Natural Language Processing
(IST-664 2022-1005)

Homework Assignment: #2
Student Name: Ryan Tervo

ASSIGNMENT OBJECTIVE:

- Conduct sentiment analysis from the Homework #1 texts.
- The first homework assignment texts were 'melville-moby_dick.txt' and 'carroll-alice.txt'.

PART 1: DESCRIPTIVE STATISTICS

- To gain understanding of the sentiment you will extract those sentences that contain adjective or adverb phrases.
- Once you extract these sentences, you will do a descriptive statistics about them, e.g.,
  -- Average length of a sentence
  -- Top 50 adjective/adverb phrases (by frequency)
  -- Top 50 adjective/adverb words
  -- Top 50 nouns or verbs
- You are encouraged to combine these findings with the other fields of the text (ngrams, frequency of POS tags by group, etc).
  -- You can use tables or figures to show these summaries.

PART 2: INTERPRETATION OF THE RESULTS:

- Provide interpretation of the Part 1 results.
- What do you learn from these results?

PART 3: VISION FOR THE FUTURE

- The last thing to do is to envision how you would conduct a sentiment analysis on the review texts.

SUBMISSION REQUIREMENTS:

- Use Jupyter Notebook to develop a report to explain your processing and analysis process.
- Integrate all of your Python codes, outputs, and detailed writeup explaining your workflow and interpretation in the notebook and export the report as a HTML file for submission.
- As always, please explain your process – both for cleaning and for analysis.
- Each student should submit a HTML report exported from Jupyter Notebook that includes your Python codes, outputs, and detailed writeup explaining your workflow and interpretation of the results.

- How to Submit this Homework:
  -- Go to the 2U system and the Assignment for Homework #2 and attach your report.

Your submission should include:

1. A HTML report with description of the cleaning and analysis processes and results of the analysis and interpretation of the results, including tables/figures.
2. Include your thoughts on how to conduct sentiment analysis in the above report.

In [1]:
```
#    PART 0:  INITIAL SETUP
##################################################################
```

```
#   IMPORT LIBRARIES:
import nltk
from   nltk import FreqDist
import re
import pandas as pd

#   DEFINE FUNCTIONS:
################################################################
def TagFunction(grammar_re, taggedtext):
    grammar_label = grammar_re.split(':')[0]

    #   PARSE EACH SENTENCE:
    chunk_parser = nltk.RegexpParser(grammar_re)

    grammar_tags = []
    for sent in taggedtext:
        if len(sent) > 0:
            tree = chunk_parser.parse(sent)
            for subtree in tree.subtrees():
                if subtree.label() == grammar_label:
                    grammar_tags.append(subtree)

    #   STORE PHRASES:
    grammar_phrases = []
    for sent in grammar_tags:
        temp = ''
        for w, t in sent:
            temp += w+ ' '
        grammar_phrases.append(temp)

    #   FREQUENCY DISTRIBUTIONS
    freq_grammar = nltk.FreqDist(grammar_phrases)
    return grammar_tags, grammar_phrases, freq_grammar


#   DEFINE VARIABLES:
################################################################
#   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 it has ' +
str(fileTextLength1) + '  tokens prior to any processing.')
print('For for the second text I selected was "' + fileName2 + '" and it has ' +
str(fileTextLength2) + ' tokens prior to any processing.')
```

```
For for the first  text I selected was "carroll-alice.txt"      and it has 33494  tokens pri
or to any processing.
```

```
For for the second text I selected was "melville-moby_dick.txt" and it has 255028 tokens pri
or to any processing.
```

In [2]:
```
#######################################################################################
```

PART 1: DESCRIPTIVE STATISTICS

To gain understanding of the sentiment you will extract those sentences that contain adjective or adverb phrases. Once you extract these sentences, you will do a descriptive statistics about them, e.g.,

- Average length of a sentence
- Top 50 adjective/adverb phrases (by frequency)
- Top 50 adjective/adverb words
- Top 50 nouns or verbs

You are encouraged to combine these findings with the other fields of the text (ngrams, frequency of POS tags by group, etc).

- You can use tables or figures to show these summaries.

In [3]:
```
#   PART 1.1:  TOKENIZE:
#   SPLIT TEXT INTO SENTENCES:
textsplit1 = nltk.sent_tokenize(fileText1)
textsplit2 = nltk.sent_tokenize(fileText2)

#   APPLY WORD TOKENIZER TO EACH SENTENCE:
tokentext1 = [nltk.word_tokenize(sent) for sent in textsplit1]
tokentext2 = [nltk.word_tokenize(sent) for sent in textsplit2]

#   TAG TEXT:
taggedtext1 = [nltk.pos_tag(tokens) for tokens in tokentext1]
taggedtext2 = [nltk.pos_tag(tokens) for tokens in tokentext2]

#   DEFINE RE TAGS
grammar_adjph = "ADJPH: {<RB.?>+<JJ.?>}"
grammar_advph = "ADVPH: {<RB>+<RB>}"
#grammar_other = "ahhaha"

#   RUN FREQUENCY DISTRIBUTION FUNCTION:
grammar_adjph_tags1, grammar_adjph_phrases1, grammar_adjph_freqDist1 =
TagFunction(grammar_adjph, taggedtext1)
grammar_advph_tags1, grammar_advph_phrases1, grammar_advph_freqDist1 =
TagFunction(grammar_advph, taggedtext1)

#   RUN FREQUENCY DISTRIBUTION FUNCTION:
grammar_adjph_tags2, grammar_adjph_phrases2, grammar_adjph_freqDist2 =
TagFunction(grammar_adjph, taggedtext2)
grammar_advph_tags2, grammar_advph_phrases2, grammar_advph_freqDist2 =
TagFunction(grammar_advph, taggedtext2)
```

In [4]:
```
#-------------------------------------------------------------------------------------
-------------
```

In [5]:
```
#   PART 1.2a:  IDENTIFY
##################################################################
print('For for the first  text I selected was "' + fileName1)
```

```
#   TOP 50 ADJECTIVE PHRASES
###################################################################
print('\nText1 - Top adjective phrases by frequency: \n' + '-'*75)
for word, freq in grammar_adjph_freqDist1.most_common(50):
    if len(word) <= 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1

    print(word, '\t'*tabMult, freq)
print('Length of adjective phrase sentences: ', len(grammar_adjph_tags1))


#   TOP 50 ADVERB PHRASES
###################################################################
print('\nText1 - Top adverb phrases by frequency:  \n' + '-'*75)
for word, freq in grammar_advph_freqDist1.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)
print('Length of adverb phrase sentences: ', len(grammar_advph_tags1))


#   TOP 50 ADJECTIVE TOKENS:
#####################################################################
adjective_tokens1 = []
for sentence in taggedtext1:
    for word, pos in sentence:
        if pos in ['JJ', 'JJR', 'JJS']: # adjective, comparative, superlative
            if len(word) > 1:
                adjective_tokens1.append(word)
freq_adjective1 = nltk.FreqDist(adjective_tokens1)

print('\nText1 - Top 50 Adjective Tokens\n' + '-'*75 )
for word, freq in freq_adjective1.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)


#   TOP 50 ADVERB TOKENS:
#####################################################################
adverb_tokens1 = []
for sentence in taggedtext1:
    for word, pos in sentence:
        if pos in ['RB', 'RBR', 'RBS']: # adverb, comparative, superlative
            if len(word)>1:
                adverb_tokens1.append(word)
freq_adverb1 = nltk.FreqDist(adverb_tokens1)

print('\nText1 - Top 50 Adverb Tokens\n' + '-'*75 )
for word, freq in freq_adverb1.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)


#   TOP 50 NOUN TOKENS:
####################################################################
noun_tokens1 = []
for sentence in taggedtext1:
    for word, pos in sentence:
        if pos in ['NN', 'NNS', 'NNP', 'NNPS']: # noun, noun plural, proper noun,
proper noun plural
            if len(word) > 1:
                noun_tokens1.append(word)
freq_noun1 = nltk.FreqDist(noun_tokens1)

print('\nText1 - Top 50 Noun Tokens\n' + '-'*75 )
for word, freq in freq_noun1.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)


#   TOP 50 VERB TOKENS:
####################################################################
verb_tokens1 = []
for sentence in taggedtext1:
    for word, pos in sentence:
        if pos in ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']: # verb, verb past tense,
verb gerund, verb past principle, vrb non 3rd person singular, verb 3rd person
singular present   ural
            if len(word) > 1:
                verb_tokens1.append(word)
freq_verb1 = nltk.FreqDist(verb_tokens1)

print('\nText1 - Top 50 Verb Tokens\n' + '-'*75 )
for word, freq in freq_verb1.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)
```

```
#####################################################################
# Now we have two lists of POS tags combinations we can compare
# We need to get the sentences back from the tagging exercise and run some stats

# Create a list of original sentences from the ADJECTIVE phrase subset:
adjph_whole_sentences1 = []

# loop over the sentences in the adjective phrase sentences we created:
for sents in grammar_adjph_tags1:
    temp=''
    for (word, tag) in sents:
        temp += word+' '
        adjph_whole_sentences1.append(temp)

print('\n', '=' * 50, '\n', len(adjph_whole_sentences1))
```

```
For for the first  text I selected was "carroll-alice.txt

Text1 - Top adjective phrases by frequency:
---------------------------------------------------------------------------
so much                  8
very curious             6
very glad                5
very much                4
very little              4
very likely              3
so many                  3
too much                 3
very tired               2
quite natural            2
very deep                2
n't much                 2
very few                 2
very good                2
as much                  2
so grave                 2
always ready             2
very uncomfortable       2
almost wish              2
very difficult           2
quite silent             2
certainly too much       2
not much                 2
so large                 2
n't very civil           2
very interesting         2
once more                2
very sleepy              1
so VERY remarkable       1
so VERY much             1
too dark                 1
VERY good                1
rather glad              1
no longer                1
too large                1
too small                1
```

```
not much larger          1
really impossible        1
almost certain           1
very nice                1
now only ten             1
too slippery             1
very fond                1
very small               1
quite surprised          1
quite dull               1
now more                 1
so desperate             1
very hot                 1
ever so many             1
Length of adjective phrase sentences:  222

Text1 - Top adverb phrases by frequency:
---------------------------------------------------------------------------
as well                  15
very soon                6
very politely            5
down here                4
very much                4
just as well             4
so VERY                  3
very well                3
so far                   3
Just then                3
back again               3
well enough              3
down again               3
as soon                  3
very carefully           3
As soon                  3
so often                 3
not quite                3
very nearly              3
n't quite                3
as long                  3
just now                 3
very slowly              2
very earnestly           2
not even                 2
too far                  2
as hard                  2
rather not               2
very gravely             2
very humbly              2
very angrily             2
so easily                2
certainly too            2
rather timidly           2
'All right               2
n't very                 2
'Exactly so              2
asleep again             2
never even               2
```

```
n't even                 2
never before             1
never once               1
suddenly down            1
not much                 1
not here before          1
too long                 1
VERY deeply              1
now only                 1
not possibly             1
quite plainly            1
Length of adverb phrase sentences:  235

Text1 - Top 50 Adjective Tokens
---------------------------------------------------------------------------
little                   124
other                    40
great                    39
much                     34
large                    33
last                     32
more                     31
first                    31
such                     26
poor                     25
thought                  24
good                     24
long                     23
same                     23
curious                  19
sure                     19
next                     18
old                      17
right                    16
low                      14
high                     14
whole                    13
mad                      13
many                     12
glad                     11
own                      10
small                    10
few                      9
best                     9
different                9
least                    9
afraid                   8
white                    8
ready                    8
dear                     8
beautiful                8
golden                   7
larger                   7
enough                   7
deep                     6
nice                     6
dry                      6
```

```
bright                   6
melancholy               6
offended                 6
full                     6
sharp                    6
hot                      5
likely                   5
nervous                  5

Text1 - Top 50 Adverb Tokens
---------------------------------------------------------------------------
n't                      203
not                      128
very                     126
so                       91
again                    83
then                     72
quite                    48
now                      47
as                       45
just                     44
never                    41
only                     41
here                     39
down                     36
once                     31
well                     31
back                     31
too                      25
rather                   25
soon                     24
up                       24
away                     23
yet                      21
ever                     20
even                     17
much                     17
more                     16
indeed                   15
perhaps                  14
anxiously                14
hastily                  14
first                    13
However                  13
certainly                13
far                      13
suddenly                 12
there                    12
still                    12
about                    12
always                   12
else                     11
hardly                   11
enough                   11
really                   10
nearly                   10
So                       9
```

```
Then                     9
angrily                  9
together                 9
timidly                  9

Text1 - Top 50 Noun Tokens
---------------------------------------------------------------------------
Alice                    390
Queen                    72
time                     65
King                     60
Turtle                   58
Mock                     56
Hatter                   55
Gryphon                  54
way                      53
head                     50
thing                    49
voice                    47
Rabbit                   44
Duchess                  42
tone                     40
Dormouse                 39
March                    34
moment                   31
'It                      31
Hare                     31
nothing                  30
things                   30
door                     30
Mouse                    29
eyes                     28
Caterpillar              27
day                      25
course                   25
Cat                      25
'You                     24
round                    23
White                    22
words                    21
minute                   21
sort                     20
feet                     19
anything                 19
hand                     19
dear                     18
house                    18
use                      17
question                 17
side                     17
table                    17
something                17
jury                     17
court                    17
garden                   16
end                      15
idea                     15
```

```
Text1 - Top 50 Verb Tokens
----------------------------------------------------------------------------
said                     456
was                      361
had                      183
be                       145
do                       113
's                       105
is                       104
know                     87
were                     86
went                     83
have                     80
did                      74
see                      66
began                    58
'm                       54
say                      51
think                    49
thought                  47
go                       46
looked                   45
got                      45
get                      44
are                      42
came                     40
herself                  38
've                      38
been                     38
're                      36
made                     30
found                    30
looking                  30
put                      29
replied                  28
seemed                   27
going                    27
make                     27
heard                    26
come                     25
tell                     25
took                     23
felt                     23
added                    23
getting                  22
like                     22
find                     21
does                     20
tried                    19
being                    19
take                     18
spoke                    17

 ========================================================
 464
```

```
In [6]: #--------------------------------------------------------------------
```

In [7]:

```
#   PART 1.2b:  IDENTIFY
##################################################################
print('For for the second  text I selected was "' + fileName2)

#   TOP 50 ADJECTIVE PHRASES
###################################################################
print('\nText2 - Top adjective phrases by frequency: \n' + '-'*75)
for word, freq in grammar_adjph_freqDist2.most_common(50):
    if len(word) <= 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1

    print(word, '\t'*tabMult, freq)
print('Length of adjective phrase sentences: ', len(grammar_adjph_tags2))


#   TOP 50 ADVERB PHRASES
###################################################################
print('\nText2 - Top adverb phrases by frequency:  \n' + '-'*75)
for word, freq in grammar_advph_freqDist2.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)
print('Length of adverb phrase sentences: ', len(grammar_advph_tags2))


#   TOP 50 ADJECTIVE TOKENS:
#####################################################################
adjective_tokens2 = []
for sentence in taggedtext2:
    for word, pos in sentence:
        if pos in ['JJ', 'JJR', 'JJS']: # adjective, comparative, superlative
            if len(word) > 1:
                adjective_tokens2.append(word)
freq_adjective2 = nltk.FreqDist(adjective_tokens2)

print('\nText2 - Top 50 Adjective Tokens\n' + '-'*75 )
for word, freq in freq_adjective2.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)


#   TOP 50 ADVERB TOKENS:
#####################################################################
```

```
adverb_tokens2 = []
for sentence in taggedtext2:
    for word, pos in sentence:
        if pos in ['RB', 'RBR', 'RBS']: # adverb, comparative, superlative
            if len(word) > 1:
                adverb_tokens2.append(word)
freq_adverb2 = nltk.FreqDist(adverb_tokens2)

print('\nText2 - Top 50 Adverb Tokens\n' + '-'*75 )
for word, freq in freq_adverb2.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)


#   TOP 50 NOUN TOKENS:
######################################################################
noun_tokens2 = []
for sentence in taggedtext2:
    for word, pos in sentence:
        if pos in ['NN', 'NNS', 'NNP', 'NNPS']: # noun, noun plural, proper noun,
proper noun plural
            if len(word) > 1:
                noun_tokens2.append(word)
freq_noun2 = nltk.FreqDist(noun_tokens2)

print('\nText2 - Top 50 Noun Tokens\n' + '-'*75 )
for word, freq in freq_noun2.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)


#   TOP 50 VERB TOKENS:
######################################################################
verb_tokens2 = []
for sentence in taggedtext2:
    for word, pos in sentence:
        if pos in ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']: # verb, verb past tense,
verb gerund, verb past principle, vrb non 3rd person singular, verb 3rd person
singular present   ural
            if len(word) > 1:
                verb_tokens2.append(word)
freq_verb2 = nltk.FreqDist(verb_tokens2)

print('\nText2 - Top 50 Verb Tokens\n' + '-'*75 )
for word, freq in freq_verb2.most_common(50):
    if len(word) < 7:
        tabMult = 3
    elif len(word) <= 14:
        tabMult = 2
    else:
        tabMult = 1
    print(word, '\t'*tabMult, freq)

#####################################################################
# Now we have two lists of POS tags combinations we can compare
# We need to get the sentences back from the tagging exercise and run some stats

# Create a list of original sentences from the ADJECTIVE phrase subset:
adjph_whole_sentences2 = []

# loop over the sentences in the adjective phrase sentences we created:
for sents in grammar_adjph_tags2:
    temp=''
    for (word, tag) in sents:
        temp += word+' '
        adjph_whole_sentences2.append(temp)

print('\n', '=' * 50, '\n', len(adjph_whole_sentences2))
```

```
For for the second  text I selected was "melville-moby_dick.txt

Text2 - Top adjective phrases by frequency:
---------------------------------------------------------------------------
so much                  23
so many                  21
once more                19
not so much              10
too much                 9
as much                  8
so wide                  6
very large               5
so good                  4
very little              4
as good                  4
so saying                4
little more              4
not much                 4
so small                 4
so full                  4
most appalling           4
so nigh                  4
very curious             4
very much                4
so great                 3
so important             3
still stranger           3
very similar             3
still more curious       3
very old                 3
Very good                3
so remarkable            3
well nigh                3
still greater            3
so strange               3
```

```
very easy                3
however much             3
far other                3
very learned             3
very severe              3
not true                 2
most monstrous           2
very great               2
so little                2
most convenient          2
most dangerous           2
just previous            2
too late                 2
more wonderful           2
still further            2
most direful             2
most violent             2
somewhat similar         2
most conscientious       2
Length of adjective phrase sentences:  1627

Text2 - Top adverb phrases by frequency:
---------------------------------------------------------------------------
not only                 50
as well                  35
not so                   28
so long                  18
so much                  15
so far                   13
not yet                  13
thus far                 12
so soon                  10
ere long                 10
so very                  9
as soon                  8
even so                  8
back again               8
so often                 7
very much                6
even then                6
down there               6
never yet                6
not altogether           5
far away                 5
Well then                5
not very                 5
so strangely             5
just now                 5
almost entirely          5
not always               5
still further            4
not then                 4
even now                 4
not possibly             4
not well                 4
ye now                   4
then again               4
```

```
very well                4
yet not                  4
there again              4
not even                 4
almost wholly            4
not wholly               4
aloft there              4
ye not                   4
not now                  4
very often               4
as much                  3
forward again            3
not far                  3
pretty nearly            3
very probably            3
just here                3
Length of adverb phrase sentences:  1235

Text2 - Top 50 Adjective Tokens
---------------------------------------------------------------------------
old                      429
other                    409
great                    290
last                     275
such                     258
more                     248
little                   238
same                     210
own                      201
long                     188
good                     173
first                    172
many                     161
white                    158
much                     131
small                    121
whole                    115
full                     111
poor                     104
few                      92
thy                      92
certain                  87
strange                  87
high                     86
most                     84
dead                     80
black                    79
large                    76
wild                     76
least                    75
young                    75
true                     73
vast                     69
sperm                    67
general                  64
present                  64
whale                    61
```

```
hard                     60
thou                     60
entire                   55
fine                     54
curious                  53
open                     52
second                   52
short                    51
broad                    51
lower                    51
best                     50
particular               49
ye                       48

Text2 - Top 50 Adverb Tokens
---------------------------------------------------------------------------
not                      1170
so                       769
now                      645
then                     570
only                     324
there                    321
still                    299
very                     294
here                     270
yet                      263
more                     252
again                    252
n't                      250
never                    195
ever                     190
most                     190
almost                   186
too                      182
even                     179
far                      155
away                     151
well                     148
once                     142
So                       141
Now                      139
down                     138
as                       119
long                     114
soon                     113
just                     110
thus                     107
also                     88
much                     87
back                     87
perhaps                  82
indeed                   82
up                       81
however                  80
always                   80
sometimes                76
rather                   68
```

```
  often                    68
  together                 64
  enough                   59
  Then                     55
  ere                      54
  ye                       51
  Well                     51
  aloft                    49
  therefore                49

  Text2 - Top 50 Noun Tokens
  ---------------------------------------------------------------------------
  whale                    718
  Ahab                     495
  man                      472
  ship                     441
  sea                      349
  time                     316
  boat                     280
  Whale                    270
  head                     266
  way                      262
  Stubb                    251
  Queequeg                 244
  whales                   232
  men                      231
  Captain                  212
  hand                     194
  Starbuck                 191
  thing                    188
  side                     178
  ye                       172
  world                    166
  water                    166
  Pequod                   164
  day                      157
  deck                     157
  eyes                     155
  sort                     151
  CHAPTER                  150
  part                     148
  boats                    140
  air                      138
  life                     135
  crew                     135
  Sperm                    133
  things                   132
  round                    131
  night                    130
  God                      128
  feet                     125
  hands                    120
  something                119
  thou                     113
  body                     110
  sir                      110
  times                    110
```

```
line                     108
captain                  108
moment                   105
place                    105
Flask                    104

Text2 - Top 50 Verb Tokens
---------------------------------------------------------------------------
is                       1722
was                      1639
be                       1027
had                      767
have                     763
were                     679
are                      586
's                       426
been                     415
do                       363
said                     302
has                      291
seemed                   282
did                      264
say                      233
see                      229
being                    219
go                       178
made                     175
seen                     160
know                     147
come                     138
cried                    138
came                     130
take                     117
thought                  114
found                    113
tell                     113
called                   112
saw                      109
let                      108
think                    107
make                     105
ye                       98
went                     97
heard                    96
does                     91
get                      90
stood                    86
am                       84
seems                    84
seem                     84
going                    79
look                     76
done                     74
known                    73
put                      70
give                     68
round                    67
```

```
stand                   66

 ==================================================
 3487
```

In [8]:
```
#---------------------------------------------------------------
```

In [9]:
```
#   PART 1.3a:  PERFORM STATISTICS:
    #   Now we have two lists of POS tags combinations we can compare
    #   We need to get the sentences back from the tagging exercise and run some stats
print('For for the first text I selected was "' + fileName1)

#   CREATE LIST OF ORIGINAL SENTENCES FROM THE ADJECTIVE PHRASE SUBSET:
adjph_whole_sentences1 = []
for sents in grammar_adjph_tags1:
    temp=''
    for (word, tag) in sents:
        temp += word+' '
        adjph_whole_sentences1.append(temp)
print('\nText1 - Length of Adjective Phrase Whole Sentences:\n', '-'*75)
print(len(adjph_whole_sentences1))


#   CREATE LIST OF ORIGINAL SENTENCES FROM THE ADVERB PHRASE SUBSET:
advph_whole_sentences1 = []
for sents in grammar_advph_tags1:
    temp = ''
    for (word, tag) in sents:
        temp += word+' '
        advph_whole_sentences1.append(temp)
print('\nText1 - Length of Adverbe Phrase Whole Sentences:\n', '-'*75)
print(len(advph_whole_sentences1))


#   COBMINE LISTS TOGETHER TO HAVE A SINGLE LIST OF ADJECTIVE/ADVERB PHRASES:
    # Useful to know which sentences are heavy in qualifiers
adv_adj_phrase_sentences1 = adjph_whole_sentences1
for sent in advph_whole_sentences1:
    # if a sentence is not in the adjective phrases list imported
    if sent not in adv_adj_phrase_sentences1:
        # attach that sentence
        adv_adj_phrase_sentences1.append(sent)
print('\nText1 - Length of Adverbe + Adjective Phrase Whole Sentences:\n', '-'*75)
print(len(adv_adj_phrase_sentences1))

#######################################################################################

# Following our NLTK textbook, Writing Structural Programs chapter
# section on Procedural vs Declarative style (http://www.nltk.org/book_1ed/ch04.html)
#######################################################################################


#   CORPUS STATISTICS--SENTENCES LENGTH
    # Calculating the average length of sentences in the entire corpus
    # from http://www.nltk.org/book_1ed/ch04.html
total_corpus1 = sum(len(sent) for sent in textsplit1) # remember: 'textsplit' is our
text split into sentences
print('\nText1 - Average Length of Sentences:\n', '-'*75)
```

```
    print(total_corpus1 / len(textsplit1))


    #   CALCULATE THE AVERAGE LENGTH OF ADJECTIVE PHRASE SENTENCE:
        # We can then compare the average length of the adjective phrases to
        # the average sentences we calculated for all sentences in the corpus
    total_adjph_sentences1 = sum(len(sent) for sent in adjph_whole_sentences1)
    # adjph_whole_sentences stores our adjective phrases
    print('\nText1 - Average Length of Adjective Phrases:\n', '-'*75)
    print(total_adjph_sentences1 / len(adjph_whole_sentences1))
```

```
For for the first text I selected was "carroll-alice.txt

Text1 - Length of Adjective Phrase Whole Sentences:
 ---------------------------------------------------------------------------
464

Text1 - Length of Adverbe Phrase Whole Sentences:
 ---------------------------------------------------------------------------
484

Text1 - Length of Adverbe + Adjective Phrase Whole Sentences:
 ---------------------------------------------------------------------------
633

Text1 - Average Length of Sentences:
 ---------------------------------------------------------------------------
87.26892307692307

Text1 - Average Length of Adjective Phrases:
 ---------------------------------------------------------------------------
9.262243285939968
```

In [10]:
```
#----------------------------------------------------------------------------------------
--------------------
```

In [11]:
```
#   PART 1.3b:  PERFORM STATISTICS:
    #   Now we have two lists of POS tags combinations we can compare
    #   We need to get the sentences back from the tagging exercise and run some stats
print('For for the second text I selected was "' + fileName2)


#   CREATE LIST OF ORIGINAL SENTENCES FROM THE ADJECTIVE PHRASE SUBSET:
adjph_whole_sentences2 = []
for sents in grammar_adjph_tags2:
    temp=''
    for (word, tag) in sents:
        temp += word+' '
        adjph_whole_sentences2.append(temp)
print('\nText2 - Length of Adjective Phrase Whole Sentences:\n', '-'*75)
print(len(adjph_whole_sentences2))


#   CREATE LIST OF ORIGINAL SENTENCES FROM THE ADVERB PHRASE SUBSET:
advph_whole_sentences2 = []
for sents in grammar_advph_tags2:
    temp = ''
    for (word, tag) in sents:
        temp += word+' '
        advph_whole_sentences2.append(temp)
print('\nText2 - Length of Adverbe Phrase Whole Sentences:\n', '-'*75)
print(len(advph_whole_sentences2))


#   COBMINE LISTS TOGETHER TO HAVE A SINGLE LIST OF ADJECTIVE/ADVERB PHRASES:
    # Useful to know which sentences are heavy in qualifiers
adv_adj_phrase_sentences2 = adjph_whole_sentences2
for sent in advph_whole_sentences2:
    # if a sentence is not in the adjective phrases list imported
    if sent not in adv_adj_phrase_sentences2:
        # attach that sentence
        adv_adj_phrase_sentences2.append(sent)
print('\nText2 - Length of Adverbe + Adjective Phrase Whole Sentences:\n', '-'*75)
print(len(adv_adj_phrase_sentences2))

#############################################################################################

# Following our NLTK textbook, Writing Structural Programs chapter
# section on Procedural vs Declarative style (http://www.nltk.org/book_1ed/ch04.html)
#############################################################################################


#   CORPUS STATISTICS--SENTENCES LENGTH
    # Calculating the average length of sentences in the entire corpus
    # from http://www.nltk.org/book_1ed/ch04.html
total_corpus2 = sum(len(sent) for sent in textsplit2) # remember: 'textsplit' is our text split into sentences
print('\nText2 - Average Length of Sentences:\n', '-'*75)
print(total_corpus2 / len(textsplit2))


#   CALCULATE THE AVERAGE LENGTH OF ADJECTIVE PHRASE SENTENCE:
    # We can then compare the average length of the adjective phrases to
    # the average sentences we calculated for all sentences in the corpus
total_adjph_sentences2 = sum(len(sent) for sent in adjph_whole_sentences2)
# adjph_whole_sentences stores our adjective phrases
print('\nText2 - Average Length of Adjective Phrases:\n', '-'*75)
print(total_adjph_sentences2 / len(adjph_whole_sentences2))
```

```
For for the second text I selected was "melville-moby_dick.txt

Text2 - Length of Adjective Phrase Whole Sentences:
 ---------------------------------------------------------------------------
3487

Text2 - Length of Adverbe Phrase Whole Sentences:
 ---------------------------------------------------------------------------
2563

Text2 - Length of Adverbe + Adjective Phrase Whole Sentences:
 ---------------------------------------------------------------------------
4234

Text2 - Average Length of Sentences:
 ---------------------------------------------------------------------------
```

```
123.65489240763297

Text2 - Average Length of Adjective Phrases:
 -----------------------------------------------------------------------------
10.43410486537553
```

PART 2: INTERPRETATION OF THE RESULTS:

- Provide interpretation of the Part 1 results.
- What do you learn from these results?

The sentiment analysis had mixed results. In terms of being able to distinguish between the two books (Alice in Wonderland and Moby Dick), the sentiment analysis did allow these books to be distinguished and identified. This is because the analysis did a good job at picking out specific sentiment phrases and did a great job with finding key nouns. The issue came in when trying to connect the specific sentiment to the specific objects or nouns with the two stories. This portion, in my opinion, is where the sentiment analysis faultered or stumbled. It was like we had a collection of objects and a collection of sentiments but not a good way to know which one went with which.

A good analogy of the current state of sentiment analysis would be two tables in a database. One table has the sentiment phrases and the second had the objects or nouns. I believe the sentiment analysis techniques used can successfully populate those two tables. The part that is still missing and needs further development is relating those two tables together correctly. That is, identifying which records in one table are associated with the records in the other. Without that proper relating the sentiment analysis loses the context and thus becomes difficult to interpret. The good news is that since we know what the 'missing link' is (pun intended) we know where we need to spend our efforts.

I learned that NLP is much more complicated than I ever gave thought to in the past and that substantial progess has been made in this area. However, there is still much more to do to get to a point in which a computer can understand Natural Language. The challenges have been identified and while progress is slow it is definitely something that we will figure out one day. I'm not sure we will figure it out in 5 years or 50 years from now but I strongly believe it is something humanity will accomplish eventually.

On very macro scale I have come to believe that AI and NLP are far more related then I ever thought possible. While many metrics are undoubtadly available to give an indication of how advance an AI system is I suspect in the end the truest test will can we talk to it and it understand what we are saying. That will be the point in which AI will have reached a turning point in the development of these systems.

In [12]:
```
#------------------------------------------------------------------------------------------------------
```

PART 3: VISION FOR THE FUTURE

- The last thing to do is to envision how you would conduct a sentiment analysis on the review texts.

I can envision sentiment analysis being used in two complementary ways. In the short term, I think the sentiment analysis will be used to process large volumes of short texts in which the topic is highly focused. For example, these techniques could be used to analyze customer product reviews or comment boards. The reason for these limitations is due to the challenge of accurately and consistently attributing specific sentiment to a specific topic or subject.

In addition, in the near term, I believe there will a great deal to improve our systems ability to accurately contribute sentiment to a specific subject or topic. This will be a major topic of study, analysis, and will sure to have triumphs. Unfortunately it is highly likely that progress will remain steady and slow. This gives a great deal of opportunity for research and develoment for those interested in this field of exploration.

In the long term, I can see sentiment analysis being used in virtually every area of our lives. As sentiment analysis improves and accuracy increases the application space will grow wider and more complicated. So, in the short term, perhaps relatively simple texts in which the subject or topic is limited. Over time more complicated texts and stories until reading is as easy for the computer to do as doing mathematics currently is. This is truly an exciting time in the field.

In [13]:

```
#-------------------------------------------------------------------------------------------------------------------------------
```

In [ ]:

In [ ]: