(('four', 'oceans'), 8.942336615455538)
(('thousand', 'miles'), 8.871370094101396)
(('drawing', 'nigh'), 8.822174988479121)
(('new', 'england'), 8.783907252851057)
(('pagan', 'harpooneers'), 8.763012916010975)
(('closed', 'eyes'), 8.685503548790063)
(('queer', 'queer'), 8.596363046601526)
(('full', 'grown'), 8.565764304515968)
```

Part 2 Questions:

a)Briefly state why you chose the processing options that you did.
I initiall wanted to see what would happen if I applied the techniques without filtering or removing of stop words. The results can be seen in my "Initial Top 50 Words" and "Initial Top 50 Bigrams". As you can see, the results are not helpful and gave very little context to the text.

I then applied the filtering techniques we were taught and reran the Top 50 words and Top 50 Bigrams. This can be seen in "Final Top 50 Words" and "Final Top 50 Bigrams". The results were substantially improved. The words provided by and large gave context and insight into the text.

b)Are there any problems with the word or bigram lists that you found? Could you get abetter list of bigrams?
After applying the filtering techniques (lowercase all of the words, remove symbols and stop words) the Top 50 Frequency Words and Top 50 Bigrams were useful and the results were meaningful. A better bigram list could be accomplished by further stop word customization. Essentially it's a matter of identifying the "noise" in the text so we can filter that out and get the meaning out of it.

In addition, the Bigrams were ordered in the list by their score. After reviewing the list I think a better method would be to group the top 50 elements that are related by topic. For instance, I noticed that in the text 'Moby Dick' many Bigrams were related to the water/sea navigation. In a similar vain if the names and placed could be grouped together that would provide a lot of quick useful information. Idenfying which items should be included should be based on the score but how they are grouped should be based on topic. I think the reader would gain additional insight more quickly.

c)How are the top 50 bigrams by frequency different from the top 50 bigrams scored by Mutual Information?
The top 50 Bigrams by frequency and my mutual score contain several of the same elements but they differ greatly in their scoring. For instance, in Moby Dick the main two characters have a high frequency score but a much lower mutual information score. I think what is happening is when two words occur together it improves both the frequency score and the mutual information score. However, when those two words appear apart then it does not hurt the frequency score but the mutual information score is diminished. Both scores are important but the mutual information score provides a sort of counter balance to how much stock we should put in the two words being together.

I initially suspected that I would gain more insight into the texts from the mutual information score. However, after reviewing the lists the frequency score provides more insight and context then the mutual score. There a number of elements that are on the mutal information top 50 that were not on the frequency top 50 but could easily be grouped together to provide a more complete picture.

d)If you modify the stop word list, or expand the methods of filtering, describe that here.
I did modify the stop word list. I initiall applied the NLTK stop words and then manually inspected the top 200 frequency words. I then created a secondary stop word list that was customized for each of the two corpus analyzed. This greatly improved the top 50 words relevancy. For the Bigrams analysis I initially used all of the words in lower case to see what it would produce. Again the Bigram analysis did not provided much useful information. I then applied the custom stop word list as well as other filtering techniques. Prior to filtering the bigrams gave little insight into the text but after the filtering the bigrams gave a great deal of context.

e)You may choose to also run top trigram lists, and include them in the analysis in part 3.

I did not complete a top trigram list.

Part 3 Questions:

a. Define a comparison question between the two documents.
1) Do the three lists clearly identify two distinct texts?
2) Armed with general knowledge of both stories could a person identify each story without any additional guiance or help?
3) Could the heart of the story be determine?
-- Namely a dangerous adventure in wonderland or a hunt for revenge on the open seas?

b. (20 pts) Answer the question by picking examples from the lists and discussing how they show that the documents are different, not just reporting numbers.
- Discussion may include collection steps if significant, which will count towards discussion of differences.
1) The two lists differ greatly. First, each respective lists identifies the main characters within the stories. In addition, Moby Dick has many reference to the sea. These are clearly to different texts. Text distinction is overwhelmingly met.

2) In both texts we have the main characters and many iconic supporting elements. The stories can be identified by the characters names. For Alice in Wonderland we have Alice, Queen, Mad Hatter, rabbit, little door, cheshir cat, golden key, croquet, and caterpillar. For Moby Dick we have Captain Ahab and many references to a whale. In addition, there are many references to the sea. Examples include latitude, longitude, aft, harpoons, lances, whaler, oceans, whale, captain, starbuck, fishery, and voyage. Story identification is clearly met.

3) This last part fails. The lists do provide enough information to recognize which story they came from but unfortunatley the lists are not sufficient to convey the heart of the story itself. While we recognize this is the sotry Alice in Wonderland we don't have any indication that Alice is going on an adventure in wonderland trying to escape the Queen. We clearly have indications of Moby Dick and the sea in which somehow a whale is involved but we don't know their relationship. We don't know the pain, anger, and thirst for revenge. Unfortunately these techniques failed pass the third test.

Additional Merit
I think if words could be segmented into various categories and then the word frequency and bigram analysis conducted within those categories it would provide a great deal of insight. For instance, one category could be emotions and a second category could be main (recurring) characters. If those words were then scored words it could convey or express emotion would be beneficial in relation to main characters. Since emotional words might not occur often they might not ever score high enough on the word frequency list or bigram scores to be picked up. However, if we looked for combinations of just character and emotional type words we could see if a relationship exists or not.

In some lines of thinking systems are looked at in terms of centers of gravity and how they relate to each other. I think in terms of NLP something similar could be done. Step 1 would be identify the centers of gravity. Step 2 would be to determine how those centers of gravity relate to each other. Finally, step 3 how do the centers of gravity and their associated relationships change over time. In terms of this assignment I believe we identified the centers of gravity but failed to see how they relate to each other and how they changed over time.

In [ ]:

In [ ]: