DATA.STAT.840 Statistical Methods for Text Data Analysis

Exercises 3

Daniel Kusnetsoff

In [1]:

```
import requests
import bs4
import re
import numpy as np
```

In [2]:

```
#%% Get the text content of the page
def getpagetext(parsedpage):
    # Remove HTML elements that are scripts
    scriptelements=parsedpage.find_all('script')
    # Concatenate the text content from all table cells
    for scriptelement in scriptelements:
        # Extract this script element from the page.
        # This changes the page given to this function!
        scriptelement.extract()
    pagetext=parsedpage.get_text()
    return(pagetext)
```

In [3]:

```
def top_gutenberg_ebooks(seed_page_url, top_k):
    ebook_titles=[]
    amount_books_downloaded=0
    texts_downloaded=[]
    ebook_addresses=[]
    url_to_download= "https://www.gutenberg.org/files/"

    seed_page_parsed = requests.get(seed_page_url)
    h2_tag=bs4.BeautifulSoup(seed_page_parsed.content, 'html.parser').find(id='books-last30
    ol_tag = h2_tag.next_sibling.next_sibling
    for a_tag in ol_tag.find_all('a'):
        # find matching pattern for ebook name
        name_match = re.match(r'(.*)(\(\d+\))', a_tag.text)
        ebook_name = name_match.group(1).strip()
        # find matching pattern for ebook id
        id_match = re.match(r'/ebooks/(\d+)', a_tag.get('href'))
        ebook_id = id_match.group(1)
        ebook_url = url_to_download + ebook_id + '/' + ebook_id + '-0.txt'
        # checking book is not alreydy downloaded
        if (ebook_url not in ebook_addresses) & (amount_books_downloaded < top_k):
            print('Downloading text file from:')
            print(ebook_url)
            ebook_page = requests.get(ebook_url)
            parsed_page = bs4.BeautifulSoup(ebook_page.content, 'html.parser')
            # get text from the ebook
            ebook_text = getpagetext(parsed_page)
            start_index = ebook_text.find('*** START OF THE PROJECT GUTENBERG EBOOK')
            end_index = ebook_text.find('*** END OF THE PROJECT GUTENBERG EBOOK')
            ebook_text = ebook_text[start_index:end_index]
            # remove leading and trailing whitespaces
            ebook_text = ebook_text.strip()
            ebook_text = ' '.join(ebook_text.split())
            # store book content
            texts_downloaded.append(ebook_text)
            ebook_addresses.append(ebook_url)
            ebook_titles.append(ebook_name)
            amount_books_downloaded += 1

    return(texts_downloaded, ebook_titles, ebook_addresses, amount_books_downloaded)
```

In [4]:

```
texts_downloaded, ebook_titles, ebook_addresses, amount_books_downloaded =top_gutenberg_ebo
```

```
Downloading text file from:
https://www.gutenberg.org/files/2641/2641-0.txt (https://www.gutenberg.org/f
iles/2641/2641-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/145/145-0.txt (https://www.gutenberg.org/fil
es/145/145-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/37106/37106-0.txt (https://www.gutenberg.or
g/files/37106/37106-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/16389/16389-0.txt (https://www.gutenberg.or
g/files/16389/16389-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/67979/67979-0.txt (https://www.gutenberg.or
g/files/67979/67979-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/394/394-0.txt (https://www.gutenberg.org/fil
es/394/394-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/1342/1342-0.txt (https://www.gutenberg.org/f
iles/1342/1342-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/84/84-0.txt (https://www.gutenberg.org/file
s/84/84-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/2701/2701-0.txt (https://www.gutenberg.org/f
iles/2701/2701-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/345/345-0.txt (https://www.gutenberg.org/fil
es/345/345-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/11/11-0.txt (https://www.gutenberg.org/file
s/11/11-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/1661/1661-0.txt (https://www.gutenberg.org/f
iles/1661/1661-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/1952/1952-0.txt (https://www.gutenberg.org/f
iles/1952/1952-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/25344/25344-0.txt (https://www.gutenberg.or
g/files/25344/25344-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/100/100-0.txt (https://www.gutenberg.org/fil
es/100/100-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/174/174-0.txt (https://www.gutenberg.org/fil
es/174/174-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/98/98-0.txt (https://www.gutenberg.org/file
s/98/98-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/1080/1080-0.txt (https://www.gutenberg.org/f
iles/1080/1080-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/6761/6761-0.txt (https://www.gutenberg.org/f
```

```
iles/6761/6761-0.txt)
Downloading text file from:
https://www.gutenberg.org/files/2160/2160-0.txt (https://www.gutenberg.org/f
iles/2160/2160-0.txt)
```

In [5]:

```
ebook_titles, ebook_addresses, amount_books_downloaded
```

Out[5]:

```
(['A Room with a View by E. M.  Forster',
  'Middlemarch by George Eliot',
  'Little Women; Or, Meg, Jo, Beth, and Amy by Louisa May Alcott',
  'The Enchanted April by Elizabeth Von Arnim',
  'The Blue Castle: a novel by L. M.  Montgomery',
  'Cranford by Elizabeth Cleghorn Gaskell',
  'Pride and Prejudice by Jane Austen',
  'Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley',
  'Moby Dick; Or, The Whale by Herman Melville',
  'Dracula by Bram Stoker',
  "Alice's Adventures in Wonderland by Lewis Carroll",
  'The Adventures of Sherlock Holmes by Arthur Conan Doyle',
  'The Yellow Wallpaper by Charlotte Perkins Gilman',
  'The Scarlet Letter by Nathaniel Hawthorne',
  'The Complete Works of William Shakespeare by William Shakespeare',
  'The Picture of Dorian Gray by Oscar Wilde',
  'A Tale of Two Cities by Charles Dickens',
  'A Modest Proposal by Jonathan Swift',
  'The Adventures of Ferdinand Count Fathom — Complete by T.  Smollett',
  'The Expedition of Humphry Clinker by T.  Smollett'],
 ['https://www.gutenberg.org/files/2641/2641-0.txt',
  'https://www.gutenberg.org/files/145/145-0.txt',
  'https://www.gutenberg.org/files/37106/37106-0.txt',
  'https://www.gutenberg.org/files/16389/16389-0.txt',
  'https://www.gutenberg.org/files/67979/67979-0.txt',
  'https://www.gutenberg.org/files/394/394-0.txt',
  'https://www.gutenberg.org/files/1342/1342-0.txt',
  'https://www.gutenberg.org/files/84/84-0.txt',
  'https://www.gutenberg.org/files/2701/2701-0.txt',
  'https://www.gutenberg.org/files/345/345-0.txt',
  'https://www.gutenberg.org/files/11/11-0.txt',
  'https://www.gutenberg.org/files/1661/1661-0.txt',
  'https://www.gutenberg.org/files/1952/1952-0.txt',
  'https://www.gutenberg.org/files/25344/25344-0.txt',
  'https://www.gutenberg.org/files/100/100-0.txt',
  'https://www.gutenberg.org/files/174/174-0.txt',
  'https://www.gutenberg.org/files/98/98-0.txt',
  'https://www.gutenberg.org/files/1080/1080-0.txt',
  'https://www.gutenberg.org/files/6761/6761-0.txt',
  'https://www.gutenberg.org/files/2160/2160-0.txt'],
 20)
```

In [6]:

```
#%% Tokenize loaded texts and change them to NLTK format
import nltk

mycrawled_nltktexts=[]
for k in range(len(texts_downloaded)):
    temp_tokenizedtext=nltk.word_tokenize(texts_downloaded[k])
    temp_nltktext=nltk.Text(temp_tokenizedtext)
    mycrawled_nltktexts.append(temp_nltktext)
```

In [7]:

```
mycrawled_nltktexts[19]
```

Out[7]:

```
<Text: ...>
```

In [8]:

```
mycrawled_lowercasetexts = []
for k in range(len(mycrawled_nltktexts)):
    temp_lowercasetext = []
    for l in range(len(mycrawled_nltktexts[k])):
        lowercaseword = mycrawled_nltktexts[k][l].lower()
        temp_lowercasetext.append(lowercaseword)
    temp_lowercasetest = nltk.Text(temp_lowercasetext)
    mycrawled_lowercasetexts.append(temp_lowercasetext)
```

In [9]:

```
def tagtowordnet(postag):
    wordnettag=-1
    if postag[0]=='N':
        wordnettag='n'
    elif postag[0]=='V':
        wordnettag='v'
    elif postag[0]=='J':
        wordnettag='a'
    elif postag[0]=='R':
        wordnettag='r'
    return(wordnettag)
```

In [10]:

```
# Download wordnet resource if you do not have it already
nltk.download('wordnet')
# Download tagger resource if you do not have it already
nltk.download('averaged_perceptron_tagger')

lemmatizer=nltk.stem.WordNetLemmatizer()
```

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

In [11]:

```
def lemmatizetext(nltktexttolemmatize):
    # Tag the text with POS tags
    taggedtext=nltk.pos_tag(nltktexttolemmatize)
    # Lemmatize each word text
    lemmatizedtext=[]
    for l in range(len(taggedtext)):
        # Lemmatize a word using the WordNet converted POS tag
        wordtolemmatize=taggedtext[l][0]
        wordnettag=tagtowordnet(taggedtext[l][1])
        if wordnettag!=-1:
            lemmatizedword=lemmatizer.lemmatize(wordtolemmatize,wordnettag)
        else:
            lemmatizedword=wordtolemmatize
        # Store the lemmatized word
        lemmatizedtext.append(lemmatizedword)
    return(lemmatizedtext)

mycrawled_lemmatizedtexts=[]
for k in range(len(mycrawled_lowercasetexts)):
    lemmatizedtext=lemmatizetext(mycrawled_lowercasetexts[k])
    lemmatizedtext=nltk.Text(lemmatizedtext)
    mycrawled_lemmatizedtexts.append(lemmatizedtext)
```

In [12]:

```
import numpy as np
myvocabularies=[]
myindices_in_vocabularies=[]
# Find the vocabulary of each document
for k in range(len(mycrawled_lemmatizedtexts)):
    # Get unique words and where they occur
    temptext=mycrawled_lemmatizedtexts[k]
    uniqueresults=np.unique(temptext,return_inverse=True)
    uniquewords=uniqueresults[0]
    wordindices=uniqueresults[1]
    # Store the vocabulary and indices of document words in it
    myvocabularies.append(uniquewords)
    myindices_in_vocabularies.append(wordindices)
myvocabularies[0]
```

Out[12]:

```
array(['!', '(', ')', ..., '’', '“', '”'], dtype='<U22')
```

In [13]:

```
tempvocabulary=[]
for k in range(len(mycrawled_lemmatizedtexts)):
    tempvocabulary.extend(myvocabularies[k])

# Find the unique elements among all vocabularies
uniqueresults=np.unique(tempvocabulary,return_inverse=True)
unifiedvocabulary=uniqueresults[0]
wordindices=uniqueresults[1]
```

In [14]:

```
# Translate previous indices to the unified vocabulary.

vocabularystart=0
myindices_in_unifiedvocabulary=[]
for k in range(len(mycrawled_lemmatizedtexts)):
    # In order to shift word indices, we must temporarily
    # change their data type to a Numpy array
    tempindices=np.array(myindices_in_vocabularies[k])
    tempindices=tempindices+vocabularystart
    tempindices=wordindices[tempindices]
    myindices_in_unifiedvocabulary.append(tempindices)
    vocabularystart=vocabularystart+len(myvocabularies[k])
```

In [15]:

```
unifiedvocabulary_totaloccurrencecounts=np.zeros((len(unifiedvocabulary),1))
unifiedvocabulary_documentcounts=np.zeros((len(unifiedvocabulary),1))
unifiedvocabulary_meancounts=np.zeros((len(unifiedvocabulary),1))
unifiedvocabulary_countvariances=np.zeros((len(unifiedvocabulary),1))
```

In [16]:

```
# First pass: count occurrences
for k in range(len(mycrawled_lemmatizedtexts)):
    print(k)
    occurrencecounts=np.zeros((len(unifiedvocabulary),1))
    for l in range(len(myindices_in_unifiedvocabulary[k])):
        occurrencecounts[myindices_in_unifiedvocabulary[k][l]]= \
            occurrencecounts[myindices_in_unifiedvocabulary[k][l]]+1
    unifiedvocabulary_totaloccurrencecounts= \
        unifiedvocabulary_totaloccurrencecounts+occurrencecounts
    unifiedvocabulary_documentcounts= \
        unifiedvocabulary_documentcounts+(occurrencecounts>0)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [17]:

```
# Mean occurrence counts over documents
unifiedvocabulary_meancounts= \
    unifiedvocabulary_totaloccurrencecounts/len(mycrawled_lemmatizedtexts)
```

In [19]:

```
# Second pass to count variances
for k in range(len(mycrawled_lemmatizedtexts)):
    print(k)
    occurrencecounts=np.zeros((len(unifiedvocabulary),1))
    for l in range(len(myindices_in_unifiedvocabulary[k])):
        occurrencecounts[myindices_in_unifiedvocabulary[k][l]]= \
            occurrencecounts[myindices_in_unifiedvocabulary[k][l]]+1
    unifiedvocabulary_countvariances=unifiedvocabulary_countvariances+ \
        (occurrencecounts-unifiedvocabulary_meancounts)**2
unifiedvocabulary_countvariances= \
    unifiedvocabulary_countvariances/(len(mycrawled_lemmatizedtexts)-1)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [20]:

```
#%% Inspect frequent words
# Sort words by largest total (or mean) occurrence count
highest_totaloccurrences_indices=np.argsort(\
    -1*unifiedvocabulary_totaloccurrencecounts,axis=0)
print(np.squeeze(unifiedvocabulary[\
    highest_totaloccurrences_indices[1:100]]))
print(np.squeeze(\
    unifiedvocabulary_totaloccurrencecounts[\
    highest_totaloccurrences_indices[1:100]]))
```

```
['.' 'the' 'be' 'and' 'to' 'of' 'i' 'a' 'in' 'have' 'that' '’' ';' 'you'
 'it' 'he' 'his' 'not' 'with' '“' 'my' '”' 'for' 'her' 'as' '?' 'but'
 'she' '!' 'me' 'do' 'this' 'him' 'at' 's' 'on' 'say' 'so' 'all' 'by'
 'what' 'your' 'which' 'will' 'no' 'if' 'from' 'we' 'come' 'would' 'there'
 'they' 'go' 'one' ':' 'when' 'make' 'or' 'know' 'now' 'an' 'thou' 'more'
 'good' 'them' 'then' 'their' 'see' 'shall' 'who' 'like' 'out' 'can' 'our'
 'think' 'man' 'look' 'could' 'some' 'than' 'up' 'well' 'take' 'd' "'s"
 'how' 'thy' 'mr.' 'very' "'d" 'should' 'give' 'time' 'must' 'here' 'lord'
 'upon' 'into' 'sir']
[147065. 109843.  92069.  78476.  65040.  63207.  52477.  49821.  38925.
  36317.  35301.  31748.  31470.  29285.  29181.  27465.  23945.  22741.
  22358.  22348.  21718.  21289.  20521.  20150.  19830.  18896.  18055.
  16900.  16332.  15394.  15374.  13951.  13831.  13280.  13137.  12649.
  12412.  12400.  11748.  10694.  10391.  10369.   9593.   9416.   9247.
   9121.   9041.   8833.   8370.   8350.   8321.   8315.   7959.   7620.
   7378.   7370.   7130.   6969.   6814.   6504.   6492.   6407.   6146.
   6122.   5983.   5816.   5772.   5747.   5550.   5521.   5331.   5300.
   5109.   5103.   5098.   5086.   4919.   4918.   4915.   4849.   4833.
   4816.   4808.   4730.   4670.   4567.   4565.   4504.   4438.   4410.
   4278.   4264.   4252.   4244.   4233.   4202.   4055.   4034.   3928.]
```

In [21]:

```
# Sort words by largest total document count
highest_documentoccurrences_indices=np.argsort(\
    -1*unifiedvocabulary_documentcounts,axis=0)
print(np.squeeze(unifiedvocabulary[\
    highest_documentoccurrences_indices[1:100]]))
print(np.squeeze(\
    unifiedvocabulary_documentcounts[\
    highest_documentoccurrences_indices[1:100]]))
```

```
['proper' 'a' 'put' 'question' 'ready' 'real' 'reason' 'repeat' 'rest'
 'round' 'child' 'same' 'project' 'say' 'seem' 'seldom' 'several' 'shall'
 'she' 'should' 'since' 'certain' 'so' 'some' 'something' 'soon' 'see'
 'able' 'possibly' 'about' 'again' 'after' 'of' 'off' 'often' 'old' 'on'
 'one' 'or' 'order' 'other' 'others' 'our' 'out' 'own' 'common' 'add'
 'people' 'act' 'perhaps' 'come' 'person' 'piece' 'account' 'please'
 'pleased' 'poor' 'sound' 'now' 'stand' 'stay' 'till' 'by' 'time' 'to'
 'too' 'but' 'trouble' 'turn' 'two' 'under' 'up' 'upon' 'through' '.'
 'use' 'bring' ',' '***' 'very' 'walk' 'want' 'way' 'we' 'well' 'what'
 'when' 'us' 'three' 'those' 'this' 'steal' 'stir' 'strength' 'such'
 'summer' 'care' '?' ';']
[17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17.
 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17.
 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17.
 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17.
 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17. 17.
 17. 17. 17. 17. 17. 17. 17. 17. 17.]
```

In [22]:

```
# Sort by largest variance of count over documents
highest_countvariances_indices=np.argsort(\
    -1*unifiedvocabulary_countvariances,axis=0)
print(np.squeeze(unifiedvocabulary[\
    highest_countvariances_indices[1:100]]))
print(np.squeeze(\
    unifiedvocabulary_countvariances[\
    highest_countvariances_indices[1:100]]))
```

```
['.' 'the' 'be' 'and' 'i' 'to' 'of' '’' ';' 'a' 'you' 'my' 'in' 'that' '?'
 'not' 'have' 'with' '!' 'it' 'he' 'for' 'me' 'his' 'this' 'your' 'but'
 'do' 'as' 'thou' '“' '”' 'her' 'him' 'so' 'will' 's' 'what' 'd' ':' 'thy'
 "'d" 'all' 'she' 'by' 'no' 'we' 'shall' 'if' 'come' 'on' 'lord' "'s"
 'say' ']' '[' 'good' 'thee' 'king' 'our' 'make' 'at' 'o' 'sir' 'which'
 'from' 'now' 'go' 'they' 'love' 'would' 'or' 'more' 'let' 'there' 'know'
 'then' 'here' 'well' 'their' 'enter' 'when' 'how' 'can' 'one' 'man'
 'give' 'them' 'hath' 'mr.' 'an' 'like' '``' 'than' '--' "'" 'see' 'upon'
 'us']
[3.32562821e+08 5.18423892e+07 4.63649997e+07 4.03818997e+07
 2.79569457e+07 2.35751284e+07 1.96035854e+07 1.48161824e+07
 1.45613922e+07 1.40028851e+07 1.06789191e+07 8.76503323e+06
 8.23200298e+06 7.92677734e+06 6.16240417e+06 4.95080803e+06
 4.85116480e+06 3.72885905e+06 3.60667473e+06 3.52609579e+06
 3.47089229e+06 3.44034687e+06 3.41025043e+06 3.39194414e+06
 2.54568659e+06 2.52365978e+06 2.35781805e+06 2.34274394e+06
 2.12181424e+06 1.80109288e+06 1.79826586e+06 1.70809446e+06
 1.60978399e+06 1.60206598e+06 1.46553717e+06 1.46404162e+06
 1.46152302e+06 1.29290288e+06 1.05987773e+06 1.02328220e+06
 9.84498380e+05 9.83193850e+05 9.41548410e+05 8.78938615e+05
 8.66842670e+05 8.23411776e+05 8.18789172e+05 7.55788089e+05
 7.33742324e+05 7.21917175e+05 7.04782213e+05 6.95467690e+05
 6.33895900e+05 6.27029518e+05 6.13325861e+05 6.12966679e+05
 5.99984698e+05 5.97992454e+05 5.54831036e+05 5.49647343e+05
 5.23241496e+05 5.16753019e+05 4.71901773e+05 4.71546194e+05
 4.70967122e+05 4.64080939e+05 4.60907324e+05 3.82078834e+05
 3.77748573e+05 3.74384310e+05 3.64966704e+05 3.52448141e+05
 3.29043889e+05 3.23639956e+05 3.17390745e+05 3.11313141e+05
 3.08994083e+05 3.07563022e+05 2.97475191e+05 2.87115169e+05
 2.76697560e+05 2.69799280e+05 2.60918701e+05 2.35919055e+05
 2.29774626e+05 2.25285551e+05 2.23008488e+05 2.20557593e+05
 2.20517061e+05 2.15847601e+05 2.14449241e+05 2.09889028e+05
 2.08277607e+05 2.08204042e+05 2.07656443e+05 2.05466357e+05
 2.02217427e+05 1.87858158e+05 1.67042759e+05]
```

Ex 2.4 Pruning

In [23]:

```
nltk.download('stopwords')

#%% Vocabulary pruning
nltkstopwords=nltk.corpus.stopwords.words('english')
pruningdecisions=np.zeros((len(unifiedvocabulary),1))
for k in range(len(unifiedvocabulary)):
    # Rule 1: check the nltk stop word list
    if (unifiedvocabulary[k] in nltkstopwords):
        pruningdecisions[k]=1
    # Rule 2: if the word is in the top 1% of frequent words
    if (k in highest_totaloccurrences_indices[\
        0:int(np.floor(len(unifiedvocabulary)*0.01))]):
        pruningdecisions[k]=1
   # Rule 3: if the word occurs less than 4 times
    if(unifiedvocabulary_totaloccurrencecounts[k] < 4):
        pruningdecisions[k] = 1
    # Rule 4: if the word is too short
    if len(unifiedvocabulary[k])<2:
        pruningdecisions[k]=1
    # Rule 5: if the word is too long
    if len(unifiedvocabulary[k])>20:
        pruningdecisions[k]=1
    # Rule 6: if the word has unwanted characters
    # (here for simplicity only a-z allowed)
    if unifiedvocabulary[k].isalpha()==False:
        pruningdecisions[k]=1
```

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

In [24]:

```
#%% Get indices of documents to remaining words
oldtopruned=[]
tempind=-1
for k in range(len(unifiedvocabulary)):
    if pruningdecisions[k]==0:
        tempind=tempind+1
        oldtopruned.append(tempind)
    else:
        oldtopruned.append(-1)
```

In [25]:

```
#%% Create pruned texts
mycrawled_prunedtexts=[]
myindices_in_prunedvocabulary=[]
for k in range(len(mycrawled_lemmatizedtexts)):
    print(k)
    temp_newindices=[]
    temp_newdoc=[]
    for l in range(len(mycrawled_lemmatizedtexts[k])):
        temp_oldindex=myindices_in_unifiedvocabulary[k][l]
        temp_newindex=oldtopruned[temp_oldindex]
        if temp_newindex!=-1:
            temp_newindices.append(temp_newindex)
            temp_newdoc.append(unifiedvocabulary[temp_oldindex])
    mycrawled_prunedtexts.append(temp_newdoc)
    myindices_in_prunedvocabulary.append(temp_newindices)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [26]:

```
#%% Inspect remaining frequent words
# Sort remaining words by largest total (or mean) occurrence count
remainingindices=np.squeeze(np.where(pruningdecisions==0)[0])
remainingvocabulary=unifiedvocabulary[remainingindices]
remainingvocabulary_totaloccurrencecounts= \
    unifiedvocabulary_totaloccurrencecounts[remainingindices]
remaining_highest_totaloccurrences_indices= \
    np.argsort(-1*remainingvocabulary_totaloccurrencecounts,axis=0)
print(np.squeeze(remainingvocabulary[remaining_highest_totaloccurrences_indices[1:100]]))
print(np.squeeze(remainingvocabulary_totaloccurrencecounts[ \
remaining_highest_totaloccurrences_indices[1:100]]))
```

```
['fit' 'rome' 'promise' 'drop' 'pain' 'free' 'four' 'note' 'red' 'glad'
 'view' 'consider' 'hence' 'fell' 'trust' 'holmes' 'ill' 'drive' 'duty'
 'chance' 'catch' 'marriage' 'possible' 'line' 'devil' 'neither' 'clear'
 'alice' 'past' 'comfort' 'crown' 'offer' 'mouth' 'common' 'count' 'start'
 'wall' 'sorrow' 'afraid' 'ring' 'pardon' 'mad' 'object' 'opinion'
 'fisher' 'attend' 'warwick' 'wild' 'worth' 'company' 'effect' 'breath'
 'ah' 'eat' 'forward' 'able' 'messenger' 'seat' 'enemy' 'quickly' 'ten'
 'blow' 'except' 'law' 'evening' 'hester' 'hat' 'five' 'darcy' 'choose'
 'secret' 'farewell' 'dorian' 'visit' 'third' 'false' 'grief' 'regard'
 'wit' 'spend' 'stone' 'sake' 'brooke' 'service' 'fault' 'hundred' 'er'
 'gold' 'command' 'celia' 'edward' 'james' 'interest' 'behold' 'dress'
 'exit' 'content' 'matty' 'observe']
[465. 465. 465. 464. 463. 463. 463. 463. 463. 461. 461. 460. 460. 460.
 460. 457. 454. 453. 453. 452. 449. 447. 447. 445. 445. 443. 442. 441.
 440. 439. 438. 436. 436. 435. 435. 435. 435. 434. 433. 432. 428. 428.
 425. 425. 424. 423. 423. 423. 423. 423. 422. 421. 421. 419. 419. 419.
 417. 417. 417. 416. 416. 416. 415. 414. 412. 411. 411. 411. 411. 410.
 409. 407. 407. 406. 406. 403. 401. 401. 400. 399. 398. 397. 397. 397.
 396. 396. 395. 395. 394. 393. 393. 393. 393. 393. 392. 392. 390. 390.
 387.]
```

Ex 3.1

In [27]:

```
import scipy

def download_specific_ebook(ebook_url):
    ebook_page = requests.get(ebook_url)
    parsed_page = bs4.BeautifulSoup(ebook_page.content, 'html.parser')
    ebook_text = getpagetext(parsed_page)
    start_text = '*** START OF THIS PROJECT GUTENBERG EBOOK THE CALL OF THE WILD ***'
    start_index = ebook_text.find(start_text)
    end_index = ebook_text.find('*** END OF THE PROJECT GUTENBERG EBOOK')
    ebook_text = ebook_text[start_index + len(start_text):end_index]

    # remove whitespaces
    ebook_text = ebook_text.strip()
    ebook_text = ' '.join(ebook_text.split())
    return(ebook_text)
```

In [28]:

```
# Download book
ebook_text = download_specific_ebook('https://www.gutenberg.org/files/1342/1342-0.txt')
```

In [29]:

```
# tokenize text
tokenized_text = nltk.word_tokenize(ebook_text)
# NLTK-format text
nltk_texts = nltk.Text(tokenized_text)
# lowercase the text
lowercase_texts = []
for l in range(len(nltk_texts)):
    lowercase_word = nltk_texts[l].lower()
    lowercase_texts.append(lowercase_word)
```

In [30]:

```
def tagtowordnet(postag):
    wordnettag=-1
    if postag[0]=='N':
        wordnettag='n'
    elif postag[0]=='V':
        wordnettag='v'
    elif postag[0]=='J':
        wordnettag='a'
    elif postag[0]=='R':
        wordnettag='r'
    return(wordnettag)
```

In [31]:

```
# Download wordnet resource if you do not have it already
nltk.download('wordnet')
# Download tagger resource if you do not have it already
nltk.download('averaged_perceptron_tagger')

lemmatizer=nltk.stem.WordNetLemmatizer()
```

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

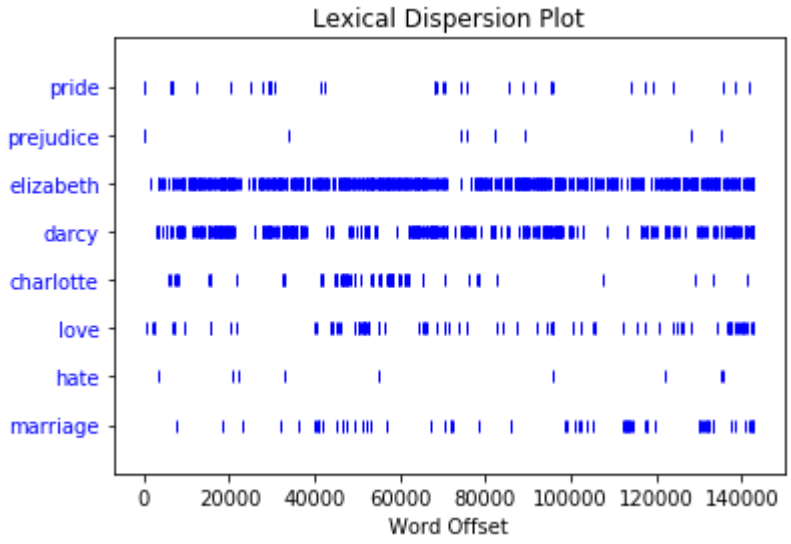
In [32]:

```
def lemmatizetext(nltktexttolemmatize):
    # Tag the text with POS tags
    taggedtext=nltk.pos_tag(nltktexttolemmatize)
    # Lemmatize each word text
    lemmatizedtext=[]
    for l in range(len(taggedtext)):
        # Lemmatize a word using the WordNet converted POS tag
        wordtolemmatize=taggedtext[l][0]
        wordnettag=tagtowordnet(taggedtext[l][1])
        if wordnettag!=-1:
            lemmatizedword=lemmatizer.lemmatize(wordtolemmatize,wordnettag)
        else:
            lemmatizedword=wordtolemmatize
        # Store the lemmatized word
        lemmatizedtext.append(lemmatizedword)
    return(lemmatizedtext)


# lemmatization of text
lemmatized_texts = lemmatizetext(lowercase_texts)
lemmatized_texts = nltk.Text(lemmatized_texts)
```

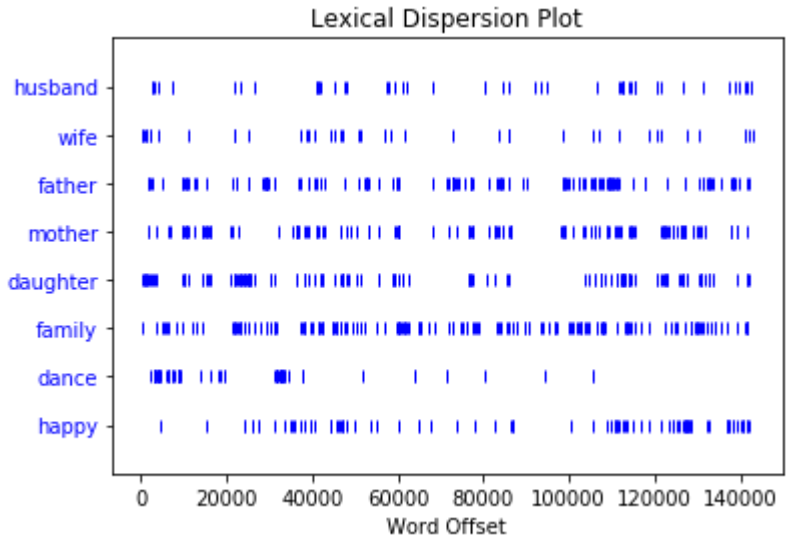
In [34]:

```
# lexical dispersion plots
lemmatized_texts.dispersion_plot(['pride', 'prejudice', 'elizabeth', 'darcy', 'charlotte',
```



In [35]:

```
lemmatized_texts.dispersion_plot(['husband', 'wife', 'father', 'mother', 'daughter', 'famil
```



Ex 3.1 C

The word elizabeth clearly appears most in the book, compared to other recurring words. This means that we can assume that Elizabeth is one of the main characters or the main charcter in the book. We can also see that the the use of the Elizabeth only pauses for shorter instances and is clearly critical for the happenings or plot of the book.

The word darcy appears to be the second most used word of these words. Darcy is also used throughout the story and appears in most of the same parts as elizabeth. We can assume, that both Darcy and Elizabeth have a connection between each other.

The third most used word is family. It is clearly used less than the two previously presented words. However it seems to be a continuous theme throughout the book.

From the plots we can make good assumptions on what the book is about. According to the plots Elizabeth and Darcy are two of the main characters of the book. As the book focuses on themes such as family, love and marriage, we may assume that there is some kind of non-platonic or even romantic connection between Elizabeth and Darcy. As the ending of the book is filled with the word darcy, elizabeth and specifically love, we can assume an ending to the book that is based on this love.

In [ ]:

In [ ]:

```
```

Ex 3.2

In [36]:

```
ebook_text = download_specific_ebook('https://www.gutenberg.org/files/84/84-0.txt')
```

In [37]:

```
# tokenize text
tokenized_text = nltk.word_tokenize(ebook_text)
# NLTK-format text
nltk_texts = nltk.Text(tokenized_text)
# lowercase the text
lowercase_texts = []
for l in range(len(nltk_texts)):
    lowercase_word = nltk_texts[l].lower()
    lowercase_texts.append(lowercase_word)
```

In [38]:

```
# lemmatize the text
lemmatized_texts = lemmatizetext(lowercase_texts)
# NLTK the text
lemmatized_texts = nltk.Text(lemmatized_texts)
```

In [39]:

```
lemmatized_texts.concordance('science',lines=80)
#Due to using NLTK 3.2 command only shows first 25 by default.
#Using concordance_list we can see there are actually 29 matches
#Also by counting the times words occur in the text.

science=lemmatized_texts
from collections import Counter
word_counts = Counter(science)

print(f'"science" appears {word_counts["science"]} time(s)')
```

```
Displaying 25 of 25 matches:
cine , and those branch of physical science from which a naval adventurer mi
ght
ch lead to my predilection for that science . when i be thirteen year of age
we
explode and that a modern system of science have be introduce which possess
muc
in the great disdain for a would-be science which could never even step with
in
e branch of study appertain to that science as be build upon secure foundati
on
t deeply imbue in the secret of his science . he ask me several question con
cer
progress in the different branch of science appertain to natural philosophy
. i
 by the modern professor of natural science . with a confusion of idea only
to
ry different when the master of the science seek immortality and power ; suc
h v
hose vision on which my interest in science be chiefly found . i be require
to
ry view of the present state of the science and explain many of its elementa
ry
get : “ the ancient teacher of this science , ” say he , “ promise impossibi
lit
ent study and to devote myself to a science for which i believe myself to po
sse
ave not neglect the other branch of science . a man would make but a very so
rry
r wish be to become really a man of science and not merely a petty experimen
tal
ould have advance far enough in the science not to derange their mechanism .
he
vate the acquaintance of the men of science of the university , and i find e
ven
m can conceive of the enticement of science . in other study you go as far a
s o
 death . i become acquaint with the science of anatomy , but this be not suf
fic
rect their inquiry towards the same science , that i alone should be reserve
to
ement which every day take place in science and mechanic , i be encourage to
ho
nishing progress i have make in the science . he soon perceive that i dislik
e t
ubject from my improvement , to the science itself , with a desire , as i ev
ide
 sympathise in my taste for natural science ; and his literary pursuit diffe
```

```
r w
 at that time i know nothing of the science of word or letter . “ the family
,
"science" appears 29 time(s)
```

In [40]:

```
lemmatized_texts.concordance_list('science', lines = 80)
```

```
heir', 'mechanism', '.', 'he', 'also', 'give', 'me', 'the', 'list', 'of',
'book', 'which', 'i', 'have', 'request'], offset=13859, left_print='ould h
ave advance far enough in the', right_print='not to derange their mechanis
m . he', line='ould have advance far enough in the science not to derange
their mechanism . he'),
 ConcordanceLine(left=['inquirer', 'have', 'write', 'on', 'these', 'subjec
t', '.', 'i', 'attend', 'the', 'lecture', 'and', 'cultivate', 'the', 'acqu
aintance', 'of', 'the', 'men', 'of'], query='science', right=['of', 'the',
'university', ',', 'and', 'i', 'find', 'even', 'in', 'm.', 'krempe', 'a',
'great', 'deal', 'of', 'sound', 'sense', 'and'], offset=13961, left_print
='vate the acquaintance of the men of', right_print='of the university , a
nd i find even', line='vate the acquaintance of the men of science of the
university , and i find even'),
 ConcordanceLine(left=['which', 'i', 'hop', 'to', 'make', '.', 'none', 'bu
t', 'those', 'who', 'have', 'experience', 'them', 'can', 'conceive', 'of',
'the', 'enticement', 'of'], query='science', right=['.', 'in', 'other', 's
tudy', 'you', 'go', 'as', 'far', 'as', 'others', 'have', 'go', 'before',
'you', ',', 'and', 'there', 'be'], offset=14221, left_print='m can conceiv
e of the enticement of', right_print='. in other study you go as far as
```

In [41]:

```
lemmatized_texts.concordance('horror', lines=100)
horror=lemmatized_texts
#Actually 49 matches according to the concordance_list and counting
word_counts = Counter(horror)

#lemmatized_texts.concordance('horror', lines=100)
print(f'"horror" appears {word_counts["horror"]} time(s)')
```

```
Displaying 25 of 25 matches:
ould be impress with no supernatural horror . i do not ever remember to ha
ve tre
ding-places . who shall conceive the horror of my secret toil as i dabble
among
of the dream vanish , and breathless horror and disgust fill my heart . un
able t
flannel . i start from my sleep with horror ; a cold dew cover my forehead
, my
e . oh ! no mortal could support the horror of that countenance . a mummy
again
 extreme weakness . mingle with this horror , i felt the bitterness of dis
appoin
his hand , and in a moment forgot my horror and misfortune ; i felt sudden
ly , a
 will and power to effect purpose of horror , such as the deed which he ha
ve now
e to announce publicly ; its astound horror would be look upon as madness
by the
t could make the murder memorable in horror . justine also be a girl of me
rit an
e place round his neck , a murmur of horror and indignation fill the court
. jus
 countenance have alter . surprise , horror , and misery be strongly expre
ss . s
d ignominy ? i could not sustain the horror of my situation , and when i p
erceiv
have before experienced sensation of horror , and i have endeavour to best
ow upo
, and then continue , “ i think with horror , my sweet lady , that you sho
uld be
phetic soul , as , tear by remorse , horror , and despair , i behold those
i lov
er ’ s health be deeply shake by the horror of the recent event . elizabet
h be s
 myself on the grass , weigh down by horror and despair . at length i arri
ve at
ave create . i tremble with rage and horror , resolve to wait his approach
and t
ntally be present at the trial ; his horror and indignation be uncontrolla
ble wh
e , in language which paint your own horror and render mine indelible . i
sicken
d turn them from me with disdain and horror . the poor that stop at their
door b
 of my person be the chief object of horror with those who have formerly b
ehold
gatha enter . who can describe their horror and consternation on behold me
? aga
fool in have expose my person to the horror of his child . i ought to have
```

```
famil
"horror" appears 49 time(s)
```

In [42]:

```
word_counts = Counter(horror)

#lemmatized_texts.concordance('horror', lines=100)
print(f'"horror" appears {word_counts["horror"]} time(s)')
```

```
"horror" appears 49 time(s)
```

In [43]:

```
lemmatized_texts.concordance('monster', lines=100)
monster=lemmatized_texts
#Actually 32 matches according to the concordance_list and counting
word_counts = Counter(monster)

print(f'"monster" appears {word_counts["monster"]} time(s)')
```

```
Displaying 25 of 25 matches:
, i behold the wretch—the miserable monster whom i have create . he hold up
the
walk about . i dread to behold this monster , but i fear still more that hen
ry
me ! save me ! ” i imagine that the monster seize me ; i struggle furiously
and
estore me to life . the form of the monster on whom i have bestow existence
be
almost begin to think that i be the monster that he say i be . he threaten e
xco
, and i live in daily fear lest the monster whom i have create should perpet
rat
come home , and men appear to me as monster thirst for each other ’ s blood
. y
d of your remain friends. ” “ abhor monster ! fiend that thou art ! the tort
ure
y convince that i be in reality the monster that i be , i be fill with the b
itt
d of none like me . be i , then , a monster , a blot upon the earth , from w
hic
accurse creator ! why do you form a monster so hideous that even _you_ turn
fro
lently . ‘ let me go , ’ he cry ; ‘ monster ! ugly wretch ! you wish to eat
me
ou must come with me. ’ “ ‘ hideous monster ! let me go . my papa be a syndi
c—h
ntent me . it be true , we shall be monster , cut off from all the world ; b
ut
t perform my engagement and let the monster depart with his mate before i al
low
happiness . my promise fulfil , the monster would depart for ever . or ( so
my
itude . i do not doubt but that the monster follow me and would discover him
sel
will only exasperate my rage. ” the monster saw my determination in my face
and
rval—all leave behind , on whom the monster might satisfy his sanguinary and
me
 at others i felt the finger of the monster already grasp my neck , and scre
am
it be the watery , cloud eye of the monster , as i first saw them in my cham
ber
 , have fall a victim to me and the monster of my creation . i repassed , in
my
ould die to make her happy . if the monster execute his threat , death be in
evi
 as if possess of magic power , the monster have blind me to his real intent
ion
rred . a grin be on the face of the monster ; he seem to jeer , as with his
fie
"monster" appears 32 time(s)
```

In [44]:

```
lemmatized_texts.concordance('fear', lines=100)
fear=lemmatized_texts
#Actually 66 matches according to the concordance_list and counting
word_counts = Counter(fear)

print(f'"fear" appears {word_counts["fear"]} time(s)')
```

```
Displaying 25 of 25 matches:
and they be sufficient to conquer all fear of danger or death and to induce
me t
 however , lay to until the morning , fear to encounter in the dark those la
rge
ore he be able to speak , and i often fear that his suffering have deprive h
im o
mong the tame scene of nature i might fear to encounter your unbelief , perh
aps
 at a tale of superstition or to have fear the apparition of a spirit . dark
ness
tion , listen attentively , catch and fear each sound as if it be to announc
e th
 if i seek to avoid the wretch whom i fear every turning of the street would
pre
my heart palpitate in the sickness of fear , and i hurry on with irregular s
tep
who , on a lonely road , doth walk in fear and dread , and , have once turn
roun
 dread to behold this monster , but i fear still more that henry should see
him
climb the hill or row on the lake . i fear that he will become an idle unles
s we
sent to my recollection , but which i fear the detail to another would only
impr
ly , might not be the less decisive . fear overcame me ; i dare no advance ,
dre
 , or to mock at my unhappiness ? ” i fear , my friend , that i shall render
mys
yet , as i draw near home , grief and fear again overcame me . night also cl
ose
he evidence of fact a weight that , i fear , leave no hope for doubt . but s
he w
 guiltless of this murder . i have no fear , therefore , that any circumstan
tial
” say i , “ and that shall be prove ; fear nothing , but let your spirit be
chee
ar , and they speak well of her ; but fear and hatred of the crime of which
they
but do not mourn , dear girl . do not fear . i will proclaim , i will prove
your
hake her head mournfully . “ i do not fear to die , ” she say ; “ that pang
be p
and deprive the soul both of hope and fear . justine die , she rest , and i
be a
nalterable evil , and i live in daily fear lest the monster whom i have crea
te s
 the past . there be always scope for fear so long as anything i love remain
beh
 mighty as omnipotence—and i cease to fear or to bend before any be less alm
ight
"fear" appears 66 time(s)
```

The amounts for the words are: science: 29 horror: 49 monster: 32 fear: 66 The word fear is clearly the most popular of the four words followed by the word horror.

As we take a closer look at the words we can see, that The word science is usually preceded by the words the, of and in.

The word horror is usualy preceded by the words with, of and the.

The word monster is usually preceded by the word the and sometimes the word a. There are often exlamation marks near the word monster.

The most popular word fear is often preceded by the word I.

EX 3.3

In [45]:

```
ebook1_text = download_specific_ebook('https://www.gutenberg.org/files/215/215-0.txt')
```

In [46]:

```
ebook1_text
```

```
almost literally torn to pieces, the swart half-breed standing over her an
d cursing horribly. The scene often came back to Buck to trouble him in hi
s sleep. So that was the way. No fair play. Once down, that was the end of
you. Well, he would see to it that he never went down. Spitz ran out his t
ongue and laughed again, and from that moment Buck hated him with a bitter
and deathless hatred. Before he had recovered from the shock caused by the
tragic passing of Curly, he received another shock. François fastened upon
him an arrangement of straps and buckles. It was a harness, such as he had
seen the grooms put on the horses at home. And as he had seen horses work,
so he was set to work, hauling François on a sled to the forest that fring
ed the valley, and returning with a load of firewood. Though his dignity w
as sorely hurt by thus being made a draught animal, he was too wise to reb
el. He buckled down with a will and did his best, though it was all new an
d strange. François was stern, demanding instant obedience, and by virtue
of his whip receiving instant obedience; while Dave, who was an experience
d wheeler, nipped Buck’s hind quarters whenever he was in error. Spitz was
the leader, likewise experienced, and while he could not always get at Buc
k, he growled sharp reproof now and again, or cunningly threw his weight i
n the traces to jerk Buck into the way he should go. Buck learned easily,
and under the combined tuition of his two mates and François made remarkab
```

In [47]:

```
# tokenize text
tokenized_text = nltk.word_tokenize(ebook1_text)
# NLTK-format text
nltk_texts = nltk.Text(tokenized_text)
# lowercase the text
lowercase_texts = []
for l in range(len(nltk_texts)):
    lowercase_word = nltk_texts[l].lower()
    lowercase_texts.append(lowercase_word)
```

In [48]:

```
# Lemmatize the text
lemmatized_texts = lemmatizetext(lowercase_texts)
# NLTK the text
lemmatized_texts = nltk.Text(lemmatized_texts)

print (lemmatized_texts[0],lemmatized_texts[1], lemmatized_texts[2])
```

```
produce by ryan
```

In [49]:

```
# get unique words, where they occur and counts of occurrence
new_unique_results = np.unique(lemmatized_texts, return_inverse = True, return_counts = Tru
new_myvocabulary = new_unique_results[0]
new_myindices_in_vocabulary = new_unique_results[1]
new_myvocabulary_occurrence_counts = new_unique_results[2]
new_highest_occurrences_indices = np.argsort(-1 * new_myvocabulary_occurrence_counts, axis=
```

In [50]:

```
new_highest_occurrences_indices
```

Out[50]:

```
array([  13, 3728,   16, ..., 1975, 2228, 2220], dtype=int64)
```

In [51]:

```
nltk.download('stopwords')

#%% Vocabulary pruning
nltkstopwords=nltk.corpus.stopwords.words('english')
new_pruningdecisions=np.zeros((len(new_myvocabulary),1))
for k in range(len(new_myvocabulary)):
    # Rule 1: check the nltk stop word list
    if (new_myvocabulary[k] in nltkstopwords):
        new_pruningdecisions[k]=1
    # Rule 2: if the word is in the top 1% of frequent words
    #if (k in new_highest_occurrences_indices[\
     #   0:int(np.floor(len(new_myvocabulary)*0.01))]):
      #  new_pruningdecisions[k]=1
   # Rule 3: if the word occurs less than 4 times
    if(new_myvocabulary_occurrence_counts[k] < 4):
        new_pruningdecisions[k] = 1
    # Rule 4: if the word is too short
    if len(new_myvocabulary[k])<2:
        new_pruningdecisions[k]=1
    # Rule 5: if the word is too long
    if len(new_myvocabulary[k])>20:
        new_pruningdecisions[k]=1
    # Rule 6: if the word has unwanted characters
    # (here for simplicity only a-z allowed)
    if new_myvocabulary[k].isalpha()==False:
        new_pruningdecisions[k]=1
```

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

In [52]:

```
print('Top-500 words after pruning the new unified vocabulary:\n')
remaining_indices = np.squeeze(np.where(new_pruningdecisions==0)[0])
remaining_vocabulary = new_myvocabulary[remaining_indices]
remainingvocabulary_occurrencecounts = new_myvocabulary_occurrence_counts[remaining_indices
remaining_highest_occurrences_indices = np.argsort(-1*remainingvocabulary_occurrencecounts)
print(remaining_vocabulary[remaining_highest_occurrences_indices[0:500]])
print(remainingvocabulary_occurrencecounts[remaining_highest_occurrences_indices[0:500]])
```

```
Top-500 words after pruning the new unified vocabulary:

['buck' 'dog' 'day' 'one' 'work' 'come' 'thornton' 'go' 'man' 'make'
 'time' 'back' 'project' 'could' 'upon' 'men' 'would' 'life' 'know'
 'spitz' 'sled' 'françois' 'great' 'thing' 'head' 'never' 'foot' 'like'
 'long' 'two' 'camp' 'get' 'though' 'take' 'night' 'way' 'run' 'eye'
 'last' 'club' 'away' 'trail' 'break' 'saw' 'hand' 'call' 'john' 'till'
 'hundred' 'perrault' 'three' 'half' 'hal' 'find' 'side' 'say' 'first'
 'place' 'trace' 'face' 'snow' 'wild' 'wolf' 'fire' 'see' 'every' 'many'
 'team' 'cry' 'stand' 'spring' 'end' 'old' 'leave' 'turn' 'give' 'body'
 'husky' 'teeth' 'seem' 'another' 'state' 'keep' 'sound' 'water' 'ice'
 'gutenberg' 'behind' 'start' 'hold' 'rest' 'law' 'leap' 'electronic'
 'become' 'around' 'fell' 'even' 'mile' 'lay' 'forth' 'good' 'dave' 'love'
 'without' 'drive' 'forest' 'hour' 'grow' 'foundation' 'look' 'travel'
 'fight' 'eat' 'follow' 'copy' 'shoulder' 'pull' 'nothing' 'still' 'pack'
 'throat' 'watch' 'well' 'fall' 'morning' 'term' 'kill' 'whip' 'bank'
 'mercedes' 'rush' 'sleep' 'leg' 'strength' 'use' 'receive' 'load' 'rope'
 'circle' 'close' 'mate' 'far' 'dead' 'nose' 'part' 'air' 'neck' 'fear'
 'less' 'stream' 'copyright' 'matter' 'drop' 'harness' 'pride' 'put' 'yet'
 'hard' 'along' 'full' 'charles' 'river' 'dawson' 'straight' 'right'
 'strike' 'draw' 'drag' 'bring' 'return' 'may' 'tree' 'driver' 'agreement'
 'judge' 'toil' 'learn' 'white' 'open' 'struggle' 'pain' 'must' 'throw'
 'distribute' 'snarl' 'pass' 'live' 'han' 'hear' 'muscle' 'bull'
 'donation' 'carry' 'pete' 'shake' 'pound' 'new' 'among' 'remain' 'little'
 'save' 'hair' 'master' 'whole' 'rise' 'alone' 'across' 'howl' 'include'
 'world' 'moment' 'land' 'much' 'movement' 'show' 'provide' 'license'
 'tent' 'cut' 'thousand' 'valley' 'charge' 'effort' 'strange' 'outside'
 'ground' 'brother' 'together' 'next' 'tell' 'death' 'bad' 'heavy' 'dat'
 'chapter' 'move' 'lie' 'cold' 'sometimes' 'snap' 'sit' 'play' 'living'
 'growl' 'archive' 'red' 'limp' 'slowly' 'united' 'ahead' 'fang' 'billee'
 'set' 'often' 'later' 'felt' 'toward' 'laugh' 'literary' 'blood' 'tongue'
 'blow' 'arm' 'pike' 'manner' 'bone' 'pay' 'four' 'paragraph' 'anything'
 'cover' 'heart' 'others' 'country' 'catch' 'fast' 'mark' 'fee' 'heel'
 'curly' 'stagger' 'fore' 'sweater' 'joe' 'jerk' 'forward' 'large' 'also'
 'rage' 'point' 'strong' 'fly' 'yelp' 'soft' 'second' 'light' 'young'
 'rapid' 'poor' 'reach' 'animal' 'wood' 'pitch' 'terrible' 'always' 'fact'
 'front' 'try' 'nature' 'sight' 'steal' 'heem' 'stop' 'instant' 'access'
 'lead' 'cunning' 'trademark' 'attack' 'distance' 'big' 'let' 'wide'
 'fashion' 'appear' 'yeehats' 'meat' 'protect' 'best' 'lash' 'warm' 'lake'
 'money' 'neither' 'meet' 'advance' 'breath' 'breed' 'agree' 'timber'
 'change' 'frost' 'beast' 'might' 'knee' 'longer' 'quarrel' 'trouble'
 'suddenly' 'condition' 'sir' 'refund' 'lose' 'matthewson' 'low' 'flee'
 'creature' 'comply' 'mad' 'swing' 'main' 'speak' 'section' 'ear' 'song'
 'die' 'seek' 'command' 'weight' 'delight' 'help' 'stretch' 'attempt'
 'hide' 'blind' 'dozen' 'walk' 'several' 'ever' 'eh' 'boat' 'information'
 'crate' 'whine' 'begin' 'fair' 'express' 'experience' 'bristle' 'cross'
 'sprang' 'require' 'nest' 'yard' 'want' 'mouth' 'read' 'within' 'winter'
 'food' 'reason' 'flash' 'trip' 'check' 'year' 'surprise' 'five' 'free'
 'beyond' 'alive' 'six' 'wind' 'reply' 'listen' 'hairy' 'forty' 'lick'
 'need' 'step' 'darkness' 'small' 'silent' 'southland' 'near' 'sharp'
 'since' 'dream' 'knock' 'speech' 'write' 'bite' 'train' 'sink' 'late'
```

```
 'ask' 'manage' 'quick' 'easy' 'cease' 'remember' 'chance' 'almost' 'fish'
 'chest' 'plunge' 'pool' 'cause' 'force' 'ration' 'general' 'proceed'
 'gain' 'coat' 'lip' 'talk' 'rabbit' 'tear' 'stir' 'sun' 'bar' 'manuel'
 'curse' 'understand' 'obtain' 'inch' 'sack' 'word' 'fifty' 'opportunity'
 'quarter' 'creek' 'crawl' 'voice' 'rock' 'past' 'fierce' 'think' 'fail'
 'runner' 'mean' 'rip' 'mother' 'slash' 'softly' 'something' 'week' 'weak'
 'strain' 'street' 'skeet' 'summer' 'sweep' 'tail' 'tax' 'volunteer'
 'tire' 'twice' 'trick' 'sure']
[359 169 121 115 107 103 102 101  95  95  92  91  86  81  81  73  69  68
  65  65  65  60  59  58  58  57  56  56  56  55  55  54  54  53  53  51
  46  45  44  43  43  42  42  41  41  40  40  39  39  39  38  38  37  37
  37  36  35  35  35  35  34  34  33  33  33  33  32  32  32  32  32  32
  32  31  31  30  30  30  30  30  30  29  29  29  28  28  28  28  27  27
  27  27  27  27  26  26  26  26  26  26  26  26  25  25  25  25  25  25
  25  25  25  25  24  24  24  24  24  23  23  23  23  22  22  22  22  21
  21  21  21  21  21  21  21  21  21  21  21  21  20  20  20  20  20  20
  20  20  20  19  19  19  19  19  19  19  19  19  19  19  19  19  19  18
  18  18  18  18  18  18  18  18  18  18  18  18  18  18  18  17  17  17
  17  17  17  17  17  17  17  17  17  17  16  16  16  16  16  16  16  16
  16  16  16  16  16  16  15  15  15  15  15  15  15  15  15  15  15  15
  15  15  15  15  15  15  15  14  14  14  14  14  14  14  14  14  14  14
  14  14  14  14  14  14  14  14  13  13  13  13  13  13  13  13  13  13
  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13
  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12
  12  12  12  12  12  12  12  12  12  11  11  11  11  11  11  11  11  11
  11  11  11  11  11  11  11  11  11  11  11  11  11  11  11  11  11  11
  11  11  11  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10
  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10
  10  10  10  10  10  10  10  10  10  10   9   9   9   9   9   9   9   9
   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9
   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9   9
   9   9   9   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
   8   8   8   8   8   8   8   8   8   8   8   8   7   7   7   7   7   7
   7   7   7   7   7   7   7   7   7   7   7   7   7   7]
```

In [53]:

```
#%% Get indices of documents to remaining words
new_oldtopruned = []
new_tempind = -1
for k in range(len(new_myvocabulary)):
    if new_pruningdecisions[k] == 0:
        new_tempind = new_tempind + 1
        new_oldtopruned.append(new_tempind)
    else:
        new_oldtopruned.append(-1)
```

In [54]:

```
#%% Create pruned texts
new_mypruned_texts = []
new_myindices_in_prunedvocabulary = []
for l in range(len(lemmatized_texts)):
    new_temp_oldindex = new_myindices_in_vocabulary[l]
    new_temp_newindex = new_oldtopruned[new_temp_oldindex]
    #print(temp_newindex)
    #print(temp_oldindex)
    if new_temp_newindex != -1:
        new_myindices_in_prunedvocabulary.append(new_temp_newindex)
        new_mypruned_texts.append(new_myvocabulary[temp_oldindex])
```

In [ ]:

```

```

In [55]:

```
import scipy
distanceoccurrences=scipy.sparse.lil_matrix(\
    (len(remainingvocabulary),len(remainingvocabulary)))
sumdistances=scipy.sparse.lil_matrix(\
    (len(remainingvocabulary),len(remainingvocabulary)))
sumabsdistances=scipy.sparse.lil_matrix(\
    (len(remainingvocabulary),len(remainingvocabulary)))
sumdistancesquares=scipy.sparse.lil_matrix(\
    (len(remainingvocabulary),len(remainingvocabulary)))
```

In [56]:

```
latestoccurrencepositions = scipy.sparse.lil_matrix((len(remaining_vocabulary), len(remaini
for m in range(len(new_mypruned_texts)):
    # Get the vocabulary index of the current word in position m
    currentword = new_myindices_in_prunedvocabulary[m]
    # Loop through previous words, counting back up to 10 words from current word
    windowsize = min(m, 10)
    for n in range(windowsize):
        # Get the vocabulary index of the previous word in position m-n-1
        previousword = new_myindices_in_prunedvocabulary[m-n-1]
        # Is this the fist time we have encountered this word while
        # counting back from the word at m? Then it is the closest pair
        if latestoccurrencepositions[currentword,previousword] < m:
            # Store the occurrence of this word pair with the word at m as the 1st word
            distanceoccurrences[currentword,previousword] = distanceoccurrences[currentwor

            sumdistances[currentword,previousword] = sumdistances[currentword,previousword]

            sumabsdistances[currentword,previousword] = sumabsdistances[currentword,previou

            sumdistancesquares[currentword,previousword] = sumdistancesquares[currentword,p

            # Store the occurrence of this word pair with the word at n as the 1st word
            distanceoccurrences[previousword,currentword] = distanceoccurrences[previouswor

            sumdistances[previousword,currentword] = sumdistances[previousword,currentword]

            sumabsdistances[previousword,currentword] = sumabsdistances[previousword,currer

            sumdistancesquares[previousword,currentword] = sumdistancesquares[previousword,
            # Mark that we found this pair while counting down from m,
            # so we do not count more distant occurrences of the pair
            latestoccurrencepositions[currentword,previousword] = m
            latestoccurrencepositions[previousword,currentword] = m
```

In [57]:

```
# Compute distribution statistics based on the counts
n_vocab=len(remainingvocabulary)
distancemeans=scipy.sparse.lil_matrix((n_vocab,n_vocab))
absdistancemeans=scipy.sparse.lil_matrix((n_vocab,n_vocab))
distancevariances=scipy.sparse.lil_matrix((n_vocab,n_vocab))
absdistancevariances=scipy.sparse.lil_matrix((n_vocab,n_vocab))
for m in range(n_vocab):
    print(m)
    # Find the column indices that have at least two occurrences
    tempindices=np.nonzero(distanceoccurrences[m,:]>1)[1]
    # The occurrence vector needs to be a non-sparse data type
    tempoccurrences=distanceoccurrences[m,tempindices].todense()
    # Estimate mean of m-n distance
    distancemeans[m,tempindices]=np.squeeze(\
    np.array(sumdistances[m,tempindices]/tempoccurrences))
    absdistancemeans[m,tempindices]=np.squeeze(\
    np.array(sumabsdistances[m,tempindices]/tempoccurrences))
        # Estimate variance of m-n distance
    meanterm=distancemeans[m,tempindices].todense()
    meanterm=np.multiply(meanterm,meanterm)
    meanterm=np.multiply(tempoccurrences/(tempoccurrences-1),meanterm)
    distancevariances[m,tempindices]=np.squeeze(\
    np.array(sumdistancesquares[m,tempindices]/(tempoccurrences-1) \
    - meanterm))
    meanterm=absdistancemeans[m,tempindices].todense()
    meanterm=np.multiply(meanterm,meanterm)
    meanterm=np.multiply(tempoccurrences/(tempoccurrences-1),meanterm)
    absdistancevariances[m,tempindices]=np.squeeze(\
    np.array(sumdistancesquares[m,tempindices]/(tempoccurrences-1) \
    - meanterm))
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [58]:

```
# Compute overall distance distribution
overalldistancecount = np.sum(distanceoccurrences)
overalldistancesum = np.sum(sumdistances)
overallabsdistancesum = np.sum(sumabsdistances)
overalldistancesquaresum = np.sum(sumdistancesquares)

overalldistancemean=overalldistancesum/overalldistancecount

overallabsdistancemean=overallabsdistancesum/overalldistancecount

overalldistancevariance = overalldistancesquaresum/(overalldistancecount-1)-overalldistance
    overalldistancecount-1)*overalldistancemean

overallabsdistancevariance = overalldistancesquaresum/(overalldistancecount-1)-overalldista
    overalldistancecount-1)*overallabsdistancemean
```

In [59]:

```
# find the index of a word
def findwordindex(wordstring):
    for k in range(len(remaining_vocabulary)):
        if remaining_vocabulary[k]==wordstring:
            return(k)
    return(-1)
```

In [ ]:

```

```

In [60]:

```
# Filter the collocation
# noun to noun collacations

def filter_collocation(text1, text2):
    acceptable_types = ('NN', 'NNS', 'NNP', 'NNPS')
    #second_type = ('NN', 'NNS', 'NNP', 'NNPS')
    collocation = nltk.word_tokenize(text1 + ' ' + text2)
    tags = nltk.pos_tag(collocation)
    if (tags[0][1] in acceptable_types) & (tags[1][1] in acceptable_types):
        #print(tags)
        return True
    else:
        return False
```

In [61]:

```
#%% Compute t-test p-values comparing abs distance distributions
absdistancepvalues = scipy.sparse.lil_matrix((n_vocab,n_vocab))

for m in range(n_vocab):
    # Find pairs of word m
    tempindices=np.nonzero(distanceoccurrences[m,:]>1)[1]
    # For computation we need to transform these to non-sparse vectors
    meanterm=absdistancemeans[m,tempindices].todense()
    varianceterm=absdistancevariances[m,tempindices].todense()
    occurrenceterm=distanceoccurrences[m,tempindices].todense()

    # Compute the t-test statistic for each pair
    tempstatistic=(meanterm - overallabsdistancemean)/np.sqrt(varianceterm/occurrenceterm +
                                                              overallabsdistancevariance/ov

    # Compute the t-test degrees of freedom for each pair
    tempdf=(np.power(varianceterm/occurrenceterm+overallabsdistancevariance/overalldistance
        np.power(varianceterm/occurrenceterm,2))/(occurrenceterm-1)+((
        overallabsdistancevariance/overalldistancecount)**2)/(overalldistancecount-1))

    # Compute the t-test p-value for each pair
    temppvalue=scipy.stats.t.cdf(tempstatistic,tempdf)
    # Store the t-test p-value for each pair
    absdistancepvalues[m,tempindices]=np.squeeze(np.array(temppvalue))
```

```
C:\Users\danie\Anaconda3\lib\site-packages\scipy\sparse\lil.py:512: FutureWa
rning: future versions will not create a writeable array from broadcast_arra
y. Set the writable flag explicitly to avoid this warning.
  if not i.flags.writeable or i.dtype not in (np.int32, np.int64):
C:\Users\danie\Anaconda3\lib\site-packages\scipy\sparse\lil.py:514: FutureWa
rning: future versions will not create a writeable array from broadcast_arra
y. Set the writable flag explicitly to avoid this warning.
  if not j.flags.writeable or j.dtype not in (np.int32, np.int64):
C:\Users\danie\Anaconda3\lib\site-packages\scipy\sparse\lil.py:518: FutureWa
rning: future versions will not create a writeable array from broadcast_arra
y. Set the writable flag explicitly to avoid this warning.
  if not x.flags.writeable:
```

In [62]:

```
def printtopcollocations(wordstring):
    # Find the chosen word and words that occurred with it at least 2 times
    mywordindex=findwordindex(wordstring)
    if mywordindex==-1:
        print('Word not found: '+ wordstring)
        return
    # Require at least 10 pair occurrences
    minpairoccurrences=10
    tempindices=np.nonzero(distanceoccurrences[mywordindex,:]>minpairoccurrences)[1]

    # Sort the pairs by lowest pvalue
    lowest_meandistances_indices=np.argsort(np.squeeze(np.array(
        absdistancepvalues[mywordindex,tempindices].todense())),axis=0)
    # Print the top-50 lowest-distance pairs
    print('\nLowest p-values\n')

    for k in range(min(50, len(lowest_meandistances_indices))):
        otherwordindex=tempindices[lowest_meandistances_indices[k]]

        filter_result = filter_collocation(remaining_vocabulary[mywordindex],remaining_voca
        if filter_result == True:
            print('{!s}--{!s}: {:d} occurrences, absdist: {:.1f} +- {:.1f}, offset: {:.1f}
              .format(remaining_vocabulary[mywordindex],remaining_vocabulary[otherwordindex
                      int(distanceoccurrences[mywordindex,otherwordindex]),
                      absdistancemeans[mywordindex,otherwordindex],
                      np.sqrt(absdistancevariances[mywordindex,otherwordindex]),
                      distancemeans[mywordindex,otherwordindex],
                      np.sqrt(distancevariances[mywordindex,otherwordindex]),
                      absdistancepvalues[mywordindex,otherwordindex]))
```

In [63]:

```
printtopcollocations('dog')
```

```
Lowest p-values

dog--husky: 21 occurrences, absdist: 3.7 +- 2.7, offset: 0.9 +- 4.6, pvalue:
0.004268
dog--men: 31 occurrences, absdist: 4.1 +- 3.0, offset: -1.0 +- 5.1, pvalue:
0.012825
dog--work: 20 occurrences, absdist: 4.6 +- 3.0, offset: 0.1 +- 5.6, pvalue:
0.120394
dog--life: 14 occurrences, absdist: 4.6 +- 2.8, offset: -3.0 +- 4.5, pvalue:
0.140312
dog--travel: 11 occurrences, absdist: 4.6 +- 2.5, offset: 0.5 +- 5.5, pvalu
e: 0.167618
dog--buck: 82 occurrences, absdist: 5.1 +- 2.8, offset: 0.9 +- 5.8, pvalue:
0.185716
dog--place: 12 occurrences, absdist: 4.7 +- 3.2, offset: -1.7 +- 5.5, pvalu
e: 0.216495
dog--see: 14 occurrences, absdist: 4.7 +- 3.6, offset: 0.6 +- 6.0, pvalue:
0.238298
dog--half: 15 occurrences, absdist: 4.9 +- 2.7, offset: -1.6 +- 5.6, pvalue:
0.254735
dog--break: 18 occurrences, absdist: 5.1 +- 2.5, offset: 0.3 +- 5.8, pvalue:
0.311012
dog--club: 11 occurrences, absdist: 5.0 +- 2.9, offset: 0.1 +- 6.0, pvalue:
0.325662
dog--team: 18 occurrences, absdist: 5.3 +- 2.6, offset: -0.1 +- 6.0, pvalue:
0.413456
dog--trail: 22 occurrences, absdist: 5.4 +- 2.6, offset: -1.7 +- 5.8, pvalu
e: 0.466091
dog--hal: 15 occurrences, absdist: 5.5 +- 2.4, offset: -1.4 +- 6.1, pvalue:
0.574850
dog--perrault: 16 occurrences, absdist: 5.6 +- 3.0, offset: 0.1 +- 6.5, pval
ue: 0.577943
dog--françois: 18 occurrences, absdist: 5.6 +- 2.2, offset: -1.2 +- 6.0, pva
lue: 0.608606
dog--turn: 12 occurrences, absdist: 5.8 +- 3.1, offset: -1.6 +- 6.5, pvalue:
0.644556
dog--camp: 19 occurrences, absdist: 5.7 +- 2.5, offset: 1.2 +- 6.3, pvalue:
0.708336
dog--day: 27 occurrences, absdist: 5.7 +- 2.7, offset: -0.6 +- 6.4, pvalue:
0.735041
dog--man: 21 occurrences, absdist: 6.0 +- 3.4, offset: -1.2 +- 6.9, pvalue:
0.778067
dog--way: 23 occurrences, absdist: 6.0 +- 2.8, offset: -0.5 +- 6.7, pvalue:
0.816422
dog--spitz: 14 occurrences, absdist: 5.9 +- 1.9, offset: -1.5 +- 6.3, pvalu
e: 0.831345
dog--thornton: 22 occurrences, absdist: 6.0 +- 2.7, offset: -0.8 +- 6.7, pva
lue: 0.839965
dog--john: 15 occurrences, absdist: 6.1 +- 2.4, offset: 1.0 +- 6.7, pvalue:
0.842407
dog--time: 13 occurrences, absdist: 6.4 +- 2.9, offset: -0.2 +- 7.2, pvalue:
0.877133

C:\Users\danie\Anaconda3\lib\site-packages\scipy\sparse\lil.py:512: Future
Warning: future versions will not create a writeable array from broadcast_
array. Set the writable flag explicitly to avoid this warning.
```

```
  if not i.flags.writeable or i.dtype not in (np.int32, np.int64):
C:\Users\danie\Anaconda3\lib\site-packages\scipy\sparse\lil.py:514: Future
Warning: future versions will not create a writeable array from broadcast_
array. Set the writable flag explicitly to avoid this warning.
  if not j.flags.writeable or j.dtype not in (np.int32, np.int64):
```

In [ ]:

```
```

In [ ]:

```
```

Ex 3.4

In [64]:

```
ebook_text_for_reg = download_specific_ebook('https://www.gutenberg.org/files/84/84-0.txt')
```

In [65]:

```
phrase_pattern = r'[Ff]or [a-zA-Z0-9]+ years'
phrase_pattern = re.compile(phrase_pattern)
phrase_pattern
```

Out[65]:

```
re.compile(r'[Ff]or [a-zA-Z0-9]+ years', re.UNICODE)
```

In [66]:

```
find_every_match = re.finditer(phrase_pattern, ebook_text_for_reg)
```

In [67]:

```
for temp_match in find_every_match:
    print(temp_match.group(), temp_match.span())
```

```
for many years (15748, 15762)
for many years (32401, 32415)
for several years (37326, 37343)
for many years (138856, 138870)
for many years (215413, 215427)
for many years (215954, 215968)
for several years (407238, 407255)
```

In [ ]:

```
```

Ex 3.5

In [ ]:

```
```

Length: Number of indices.

Metadata: Last date for accessability

Connectivity: Number of reactions by specific emojis, emoticons etc.

Popularity: Number of adding a song to a playlist

Sentiment: A feature that shows how many people that have read a comment, think the comment is well argumented.

Reception: would you recommend this to another person -> add types of person to recommend to