Lexical Diversity - Part II

(Gutenberg grade school primers)

MSDS 7337 - Natural Language Processing - Homework 02

Patrick McDevitt

16-Sep-2018

Lexical Diversity - Assignment

For this project we are requested to :

1. In Python, create a method for scoring the vocabulary size of a text, and normalize the score from 0 to 1. It does not matter what method you use for normalization as long as you explain it in a short paragraph. (Various methods will be discussed in the live session.)

2. After consulting section 3.2 in chapter 1 of Bird-Klein, create a method for scoring the long-word vocabulary size of a text, and likewise normalize (and explain) the scoring as in step 1 above.

3. Now create a "text difficulty score" by combining the lexical diversity score from homework 1, and your normalized score of vocabulary size and long-word vocabulary size, in equal weighting. Explain what you see when this score is applied to same graded texts you used in homework 1.

---

0 - Collect texts to create corpus

We were recommended to used the texts from url link: http://www.gutenberg.org/wiki/Children%27s_Instructional_Books_(Bookshelf)

The Bookshelf contains the following categories:

- Misc.
- Graded Readers
- Poetry Readers
- Readers for English Speakers Learning Other Languages
- Non-English Readers
- About Readers
- Science and Nature
- History
- Geography
- Uncategorized

In all, there are approx 104 titles identified at that collection of texts on the gutenberg.org website. To collect these texts to a local drive to facilitate the lexical processing, a python script is constructed that :

- parses the html from the parent site to extract 2 elements :
  - the hyperlink from the index page that points to a reference url for each text
  - the title of each corresponding text

- the hyperlink for each text actually refers to another web page within the gutenberg domain that includes links to each different format available for each text (html, pdf, txt, etc)
- there is a pattern evident from the 1st hyperlink that allows deconstruction and reconstruction of the url to correctly identify the url associated to the .txt version of the text
- that link construction is completed, and subsequently all of the texts are obtained using wget within the python script

The python code that identifies and downloads the 100+ texts from the Children's Bookshelf is included in Appendix A.

Subsequent to the download, an additional script is executed that removes the Gutenberg licensing texts from the beginning and ending portions of the text. Since the goal of this assignment is to assess the lexical diversity of each text by several measures, including word length, it is relevant to remove the Gutenberg license from the texts to eliminate that specific vocabulary from contributing to the diversity metrics. The python code that removes the licensing text is in Appendix B.

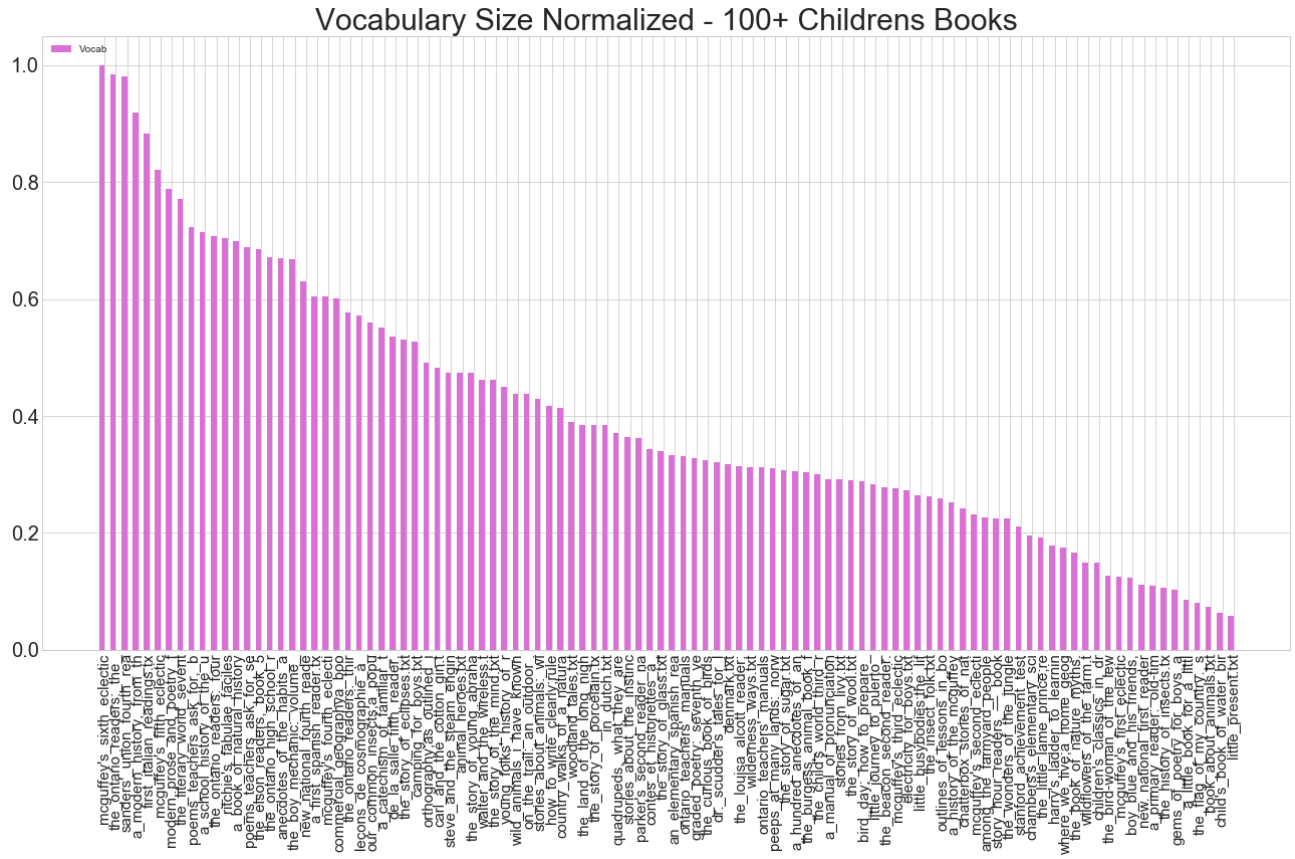
1 - Vocabulary Size

To establish a vocabulary size score for each text the following is completed :

- each of the 100+ texts is read into memory to establish the corpus
- some basic cleaning of the text to retain only alphabetic characters is completed
- all remaining words were converted to lower case letters
- the resulting number of unique vocabulary words were counted for each of the texts independently
- the maximum vocabulary size is found in the text :
  - *McGuffey's Sixth Eclectic Reader* with 14,611 unique words
- the vocabulary size for each text is then divided by the maximum vocabulary size (14,611) to produce the normalized vocabulary score for each text

The range and distribution of the scores for the collection of texts is shown in below Figure 1.

Figure 1. - Normalized vocabulary size scores - 100+ children's texts



The python code to establish the normalized vocabulary scores is included in Appendix C.

2 - Longest Words

A similar approach to the normalized vocabulary method is used to establish a text score based on longest word length.

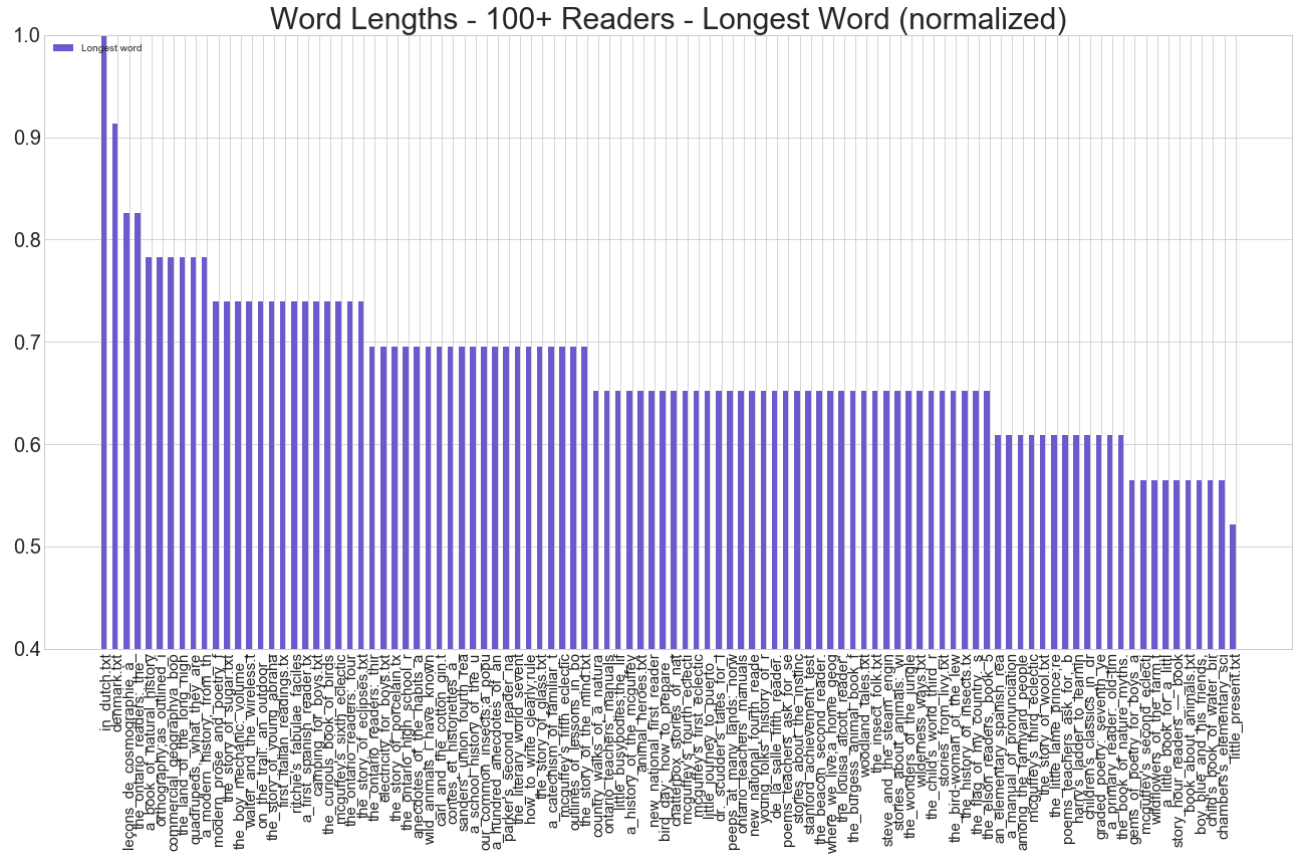
- each of the 100+ texts is read into memory to establish the corpus
- some basic cleaning of the text to retain only alphabetic characters is completed
- all remaining words were converted to lower case letters
- the resulting text is searched for all words with greater than seven characters
- this subset of words is sorted in descending order by word length
- the longest word in each text is identified, along with an integer count of the number of characters in that word
- the longest word identified among all of the texts surveyed is :
  - in the text *In Dutch* with the word *gehoorzaamheidsinstinct* which has 23 characters
- we observe that
  (a) this is not an English word, and also that
  (b) Google Translate® indicates that this is likely 2 words : *gehoorzaamheids* and *instinct*, but recognizing the German/Dutch language characteristic to concatenate words together to form new words, we choose to leave the word as it is, and
  (c) additionally, several of the texts with longer words are in non-English language texts (French, Spanish, and Latin are also among the higher word length texts). Since the English language texts are intermingled with the non-English language texts for the top 10 word lengths, (with 19 and 18 characters for the longest English words), we do not make adjustment; we retain the Dutch word

as the longest word for our measurement purposes and use that as the reference longest word for the current corpus

- the maximum word length for each text is then divided by the overall maximum word length (23) to produce the normalized longest word score for each text

The range and distribution of the scores for the collection of texts is shown in below Figure 2.

Figure 2. - Normalized longest words scores - 100+ children's texts



The python code to establish the normalized vocabulary scores is included in Appendix D.

3 - Text Difficulty Score

To provide a Text Difficulty Score (**TDS**) metric we introduce a metric based on the normalized values of lexical diversity, vocabulary size, and max word length. The lexical diversity were normalized 1 with this set of 100+ lexical diversity scores available from the same corpus as in sections 1 and 2 above. Then for a new metric, we use two different variants of combined normalized scores :

- simple addition of the three, and
- simple multiplication of the three.

For the eight selected texts for this evaluation, the normalized values for each measure along with the sum TDS and multiplication TDS are shown in below Table 1. The table rows are sorted in descending order based on value of the multiply TDS (**mlt_scores**).

Between these 2 candidate TDS values, we choose the multiplication version, due to slight improvement in order ranking of the lower level grade readers within the McGuffey Readers set.

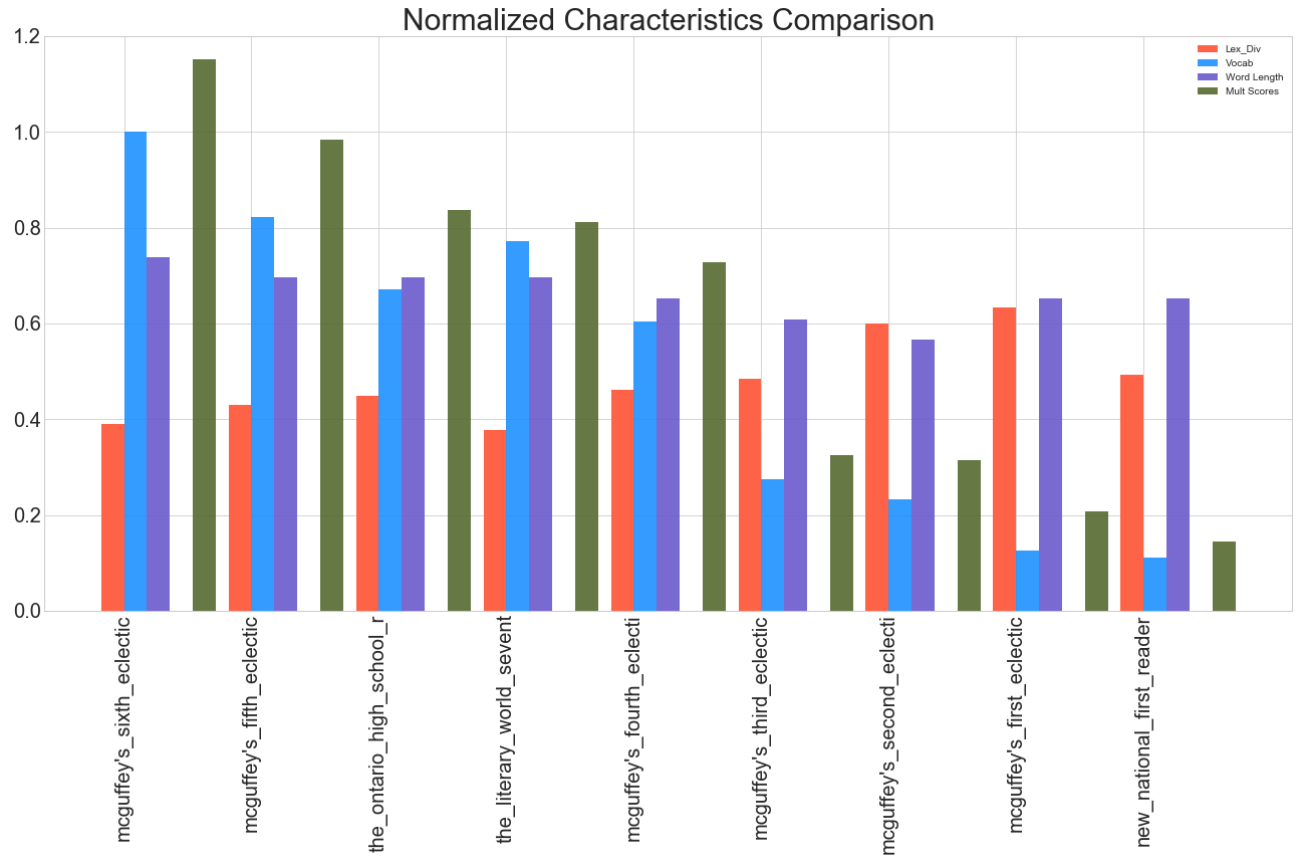
We can observe that there is, by this method, a monotonically decreasing trend in TDS associated to decreasing grade level among the McGuffey readers. Thus, this TDS score meets expectations at that level and is an improvement over the base lexical diversity score used in prior evaluation. Further, the New National First Reader is ranked lowest in TDS - again meeting expectations, since we observed previously that this reader has the smallest vocabulary among the texts and is also an early grade reader. The Literary World Seventh Reader and the Ontario High School Reader fall between the McGuffey 4th and 5th level readers - which is perhaps contrary to expectations, based on the titles of the readers. We would expect a high school reader to have higher complexity than a 5th or 6th level primer. This might be attributed to the eras in which these readers were compiled or an indication that there is further room for improvement in the TDS method. The metrics developed here do not, for instance, account for *stop words* or *stemming* prior to establishing the types within the corpus. These basic additional text preparation steps are likely to provide improvements in metrics to assess text difficulty.

The TDS value is also plotted in the below figure for each of the texts, along with the other statistics. The values are plotted in descending order of the TDS score. The TDS score is the green bar which is the furthest to the right in each grouping.

Table 1. - Normalized scores for vocab size, lexical diversity, word length, and combined

text_name	vocab	lex_div	word_length	sum_scores	mlt_scores
mcguffey's_sixth_eclectic_reader.txt	1.000	0.389	0.739	2.129	0.288
mcguffey's_fifth_eclectic_reader.txt	0.822	0.430	0.696	1.947	0.246
the_ontario_high_school_reader.txt	0.671	0.448	0.696	1.815	0.209
the_literary_world_seventh_reader.txt	0.771	0.378	0.696	1.845	0.203
mcguffey's_fourth_eclectic_reader.txt	0.604	0.462	0.652	1.718	0.182
mcguffey's_third_eclectic_reader.txt	0.276	0.485	0.609	1.370	0.081
mcguffey's_second_eclectic_reader.txt	0.232	0.600	0.565	1.397	0.079
mcguffey's_first_eclectic_reader,_	0.125	0.633	0.652	1.411	0.052
new_national_first_reader.txt	0.112	0.492	0.652	1.256	0.036

Figure 3. - Normalized scores for vocab size, lexical diversity, word length, and combined



The python code to provide the normalized combined metrics is in Appendix E.

References

[1] - http://www.nltk.org/book/ch01.html
[2] - http://www.gutenberg.org/wiki/Children%27s_Instructional_Books_(Bookshelf)

Appendix A - Download texts from gutenberg.org

```
# coding: utf-8

# In[149]:

#
# ... file : collect_texts_from_gutenberg.py
#
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ...
# ... msds 7337 NLP
# ... homework 02
# ... gutenberg - documment vocabulary normalization
# ... pmcdevitt@smu.edu
# ... 15-sep-2018
# ...
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... load packages
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import os
import numpy as np
import pandas as pd
import string
import strip

import nltk
from nltk.corpus import PlaintextCorpusReader

from lxml import html
import requests

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... some plotting defaults
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
from matplotlib import rcParams
rcParams.update({'figure.autolayout': True})
plt.rc('xtick', labelsize=20)
plt.rc('ytick', labelsize=20)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

```
# ... declare some directory locations
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

home_dir = "/home/mcdevitt/_ds/_smu/_src/nlp/homework_02/"
corpus_dir = "./text/"
plot_dir = "./plots/"

os.chdir(home_dir)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... extract urls for targeted texts
# ...
# ... if link is in this format : //www.gutenberg.org/ebooks/7841
# ... then texts is at this corresponding url : http://www.gutenberg.org/cache/epub/14880/pg14880.txt
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

page = requests.get('http://www.gutenberg.org/wiki/Children%27s_Instructional_Books_(Bookshelf)')
tree = html.fromstring(page.content)

links = tree.xpath('//a/@href')

book_link_mask = 'gutenberg.org/ebooks/'
book_links = [s for s in links if book_link_mask.lower() in s.lower()]
book_links = [lnk.replace('ebooks', 'cache/epub') for lnk in book_links]

# ... extract page number from end of url

pg_num = [lnk.split("epub/",1)[1] for lnk in book_links]
pg_num[0:5]

# ... recombine to create full url

text_link = ['http:' + lnk + '/pg' + pg + '.txt' for lnk,pg in zip(book_links, pg_num)]
text_link[0:5]

len(text_link)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... extract book titles
# ... tree.xpath('//div[@title="buyer-name"]/text()')
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

title = []

for ib in range(0, len(pg_num)) :
    path = '//a[@title="ebook:' + pg_num[ib] + '"]/text()'
    nxt_title = tree.xpath(path)[0]
    title.append(nxt_title)

title[0:5]
len(title)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

```
# ... clean up titles so filenames have standard chars and no spaces
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

''.join([x if x in string.printable else '' for x in title])

title = [x.replace('\n', '') for x in title]
title = [x.replace(' ', '_') for x in title]
title = [x.lower() for x in title]

title[0:10]

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... download texts to local directory
# ... capture errors to identify special case urls
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

import wget

print('Beginning file downloads ...')

os.chdir(home_dir)
os.chdir(corpus_dir)

error_indx = []
pipe = ' | '

for ib in range(0, len(pg_num)) :

    url = text_link[ib]
    file_name = title[ib] + '.txt'

    try :
        wget.download(url, file_name)

    except HTTPError as e:
        print ("HTTP error({0}): {1}".format(e.errno, e.strerror))
        error_indx.append(str(ib) + pipe + pg_num[ib] + pipe + title[ib])

        try :
            fix_url = text_link[ib].replace('.txt', '-0.txt')
            fix_url = fix.replace('/pg', '/')
            fix_url = fix.replace('/cache/epub/', '/files/')
            print(fix_url)
            wget.download(fix_url, file_name)

        except :
            print ('attempted fix_url did not resolve error')

print('... file downloads complete.')

error_indx

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

```
# ... end_of_file
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Appendix B - Remove Gutenberg licensing information

```
# coding: utf-8

# In[237]:

#
# ... file : remove_gutenberg_license.py
#
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ...
# ... msds 7337 NLP
# ... homework 02
# ... gutenberg - documment vocabulary normalization
# ... pmcdevitt@smu.edu
# ... 15-sep-2018
# ...
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... load packages
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import os
import re
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import PlaintextCorpusReader
from nltk.probability import ConditionalFreqDist

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... some directory and file name definitions
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

files = ".*\.txt"
home_dir = "/home/mcdevitt/_ds/_smu/_src/nlp/homework_02/"
corpus_root = "./text/"
corpus_clean = "./text_no_license/"
plot_dir = "./plots/"

os.chdir(home_dir)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... remove gutenberg license function
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

def remove_license(text_raw) :
```

```
    license_head1 = '*** START OF THIS PROJECT GUTENBERG EBOOK'
    license_head2 = '*** START OF THE PROJECT GUTENBERG EBOOK'
    license_head3 = '***START OF THE PROJECT GUTENBERG EBOOK'

    license_tail1 = 'End of the Project Gutenberg EBook'
    license_tail2 = 'End of Project Gutenberg'
    license_tail3 = '***END OF THE PROJECT GUTENBERG EBOOK'

    try :
        book_tail_text = text_raw.split(license_head1, 1)[1]
    except IndexError:
        try :
            book_tail_text = text_raw.split(license_head2, 1)[1]
        except IndexError:
            book_tail_text = text_raw.split(license_head3, 1)[1]

    text_without_license = book_tail_text.split(license_tail1, 1)[0]

    if(len(text_without_license) == len(book_tail_text)) :
        text_without_license = book_tail_text.split(license_tail2, 1)[0]

    if(len(text_without_license) == len(book_tail_text)) :
        text_without_license = book_tail_text.split(license_tail3, 1)[0]

    return text_without_license

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... read in raw downloaded texts / assemble corpus for evaluation
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

readers = PlaintextCorpusReader(corpus_root, files)

files = readers.fileids()
files[0:10]

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... remove license and write cleaned version to new directory
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

for fileid in readers.fileids():
    btxt = remove_license(readers.raw(fileid))

    text_file = open(corpus_clean + fileid, "w")
    text_file.write(btxt)
    text_file.close()

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... end_of_file
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Appendix C - Vocabulary size normalization

```
# coding: utf-8

# In[44]:

#
# ... file : vocabulary_size_normalization.py
#
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ...
# ... msds 7337 NLP
# ... homework 02
# ... gutenberg - documment vocabulary normalization
# ... pmcdevitt@smu.edu
# ... 15-sep-2018
# ...
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... load packages
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import os
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import PlaintextCorpusReader

get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
from matplotlib import rcParams
rcParams.update({'figure.autolayout': True})
plt.rc('xtick', labelsize=20)
plt.rc('ytick', labelsize=20)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... some directory and file name definitions
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

files = ".*\.txt"
home_dir = "/home/mcdevitt/_ds/_smu/_src/nlp/homework_02/"
corpus_root = "./text/"
plot_dir = "./plots/"

os.chdir(home_dir)
```

```
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... lexical diversity score - as given in nltk site
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

def lexical_diversity(my_text_data):
    tokens = len(my_text_data)
    types = len(set(my_text_data))
    diversity_score = types / tokens
    return diversity_score

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... read in texts / assemble corpus for evaluation
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

readers = PlaintextCorpusReader(corpus_root, files)

files = readers.fileids()
files[0:10]

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... create table to accumulate summary data
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_tbl = pd.DataFrame(columns =
    ['text_name',
     'num_chars',
     'num_words',
     'num_sents',
     'num_vocab',
     'tokens',
     'types',
     'lex_div'])

i_index = []
i_index = 0

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... loop thru each text to assemble metrics
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

print("Some basic statistics\n")

for fileid in readers.fileids():

    print(i_index, "---", fileid)

    num_chars = len(readers.raw(fileid))
    num_words = len(readers.words(fileid))
    num_sents = len(readers.sents(fileid))
    tokens = len(readers.words(fileid))
    types = len(set(readers.words(fileid)))

    num_vocab = len(set(w.lower() for w in readers.words(fileid)))
```

```
    rtxt = readers.words(fileid)
    ldiv = lexical_diversity(rtxt)

    table_data = {
     'text_name' : fileid,
     'num_chars' : num_chars,
     'num_words' : num_words,
     'num_sents' : num_sents,
     'num_vocab' : num_vocab,
     'tokens' : tokens,
     'types' : types,
     'lex_div' : ldiv
    }

    df_tbl = pd.DataFrame(table_data,
        columns = ['text_name',
             'num_chars',
             'num_words',
             'num_sents',
             'num_vocab',
             'tokens',
             'types',
             'lex_div'],
    index = [i_index + 1])
    i_index += 1
    results_tbl = results_tbl.append(df_tbl)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... basic statistics table complete
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_tbl = results_tbl.sort_values(results_tbl.columns[4], ascending = False)
print('Results_tbl - sorted by col num_vocab')
results_tbl

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... normalize each column by max column value
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_nrmlzd = results_tbl.copy()
results_nrmlzd = results_nrmlzd.sort_values(results_nrmlzd.columns[4], ascending = False)

df = results_nrmlzd.iloc[:, 1:8]
df_nrml = df / df.max()

df_labels = results_nrmlzd.iloc[:, 0]

results_nrmlzd = pd.concat([df_labels, df_nrml], axis = 1)

results_nrmlzd['vocab_ldiv'] = results_nrmlzd.apply(lambda x: x.lex_div * (x.num_words), axis=1)
results_nrmlzd['vocab_ldiv'] = results_nrmlzd['vocab_ldiv'] / results_nrmlzd['vocab_ldiv'].max()

print('results_nrmlzd - ')
```

```
results_nrmlzd = results_nrmlzd.sort_values(results_nrmlzd.columns[4], ascending = False)
print('results_nrmlzd - sorted by col num_vocab')
results_nrmlzd

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... add some shorter title names to fit within plotting space
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_tbl['title_short'] = [txt[:25] for txt in results_tbl['text_name']]
results_tbl['title_short'][:5]

results_nrmlzd['title_short'] = [txt[:25] for txt in results_nrmlzd['text_name']]
results_nrmlzd['title_short'][:5]

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... plot - vocabulary normalized size - sorted in descending order
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

N = len(results_nrmlzd)

ind = np.arange(N)
width = 0.5

_ = plt.figure(figsize = (18, 12))
offset = 0
_ = plt.bar(ind + offset, results_nrmlzd['num_vocab'], width, label='Vocab', color = 'orchid')

_ = plt.xticks(ind + width / 2, results_nrmlzd['title_short'], fontsize = '15')
_ = plt.xticks(rotation=90, )
_ = plt.legend(loc='upper left')
_ = plt.title('Vocabulary Size Normalized - 100+ Childrens Books', fontsize = '30')

axes = plt.gca()
_ = plt.savefig(plot_dir + 'childrens_books_normalized_vocab.png')
_ = plt.show()

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... write table to output file for future reference
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

os.chdir(home_dir)

file_name = "vocab_normalized.txt"

results_nrmlzd.to_csv(file_name,
                      header = True,
                      index = None,
                      sep=',',
                      mode = 'a')

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... end_of_file
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Appendix D - Max word length normalization

```
# coding: utf-8

# In[261]:

#
# ... file : longest_words.py
#
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ...
# ... msds 7337 NLP
# ... homework 02
# ... gutenberg - documment vocabulary normalization
# ... pmcdevitt@smu.edu
# ... 15-sep-2018
# ...
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... load packages
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import os
import re
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import PlaintextCorpusReader
from nltk.probability import ConditionalFreqDist

get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
from matplotlib import rcParams
rcParams.update({'figure.autolayout': True})
plt.rc('xtick', labelsize=20)
plt.rc('ytick', labelsize=20)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... some directory and file name definitions
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

files = ".*\.txt"
home_dir = "/home/mcdevitt/_ds/_smu/_src/nlp/homework_02/"
corpus_root = "./text/"
corpus_clean = "./text_no_license/"
plot_dir = "./plots/"
```

```
os.chdir(home_dir)


# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... long_words characterizations
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

def long_words(text_input) :
    text_unique = set(text_input)
    long_words = [w for w in text_unique if len(w) > 7]
    longest = sorted(long_words, key = len, reverse = True)[0]
    return longest


# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... remove all non-alpha characters
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

def clean_text(start_text) :
    text = [re.sub('^((?![a-zA-Z ]).)*$', '', x) for x in start_text]
    text = [re.sub('_', '', x) for x in text]
    text = [x.lower() for x in text]
    text = [x for x in text if x != '']
    return text


# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... read in text with gutenberg license removed / assemble corpus for evaluation
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

readers = PlaintextCorpusReader(corpus_clean, files)

files = readers.fileids()
files[0:10]

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... create table to accumulate summary data
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_tbl = pd.DataFrame(columns =
    ['text_name',
     'long_mot',
     'long_word_length'])

i_index = []
i_index = 0

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... loop thru each text to assemble metrics
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

print("Some basic statistics\n")
```

```python
for fileid in readers.fileids():
    rtxt = clean_text(readers.words(fileid))

    le_mot_le_plus_long = long_words(rtxt)
    print("\n", i_index, "--", fileid, " : ", le_mot_le_plus_long, "\n")

    table_data = {
     'text_name' : fileid,
     'long_mot' : le_mot_le_plus_long,
     'long_word_length' : len(le_mot_le_plus_long)
    }

    df_tbl = pd.DataFrame(table_data,
        columns = ['text_name',
             'long_mot', 'long_word_length'],
    index = [i_index + 1])
    i_index += 1
    results_tbl = results_tbl.append(df_tbl)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... basic statistics table complete
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_tbl = results_tbl.sort_values(results_tbl.columns[2], ascending = False)
results_tbl

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... normalize each column by max column value
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_nrmlzd = results_tbl.copy()
results_nrmlzd = results_nrmlzd.sort_values(results_nrmlzd.columns[2], ascending = False)

df = results_nrmlzd.iloc[:, 2]
df_nrml = df / df.max()

df_labels = results_nrmlzd.iloc[:, 0]
df_words =  results_nrmlzd.iloc[:, 1]

results_nrmlzd = pd.concat([df_labels, df_words, df_nrml], axis = 1)

print('results_nrmlzd - ')

results_nrmlzd = results_nrmlzd.sort_values(results_nrmlzd.columns[2], ascending = False)
print('results_nrmlzd - sorted by col num_vocab')
results_nrmlzd

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... add some shorter title names to fit within plotting space
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

results_tbl['title_short'] = [txt[:25] for txt in results_tbl['text_name']]
results_tbl['title_short'][:5]
```

```
results_nrmlzd['title_short'] = [txt[:25] for txt in results_nrmlzd['text_name']]
results_nrmlzd['title_short'][:5]

results_nrmlzd = results_nrmlzd.sort_values(results_nrmlzd.columns[2], ascending = False)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... plot - max word length normalized - sorted in descending order
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

N = len(results_nrmlzd)
ind = np.arange(N)
width = 0.5

_ = plt.figure(figsize = (18, 12))
offset = 0
_ = plt.bar(ind + offset, results_nrmlzd['long_word_length'],
            width,
            label='Longest word',
            color = 'slateblue')

_ = plt.xticks(ind + width / 2, results_nrmlzd['title_short'], fontsize = '15')
_ = plt.xticks(rotation=90)
_ = plt.legend(loc='upper left')
_ = plt.title('Word Lengths - 100+ Readers - Longest Word (normalized)', fontsize = '30')

_ = axes = plt.gca()
_ = axes.set_ylim([0.4, 1])

_ = plt.savefig(plot_dir + 'childrens_books_normalized_longword.png')
_ = plt.show()

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... write table to output file for future reference
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

os.chdir(home_dir)

file_name = "max_word_length_normalized.txt"

results_nrmlzd.to_csv(file_name,
                      header = True,
                      index = None,
                      sep=',',
                      mode = 'a')

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... end_of_file
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Appendix E - Combine normalized scores for overall ranking

```
# coding: utf-8

# In[34]:

#
# ... file : lexical_diversity_metrics.py
#
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ...
# ... msds 7337 NLP
# ... homework 02
# ... gutenberg - document vocabulary normalization
# ... pmcdevitt@smu.edu
# ... 15-sep-2018
# ...
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... load packages
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import os
import re
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import PlaintextCorpusReader
from nltk.probability import ConditionalFreqDist

get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
from matplotlib import rcParams
rcParams.update({'figure.autolayout': True})
plt.rc('xtick', labelsize=20)
plt.rc('ytick', labelsize=20)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... some directory and file name definitions
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

files = ".*\.txt"
home_dir = "/home/mcdevitt/_ds/_smu/_src/nlp/homework_02/"
corpus_root = "./text/"
corpus_clean = "./text_no_license/"
plot_dir = "./plots/"
```

```
os.chdir(home_dir)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... read in saved .txt files (.csv) from normalizations
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

file_wordl = 'max_word_length_normalized.csv'
file_vocab = 'vocab_normalized.csv'

selected_texts = [
    "mcguffey's_first_eclectic_reader,_revised_edition.txt",
    "mcguffey's_second_eclectic_reader.txt",
    "mcguffey's_third_eclectic_reader.txt",
    "mcguffey's_fourth_eclectic_reader.txt",
    "mcguffey's_fifth_eclectic_reader.txt",
    "mcguffey's_sixth_eclectic_reader.txt",
    "new_national_first_reader",
    "the_ontario_high_school_reader.txt",
    "the_literary_world_seventh_reader.txt"
]

df_wordl = pd.read_csv(file_wordl)
df_vocab = pd.read_csv(file_vocab)

df_vocab[:10]

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... condense to normalized metrics and selected texts
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

df_vocab_subset = df_vocab.loc[df_vocab['text_name'].isin(selected_texts)]
df_wordl_subset = df_wordl.loc[df_wordl['text_name'].isin(selected_texts)]

df_wordl_subset = df_wordl_subset[['text_name','long_word_length']]
df_vocab_subset = df_vocab_subset[['text_name', 'title_short', 'num_vocab', 'lex_div']]

df_wordl_subset = df_wordl_subset.sort_values(df_wordl_subset.columns[0])
df_vocab_subset = df_vocab_subset.sort_values(df_vocab_subset.columns[0])

df_wordl_subset = df_wordl_subset.reset_index(drop=True)
df_vocab_subset = df_vocab_subset.reset_index(drop=True)

df_wordl_subset = df_wordl_subset[['long_word_length']]

df_metrics = pd.concat([df_vocab_subset, df_wordl_subset], axis = 1)
df_metrics.columns = ['text_name', 'title_short', 'vocab_nrml', 'lex_div_nrml', 'word_length_nrml']

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... combine all 3 normalized metrics to total score
# ... - use addition (sum_scores) and multiplication (mlt_scores)
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

df_metrics['sum_scores'] = df_metrics['vocab_nrml'] + df_metrics['lex_div_nrml'] + df_metrics['word_leng
```

```
df_metrics['mlt_scores'] = df_metrics['vocab_nrml'] * df_metrics['lex_div_nrml'] * df_metrics['word_leng

df_metrics = df_metrics.sort_values(df_metrics.columns[5], ascending = False)
df_metrics
df_metrics = df_metrics.sort_values(df_metrics.columns[6], ascending = False)
df_metrics

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... plot comparison of all (normalized) metrics -
# ... sorted in descending mlt_scores order
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

N = len(df_metrics)

ind = np.arange(N)
width = 0.18

_ = plt.figure(figsize = (18, 12))

offset = width / 2
offset = 0
_ = plt.bar(ind + offset, df_metrics['lex_div_nrml'],
            width, label='Lex_Div', color = 'tomato')
_ = plt.bar(ind + offset + width, df_metrics['vocab_nrml'],
            width, label='Vocab', color = 'dodgerblue', alpha = 0.9)
_ = plt.bar(ind + offset + width*2, df_metrics['word_length_nrml'],
            width, label='Word Length', color = 'slateblue', alpha = 0.9)

#_ = plt.bar(ind + offset + width*3, df_metrics['sum_scores'], width, label='Sum Scores', color = 'orch

_ = plt.bar(ind + offset + width*4, df_metrics['mlt_scores']*4,
            width,
            label='Mult Scores',
            color = 'darkolivegreen',
            alpha = 0.9)

_ = plt.xticks(ind + width / 2, df_metrics['title_short'])
_ = plt.xticks(rotation=90)
_ = plt.legend(loc='upper right')
_ = plt.title('Normalized Characteristics Comparison', fontsize = '30')

axes = plt.gca()
axes.set_ylim([0, 1.2])

_ = plt.savefig(plot_dir + 'lex_div_normalized_scores.png')
_ = plt.show()

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... end_of_file
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```