In [1]:
```
import nltk
from nltk.corpus import gutenberg
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.collocations import BigramAssocMeasures, BigramCollocationFinder
```

In [2]:
```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('gutenberg')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
```

Out[2]:
```
True
```

In [5]:
```
files = gutenberg.fileids()
files
```

Out[5]:
```
['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']
```

In [8]:
```
files0 = files[0]
files0
```

Out[8]:
```
'austen-emma.txt'
```

In [9]:
```
files5 = files[5]
files5
```

Out[9]:
```
'bryant-stories.txt'
```

In [29]:
```
Book1 = gutenberg.raw(files0)
Book2 = gutenberg.raw(files5)
```

In [135…
```
import string
import nltk

# Downloads the 'punkt' tokenizer, which is used for sentence tokenization
```

```
nltk.download('punkt')

# Downloads the 'stopwords' corpus, which is a list of commonly used words (such as
nltk.download('stopwords')
nltk_stops = nltk.corpus.stopwords.words('english')
morestopwords = ['could','would','might','must','need','sha','wo','y',"'s","'d","'l
stopwords = nltk_stops + morestopwords


nltk.download('wordnet')
nltk.download('omw-1.4')
wnl = nltk.WordNetLemmatizer()
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [ ]:
```
#Tokenization
```

In [136…
```
tokenfile_B1 = nltk.word_tokenize(Book1)
```

In [137…
```
tokenfile_B2 = nltk.word_tokenize(Book2)
```

In [ ]:
```
#LowerCases
```

In [138…
```
Book1_lower = [i.lower() for i in tokenfile_B1]
```

In [139…
```
Book2_lower = [i.lower() for i in tokenfile_B2]
```

In [ ]:
```
# Text Filtering
```

In [140…
```
Book1_words = [w for w in Book1_lower if w.isalpha()]
```

In [ ]:
```
#StopWords
```

In [141…
```
Book2_words = [w for w in Book2_lower if w.isalpha()]
```

In [142…
```
Book1_stop = [i for i in Book1_words if i in nltk_stops]
```

In [143…
```
Book2_stop = [j for j in Book2_words if j in nltk_stops]
```

In [144…
```
Book1_withoutstop = [i for i in Book1_words if i not in nltk_stops]
```

In [145…
```
Book2_withoutstop = [j for j in Book2_words if j not in nltk_stops]
```

In [ ]:
```
#Lemmatization
```

In [146…
```
Book1_withoutstopLemma= [wnl.lemmatize(j) for j in Book1_withoutstop]
```

In [ ]:
```
#Data Analysis 1:
```

In [148…
```
from nltk import FreqDist
fdist = FreqDist(Book1_withoutstopLemma)
fdist
```

Out[148]:
```
FreqDist({'emma': 855, 'could': 836, 'would': 818, 'miss': 600, 'must': 566, 'harr
iet': 496, 'much': 484, 'said': 483, 'thing': 456, 'one': 451, ...})
```

In [149…
```
fdist1 = FreqDist(Book2_withoutstopLemma)
fdist1
```

Out[149]:
```
FreqDist({'little': 596, 'said': 453, 'came': 191, 'one': 188, 'could': 172, 'kin
g': 135, 'went': 122, 'would': 113, 'time': 110, 'great': 110, ...})
```

In [150…
```
Book1_top50content = fdist.most_common(50)

for i in Book1_top50content:
    print (i)  # finding the top 50 content words for Book1 data set
```

```
('emma', 855)
('could', 836)
('would', 818)
('miss', 600)
('must', 566)
('harriet', 496)
('much', 484)
('said', 483)
('thing', 456)
('one', 451)
('weston', 438)
('every', 435)
('think', 406)
('well', 378)
('elton', 378)
('knightley', 373)
('know', 365)
('little', 359)
('never', 358)
('say', 341)
('might', 325)
('good', 313)
('woodhouse', 308)
('time', 303)
('jane', 301)
('quite', 282)
('great', 263)
('thought', 262)
('friend', 257)
('nothing', 252)
('dear', 243)
('always', 238)
('man', 232)
('fairfax', 232)
('churchill', 229)
('see', 225)
('soon', 223)
('may', 221)
('shall', 217)
('without', 214)
('day', 209)
('frank', 207)
('first', 205)
('like', 202)
('father', 201)
('sure', 201)
('made', 199)
('indeed', 196)
('come', 195)
('body', 193)
```

In [151…

```
Book2_top50content = fdist1.most_common(50)

for i in Book2_top50content:
    print (i) # finding the top 50 content words for Book2 data set
```

```
('little', 596)
('said', 453)
('came', 191)
('one', 188)
('could', 172)
('king', 135)
('went', 122)
('would', 113)
('time', 110)
('great', 110)
('day', 108)
('man', 105)
('old', 102)
('see', 99)
('saw', 92)
('like', 91)
('come', 90)
('mother', 90)
('away', 90)
('made', 89)
('jackal', 84)
('go', 78)
('father', 78)
('good', 77)
('people', 76)
('looked', 76)
('tree', 75)
('know', 75)
('make', 71)
('margery', 71)
('thought', 70)
('ran', 69)
('big', 69)
('boy', 68)
('thing', 68)
('child', 65)
('two', 64)
('home', 64)
('put', 62)
('every', 62)
('door', 61)
('way', 60)
('lion', 60)
('long', 58)
('never', 58)
('took', 57)
('head', 57)
('look', 57)
('much', 57)
('back', 56)
```

In [ ]:
```
#Data Analysis 2: doing the frequency distrubution of top 50 stop words
```

In [152…
```
fdist_stop = FreqDist(Book1_stop)
print(fdist_stop)
```

```
<FreqDist with 124 samples and 87421 outcomes>
```

In [153…
```
fdist1_stop = FreqDist(Book2_stop)
print(fdist1_stop)
```

```
<FreqDist with 122 samples and 24471 outcomes>
```

In [154…
```
Book1_top50 = fdist_stop.most_common(50)

for i in Book1_top50:
    print (i) # top 50 stopwords for Book1 dataset
```

```
('the', 5201)
('to', 5181)
('and', 4877)
('of', 4284)
('i', 3177)
('a', 3124)
('it', 2503)
('her', 2448)
('was', 2396)
('she', 2336)
('not', 2281)
('in', 2173)
('be', 1970)
('you', 1967)
('he', 1806)
('that', 1805)
('had', 1623)
('but', 1441)
('as', 1436)
('for', 1346)
('have', 1320)
('is', 1241)
('with', 1215)
('very', 1202)
('his', 1141)
('at', 1030)
('so', 968)
('all', 841)
('been', 755)
('him', 749)
('no', 741)
('my', 728)
('on', 689)
('any', 654)
('do', 652)
('were', 599)
('by', 569)
('me', 563)
('which', 556)
('will', 556)
('there', 548)
('from', 546)
('they', 540)
('what', 536)
('this', 526)
('or', 494)
('such', 489)
('if', 485)
('more', 466)
('an', 463)
```

In [155…
```
Book2_top50 = fdist1_stop.most_common(50)

for i in Book2_top50:
    print (i) # Finding the top 50 stop words for Book2 data set
```

```
('the', 3449)
('and', 2097)
('to', 1158)
('a', 1029)
('he', 1016)
('of', 818)
('was', 720)
('in', 640)
('it', 614)
('his', 551)
('i', 533)
('that', 531)
('you', 462)
('she', 409)
('they', 396)
('for', 339)
('as', 327)
('but', 308)
('him', 296)
('so', 295)
('had', 295)
('her', 291)
('with', 284)
('when', 271)
('on', 262)
('at', 261)
('is', 247)
('not', 242)
('all', 239)
('there', 218)
('out', 195)
('were', 194)
('me', 188)
('then', 185)
('them', 185)
('up', 178)
('be', 169)
('this', 161)
('from', 158)
('very', 157)
('have', 154)
('do', 150)
('will', 147)
('down', 134)
('my', 126)
('who', 125)
('what', 124)
('their', 112)
('no', 110)
('are', 104)
```

In [ ]:
```
#Data Analysis 3: analysing the top 50 frequency bigrams
```

In [156…
```
from nltk.util import ngrams
bigrams = [w for w in ngrams(Book1_lower, 2)]
fdist_bigram = FreqDist(bigrams)
fdist_bigram.most_common(50)
```

Out[156]:
```
[((',', 'and'), 1882),
 (('.', "'"), 1157),
 (("'", '``'), 959),
 ((';', 'and'), 867),
 (('to', 'be'), 605),
 ((',', "'"), 584),
 (('.', 'i'), 570),
 ((',', 'i'), 569),
 (('of', 'the'), 559),
 (('in', 'the'), 445),
 (('it', 'was'), 442),
 ((';', 'but'), 427),
 (('.', '``'), 416),
 (('.', 'she'), 413),
 (('i', 'am'), 394),
 ((',', 'that'), 360),
 (('!', '--'), 344),
 (('--', 'and'), 334),
 (('she', 'had'), 332),
 (('she', 'was'), 328),
 (('had', 'been'), 307),
 ((',', 'she'), 304),
 ((',', 'but'), 303),
 (('.', 'he'), 303),
 (('it', 'is'), 298),
 ((',', 'as'), 292),
 (('i', 'have'), 281),
 (('could', 'not'), 278),
 (('mr.', 'knightley'), 273),
 (('.', 'it'), 266),
 (("'", 'said'), 265),
 ((',', 'to'), 264),
 (('``', 'i'), 261),
 (('of', 'her'), 260),
 (('--', 'i'), 257),
 (('.', 'the'), 251),
 ((',', '``'), 250),
 (('mrs.', 'weston'), 246),
 (('have', 'been'), 241),
 (('he', 'had'), 240),
 (('?', "'"), 238),
 ((',', 'in'), 237),
 (('to', 'the'), 237),
 (('do', 'not'), 235),
 (('--', 'but'), 232),
 ((',', 'the'), 226),
 (('and', 'the'), 224),
 (('he', 'was'), 222),
 (('would', 'be'), 215),
 ((',', 'it'), 214)]
```

In [157…
```
from nltk.util import ngrams
bigrams1 = [v for v in ngrams(Book2_lower, 2)]
fdist1_bigram = FreqDist(bigrams1)
fdist1_bigram.most_common(50)
```

Out[157]:
```
[((',', 'and'), 917),
 (('.', '``'), 329),
 ((',', "''"), 299),
 (('the', 'little'), 296),
 (("''", 'said'), 253),
 (('of', 'the'), 240),
 ((',', '``'), 238),
 (('in', 'the'), 232),
 (('!', "''"), 219),
 (('.', "''"), 218),
 (('.', 'the'), 211),
 (('said', 'the'), 191),
 (('and', 'the'), 182),
 (('to', 'the'), 176),
 (("''", '``'), 146),
 (('?', "''"), 140),
 ((',', 'the'), 140),
 (('it', 'was'), 121),
 (('said', ','), 116),
 (('.', 'and'), 115),
 (('``', 'i'), 114),
 (('he', 'was'), 112),
 (('.', 'he'), 109),
 (('.', 'but'), 105),
 ((',', 'but'), 104),
 (('on', 'the'), 102),
 ((',', '--'), 100),
 (('and', 'he'), 100),
 ((',', 'he'), 99),
 (('a', 'little'), 97),
 ((';', 'and'), 84),
 (("''", 'the'), 81),
 (('was', 'a'), 79),
 (('that', 'he'), 78),
 (('at', 'the'), 75),
 (('the', 'king'), 75),
 (('in', 'a'), 73),
 (('he', 'had'), 72),
 (('.', 'when'), 70),
 (("''", 'and'), 67),
 (('when', 'the'), 67),
 (('when', 'he'), 67),
 (('he', 'said'), 66),
 (('--', "''"), 65),
 (('to', 'be'), 65),
 (('.', 'then'), 63),
 (('.', 'it'), 62),
 (("''", 'he'), 62),
 (('from', 'the'), 61),
 ((',', 'to'), 61)]
```

In [ ]:
```
#Data Analysis 4:analysing the top 50 frequency
#bi grams with mutual information and a minimum frequency of 5
```

In [158…
```
from nltk.collocations import *
from nltk.collocations import BigramAssocMeasures
from nltk.collocations import BigramCollocationFinder
bigram_measures = nltk.collocations.BigramAssocMeasures()
finder1 = BigramCollocationFinder.from_words(Book1_lower)
finder1.apply_freq_filter(5)
scored = finder1.score_ngrams(bigram_measures.pmi)
for bscore in scored[:50]:
    print (bscore)
```

```
(('d', "'ye"), 14.964167861580208)
(('sore', 'throat'), 14.089698743664066)
(('brunswick', 'square'), 13.952195219914133)
(('william', 'larkins'), 13.089698743664067)
(('baked', 'apples'), 12.964167861580208)
(('box', 'hill'), 12.736061789049367)
(('sixteen', 'miles'), 12.613670614496076)
(('maple', 'grove'), 12.594934051914489)
(('hair', 'cut'), 12.063703535131124)
(('south', 'end'), 11.96416786158021)
(('colonel', 'campbell'), 11.412234161246522)
(('protest', 'against'), 11.347496501131715)
(('robert', 'martin'), 11.093935736550536)
(('five', 'couple'), 10.841771230220482)
(('vast', 'deal'), 10.76253400041056)
(('ready', 'wit'), 10.652293431356767)
(('donwell', 'abbey'), 10.519383018907314)
(('musical', 'society'), 10.509114683453486)
(('infinitely', 'superior'), 10.230813520966382)
(('married', 'women'), 10.05727726597169)
(('five', 'minutes'), 10.032714012931878)
(('years', 'ago'), 9.9575041312992)
(('three', 'months'), 9.941800048551755)
(('depend', 'upon'), 9.928125111654678)
(('ten', 'minutes'), 9.867013597351292)
(('sat', 'down'), 9.795356480448604)
(('hurrying', 'away'), 9.603101372785888)
(('few', 'moments'), 9.558175501904373)
(('few', 'minutes'), 9.41521754806233)
(('lovely', 'woman'), 9.400399583128175)
(('ten', 'years'), 9.372541630578041)
(('last', 'night'), 9.368910380131174)
(('sit', 'down'), 9.341844833355125)
(('frank', 'churchill'), 9.284101419843205)
(('few', 'lines'), 9.236247407017009)
(('take', 'care'), 9.18038141066101)
(('worthy', 'people'), 9.146544604068778)
(('thrown', 'away'), 9.088528199956128)
(('dare', 'say'), 9.06278824963678)
(('three', 'times'), 9.04133572210267)
(('next', 'week'), 9.035797538561239)
(('few', 'weeks'), 9.01385498568056)
(('how', 'd'), 9.01385498568056)
(('great', 'deal'), 8.98664941695205)
(('dear', 'madam'), 8.935801307930312)
(('common', 'sense'), 8.934420518186156)
(('jane', 'fairfax.'), 8.830083858371422)
(('common', 'course'), 8.811038102680875)
(('sitting', 'down'), 8.795356480448604)
(('each', 'other'), 8.767770648776706)
```

In [159…

```
bigram_measures1 = nltk.collocations.BigramAssocMeasures()
finder2 = BigramCollocationFinder.from_words(Book2_lower)
finder2.apply_freq_filter(5)
scored1 = finder2.score_ngrams(bigram_measures1.pmi)
for bscore in scored1[:50]:
    print (bscore)
```

```
(('herr', 'grupello'), 13.441491779304727)
(('rid', 'hin'), 12.17845737347093)
(('royal', 'robes'), 12.17845737347093)
(('trundle-bed', 'boat'), 11.941418176170085)
(('jack', 'rollaround'), 11.67595703294175)
(('*', '*'), 11.534601183696205)
(('new', 'orleans'), 11.239857918135073)
(('clock', 'struck'), 11.166484731804855)
(('shiny', 'acorn'), 11.108068045579532)
(('christ', 'child'), 11.062980156050996)
(('thou', 'art'), 10.997885127829111)
(('small', 'rid'), 10.54102745285564)
(('brother', 'rabbit'), 10.441491779304727)
(('white', 'garraun'), 10.371102451413325)
(('mr', 'alligator'), 10.348382374913244)
(('red', 'hen'), 10.280336987490145)
(('anything', 'else'), 10.119563684417363)
(('god', 'save'), 10.062980156050994)
(('fir', 'tree'), 9.888385547633698)
(('prince', 'cherry'), 9.694257849684695)
(('your', 'majesty'), 9.32047637834336)
(('gingerbread', 'boy'), 9.303988255554792)
(('field', 'mouse'), 9.210110574741021)
(('country', 'mouse'), 9.197619630557828)
(('ca', "n't"), 9.163507032004961)
(('ai', "n't"), 9.16350703200496)
(('city', 'mouse'), 9.104315910718427)
(('long', 'ago'), 9.030969761148373)
(('great', 'lizard'), 8.982060160667428)
(('lazy', 'man'), 8.85652927858357)
(('better', 'than'), 8.778526766582296)
(('king', 'solomon'), 8.708137438690898)
(('their', 'heads'), 8.69303054630069)
(('most', 'beautiful'), 8.685269066457439)
(('take', 'care'), 8.576421359390835)
(("'", 'thin'), 8.566554329931499)
(("n't", 'catch'), 8.447299998005553)
(('an', "'"), 8.387489340259151)
(('great', 'deal'), 8.30398825555479)
(('wonder', 'if'), 8.286066347557528)
(('took', 'hold'), 8.278453163447654)
(('place', 'where'), 8.274240912160737)
(('pretty', 'soon'), 8.232038413675774)
(('old', 'alligator'), 8.090994532220593)
(('could', 'hardly'), 8.074120713656196)
(('old', 'woman'), 7.9654636501367335)
(('should', 'eat'), 7.965295278188025)
(('tiger', 'should'), 7.891001495840708)
(('run', 'away'), 7.842723479058538)
(('grey', 'man'), 7.826781935189516)
```

In [160…

```
from nltk.collocations import *
from nltk.collocations import BigramAssocMeasures
from nltk.collocations import BigramCollocationFinder
bigram_measures = nltk.collocations.BigramAssocMeasures()
finder1 = BigramCollocationFinder.from_words(Book1_lower)
finder1.apply_freq_filter(5)
scored = finder1.score_ngrams(bigram_measures.raw_freq)
for bscore in scored[:50]:
    print (bscore)
```

```
((',', 'and'), 0.009813071929504393)
(('.', "''"), 0.006032797142633679)
(("''", '``'), 0.005000391062908987)
((';', 'and'), 0.004520687227885393)
(('to', 'be'), 0.0031545741324921135)
((',', "''"), 0.0030450765179758582)
(('.', 'i'), 0.002972078108298355)
((',', 'i'), 0.0029668639361785333)
(('of', 'the'), 0.0029147222149803163)
(('in', 'the'), 0.0023203065933206455)
(('it', 'was'), 0.0023046640769611806)
((';', 'but'), 0.002226451495163855)
(('.', '``'), 0.002169095601845817)
(('.', 'she'), 0.002153453085486352)
(('i', 'am'), 0.00205438381520974)
((',', 'that'), 0.001877101963135803)
(('!', '--'), 0.0017936752092186563)
(('--', 'and'), 0.0017415334880204396)
(('she', 'had'), 0.0017311051437807962)
(('she', 'was'), 0.0017102484553015095)
(('had', 'been'), 0.0016007508407852543)
((',', 'she'), 0.0015851083244257892)
((',', 'but'), 0.0015798941523059676)
(('.', 'he'), 0.0015798941523059676)
(('it', 'is'), 0.0015538232917068592)
((',', 'as'), 0.0015225382589879291)
(('i', 'have'), 0.0014651823656698908)
(('could', 'not'), 0.0014495398493104257)
(('mr.', 'knightley'), 0.0014234689887113175)
(('.', 'it'), 0.0013869697838725656)
(("''", 'said'), 0.0013817556117527439)
((',', 'to'), 0.0013765414396329223)
((``', 'i'), 0.0013608989232734572)
(('of', 'her'), 0.0013556847511536356)
(('--', 'i'), 0.0013400422347941705)
(('.', 'the'), 0.0013087572020752405)
((',', '``'), 0.00130354302995419)
(('mrs.', 'weston'), 0.0012826863414761322)
(('have', 'been'), 0.0012566154808770237)
(('he', 'had'), 0.0012514013087572022)
(('?', "''"), 0.0012409729645175588)
((',', 'in'), 0.00123575879239737)
(('to', 'the'), 0.00123575879239737)
(('do', 'not'), 0.0012253304481580937)
(('--', 'but'), 0.0012096879317986286)
((',', 'the'), 0.0011784028990796985)
(('and', 'the'), 0.0011679745548400552)
(('he', 'was'), 0.0011575462106004119)
(('would', 'be'), 0.0011210470057616603)
((',', 'it'), 0.0011158328336418385)
```

In [161…

```
bigram_measures1 = nltk.collocations.BigramAssocMeasures()
finder2 = BigramCollocationFinder.from_words(Book2_lower)
finder2.apply_freq_filter(5)
scored1 = finder2.score_ngrams(bigram_measures1.raw_freq)
for bscore in scored1[:50]:
    print (bscore)
```

```
((',', 'and'), 0.016485689630375378)
(('.', '``'), 0.005914713073493456)
((',', "''"), 0.005375377534877031)
(('the', 'little'), 0.005321443981015389)
(("''", 'said'), 0.00454839637566518)
(('of', 'the'), 0.004314684308931396)
((',', '``'), 0.004278728606356968)
(('in', 'the'), 0.0041708614986336835)
(('!', "''"), 0.0039371494318998996)
(('.', "''"), 0.0039191715806126855)
(('.', 'the'), 0.003793326621602186)
(('said', 'the'), 0.003433769595857903)
(('and', 'the'), 0.0032719689342729755)
(('to', 'the'), 0.003164101826549691)
(("''", '``'), 0.0026247662879332664)
((',', 'the'), 0.0025168991802099814)
(('?', "''"), 0.0025168991802099814)
(('it', 'was'), 0.0021753200057529126)
(('said', ','), 0.0020854307493168417)
(('.', 'and'), 0.0020674528980296277)
(('``', 'i'), 0.002049475046742413)
(('he', 'was'), 0.002013519344167985)
(('.', 'he'), 0.0019595857903063427)
(('.', 'but'), 0.001887674385157486)
((',', 'but'), 0.001869696533870272)
(('on', 'the'), 0.0018337408312958435)
((',', '--'), 0.0017977851287214151)
(('and', 'he'), 0.0017977851287214151)
((',', 'he'), 0.001779807277434201)
(('a', 'little'), 0.0017438515748597727)
((';', 'and'), 0.0015101395081259887)
(("''", 'the'), 0.0014562059542643463)
(('was', 'a'), 0.0014202502516899181)
(('that', 'he'), 0.0014022724004027038)
(('at', 'the'), 0.0013483388465410613)
(('the', 'king'), 0.0013483388465410613)
(('in', 'a'), 0.0013123831439666332)
(('he', 'had'), 0.0012944052926794189)
(('.', 'when'), 0.0012584495901049907)
(("''", 'and'), 0.0012045160362433483)
(('when', 'he'), 0.0012045160362433483)
(('when', 'the'), 0.0012045160362433483)
(('he', 'said'), 0.001186538184956134)
(('--', "''"), 0.0011685603336689199)
(('to', 'be'), 0.0011685603336689199)
(('.', 'then'), 0.0011326046310944915)
(("''", 'he'), 0.0011146267798072774)
(('.', 'it'), 0.0011146267798072774)
((',', 'to'), 0.0010966489285200633)
(('from', 'the'), 0.0010966489285200633)
```

In [ ]:
```
# Trigram
```

In [175…
```
from collections import Counter
from nltk.util import ngrams
```

In [176…
```
trigrams_Book1 = list(ngrams(Book1_withoutstopLemma, 3))
trigrams_Book2 = list(ngrams(Book2_withoutstopLemma, 3))
```

In [177…
```
trigram_freq_Book1 = Counter(trigrams_Book1)
trigram_freq_Book2 = Counter(trigrams_Book2)
```

In [178…
```
trigram_freq_Book1_top50 = trigram_freq_Book1.most_common(50)
trigram_freq_Book2_top50 = trigram_freq_Book2.most_common(50)
```

In [179…
```
for trigram, freq in trigram_freq_Book1_top50:
    print(trigram, ":", freq)
```

```
('dear', 'miss', 'woodhouse') : 24
('oh', 'miss', 'woodhouse') : 14
('poor', 'miss', 'taylor') : 11
('said', 'frank', 'churchill') : 10
('miss', 'woodhouse', 'would') : 8
('fine', 'young', 'man') : 7
('said', 'john', 'knightley') : 7
('frank', 'churchill', 'miss') : 7
('miss', 'smith', 'miss') : 7
('every', 'body', 'else') : 6
('miss', 'woodhouse', 'think') : 6
('would', 'never', 'marry') : 5
('great', 'deal', 'better') : 5
('miss', 'bates', 'miss') : 5
('miss', 'woodhouse', 'must') : 5
('well', 'miss', 'woodhouse') : 5
('said', 'emma', 'smiling') : 5
('amiable', 'young', 'man') : 5
('every', 'body', 'would') : 5
('miss', 'bates', 'niece') : 5
('dare', 'say', 'shall') : 5
('miss', 'woodhouse', 'miss') : 5
('bates', 'miss', 'fairfax') : 5
('miss', 'woodhouse', 'said') : 5
('think', 'miss', 'woodhouse') : 5
('frank', 'churchill', 'come') : 4
('tell', 'every', 'thing') : 4
('every', 'thing', 'else') : 4
('emma', 'could', 'feel') : 4
('said', 'emma', 'laughing') : 4
('talked', 'great', 'deal') : 4
('woman', 'lovely', 'woman') : 4
('never', 'saw', 'thing') : 4
('shall', 'never', 'forget') : 4
('dare', 'say', 'would') : 4
('charming', 'young', 'man') : 4
('jane', 'fairfax', 'could') : 4
('saw', 'jane', 'fairfax') : 4
('miss', 'fairfax', 'must') : 4
('emma', 'could', 'imagine') : 4
('emma', 'could', 'help') : 4
('marrying', 'jane', 'fairfax') : 4
('churchill', 'miss', 'woodhouse') : 4
('know', 'miss', 'woodhouse') : 4
('miss', 'bates', 'came') : 4
('box', 'hill', 'party') : 4
('miss', 'taylor', 'would') : 3
('would', 'great', 'deal') : 3
('well', 'said', 'emma') : 3
('young', 'man', 'great') : 3
```

In [180…
```
for trigram, freq in trigram_freq_Book2_top50:
    print(trigram, ":", freq)
```

```
('little', 'gingerbread', 'boy') : 25
('little', 'fir', 'tree') : 25
('little', 'red', 'hen') : 22
('said', 'little', 'jackal') : 18
('little', 'jack', 'rollaround') : 12
('little', 'red', 'man') : 10
('little', 'old', 'woman') : 9
('little', 'country', 'mouse') : 9
('small', 'rid', 'hin') : 9
('mammy', 'said', 'epaminondas') : 8
('old', 'woman', 'little') : 7
('woman', 'little', 'old') : 7
('little', 'old', 'man') : 7
('little', 'brother', 'rabbit') : 7
('said', 'little', 'red') : 6
('run', 'run', 'fast') : 6
('run', 'fast', 'ca') : 6
('fast', 'ca', 'catch') : 6
('ca', 'catch', 'gingerbread') : 6
('catch', 'gingerbread', 'man') : 6
('could', 'catch', 'little') : 6
('god', 'save', 'said') : 6
('old', 'white', 'garraun') : 6
('said', 'little', 'tulip') : 5
('upon', 'time', 'little') : 5
('gingerbread', 'boy', 'ran') : 5
('run', 'away', 'little') : 5
('away', 'little', 'old') : 5
('little', 'city', 'mouse') : 5
('little', 'field', 'mouse') : 5
('mr', 'alligator', 'kind') : 5
('eat', 'set', 'free') : 5
('stood', 'quite', 'still') : 4
('said', 'goose', 'said') : 4
('goose', 'said', 'duck') : 4
('ran', 'fast', 'could') : 4
('catch', 'little', 'gingerbread') : 4
('eat', 'little', 'gingerbread') : 4
('old', 'man', 'cow') : 4
('gingerbread', 'boy', 'jumped') : 4
('little', 'jackal', 'know') : 4
('roll', 'around', 'roll') : 4
('around', 'roll', 'around') : 4
('madrid', 'see', 'king') : 4
('idea', 'said', 'little') : 4
('last', 'one', 'day') : 4
('little', 'small', 'rid') : 4
('came', 'along', 'home') : 4
('epaminondas', 'ai', 'got') : 4
('ai', 'got', 'sense') : 4
```

In [ ]:
```
#Data Analysis 5: averaged perceptron
```

In [188…
```
nltk.download('averaged_perceptron_tagger')
tagged_token = nltk.pos_tag(Book1_withoutstopLemma)
Book1_adjs = [w for w,p in tagged_token if p=='JJ']
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\JYOTHIKA\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

In [189…
```
tagged_token1 = nltk.pos_tag(Book2_withoutstopLemma)
Book2_adjs = [v for v,q in tagged_token1 if q=='JJ']
```

In [190…
```
fadjdist = FreqDist(Book1_adjs)
Book1_top50adj = fadjdist.most_common(50)

for k in Book1_top50adj:
    print (k) # top 50 adjectives for true data set
```

```
('harriet', 417)
('little', 326)
('much', 315)
('good', 303)
('great', 263)
('miss', 254)
('emma', 220)
('young', 192)
('dear', 153)
('sure', 150)
('many', 138)
('poor', 136)
('last', 121)
('u', 121)
('happy', 120)
('first', 104)
('wish', 92)
('possible', 84)
('old', 83)
('give', 82)
('present', 76)
('subject', 73)
('able', 71)
('whole', 71)
('short', 67)
('true', 64)
('general', 61)
('ready', 61)
('enough', 58)
('know', 57)
('superior', 57)
('bad', 56)
('equal', 55)
('right', 55)
('frank', 55)
('natural', 51)
('usual', 50)
('next', 50)
('weston', 50)
('long', 48)
('oh', 48)
('particular', 48)
('afraid', 47)
('agreeable', 47)
('mean', 47)
('full', 46)
('come', 45)
('strong', 44)
('real', 42)
('different', 41)
```

In [191…
```
fadjdist1 = FreqDist(Book2_adjs)
Book2_top50adj = fadjdist1.most_common(50)

for k in Book2_top50adj:
    print (k) # top 50 adjectives for fake data set
```

```
('little', 578)
('great', 110)
('old', 102)
('good', 76)
('big', 69)
('last', 55)
('beautiful', 43)
('white', 42)
('poor', 41)
('red', 41)
('many', 41)
('nightingale', 37)
('tree', 31)
('tiny', 29)
('small', 28)
('next', 28)
('u', 28)
('gingerbread', 27)
('hard', 27)
('fir', 26)
('long', 24)
('dear', 24)
('strong', 24)
('much', 24)
('happy', 24)
('new', 23)
('jackal', 21)
('open', 20)
('green', 19)
('sweet', 18)
('give', 17)
('whole', 17)
('terrible', 17)
('know', 17)
('wonderful', 17)
('fine', 16)
('fox', 16)
('oh', 16)
('saw', 16)
('full', 15)
('first', 15)
('light', 15)
('servant', 15)
('right', 15)
('young', 15)
('lazy', 15)
('soft', 14)
('dead', 14)
('bad', 14)
('brahmin', 14)
```

In [ ]:

In [ ]:
```
#A)
#Since tokenization is an essential stage in text processing,
#I decided to use it to separate the text into distinct words.
#To guarantee consistency in word frequencies and to standardize
#the text, lowercasing was used.
#In order to eliminate popular terms that usually have little meaning,
#stopwords were eliminated.
#Words with similar meanings can be grouped together by using
#lemmatization, which reduces inflected words to their basic form.
```

In [ ]:
```
#B)
# Proper nouns or domain-specific terms that might not be,
#pertinent to the study could be one potential issue with the word lists.
#Common bigrams, which appear frequently but may,
#not necessarily have important meaning, may be found in the bigram lists.
#We could remove stopwords from bigrams or use other,
#steps to find significant collocations in order to enhance the bigram lists.
```

In [ ]:
```
#C)
#Regardless of how mutually informative their word pairs are,
#the top 50 bigrams by frequency show the most frequently occurring pairs
#of neighboring words in the text.
#Conversely, the top 50 bigrams based on Mutual Information
#show word pairings that co-occur more frequently than one would
#anticipate by chance, indicating a greater relationship between the terms.
```

In [ ]:
```
#d)
#We may add domain-specific stopwords or exclude frequent words that,
#are unrelated to our analysis if we make changes to the stopwords list.
#To weed out less instructive bigrams, we might also try varying the
#frequency criteria for bigrams or use other collocation techniques.
```

In [ ]:
```
#e)
#Trigrams can provide further light on the relationships and context of words.
#For a more thorough understanding of the text,
#we can run top trigram lists using techniques similar to those
#used for bigrams and incorporate them into the analysis.
```

In [ ]:
```

```

In [ ]:
```
#3)

#A)
#I would like to compare the language and writing style of "austen-emma.txt"
#(from Jane Austen's "Emma") with "bryant-stories.txt"
#(a compilation of William Cullen Bryant's short stories).
#I want to specifically look into the differences
#between these two texts' word choices and the frequency of particular bigrams.
```

In [ ]:
```
#B)

#Word Distributions:

#Words pertaining to relationships, social interactions, and manners
#are likely to be used frequently in "austen-emma.txt,"
#which is not surprising given Austen's emphasis on
#love themes and conventional conventions.
#In "bryant-stories.txt," we may find a broader language pertaining to the
#natural world, rural living, and landscapes,
#considering Bryant's standing as a nature writer and poet.

#Bigram Rates:

#Studying the frequent bigrams in "austen-emma.txt,"
#we could come across word pairs like "Mr. Knightley," "Miss Woodhouse,"
#or "social engagements," which correspond to the social contexts and
#interpersonal interactions that the book describes.
#By contrast, many bigrams in "bryant-stories.txt" may
#consist of word combinations such as "old oak," "forest trees," or
#"crystal stream," demonstrating Bryant's preference for environmental
#imagery and descriptive language.
```

```
#Bigram Exchange Data:

#Strong correlations between terms like "love" and "marriage,"
#"manners" and "society," or "romantic" and "entanglements" may be
#shown by Mutual Information scores for bigrams in "austen-emma.txt,"
#suggesting recurrent themes and motifs in Austen's writing.
#Relevant word pairs, like "silent woods," "rustling leaves," or
#"rippling brook," may be highlighted by Mutual Information scores for
#"bryant-stories.txt," highlighting Bryant's poetic depictions of the natural worla
```