Charles Dickens Corpus Preprocessing: LIB and CORPUS Tables

DS 5001: Exploratory Text Analytics

Cecily Wolfe (cew4pf)

Spring 2022

In [1]:
```
# read in docs

import os
from glob import glob
import numpy as np
import pandas as pd

from textparser import TextParser

import nltk
from nltk.stem.porter import PorterStemmer
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.lancaster import LancasterStemmer

from langmod import NgramCounter
from langmod import NgramLanguageModel
import itertools

import seaborn as sns
import plotly.express as px

from numpy.linalg import norm
from scipy.spatial.distance import pdist
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

from bow_tfidf_pca import create_bow, get_tfidf, get_pca
from prince import PCA

import requests
from bs4 import BeautifulSoup
import re
```

In [2]:
```
headers = {'user-agent': 'UVA example (cew4pf@virginia.edu)'}
r = requests.get("https://www.gutenberg.org/files/58157/58157-h/58157-h.htm", he
r
```

Out[2]:
```
<Response [200]>
```

In [3]:
```
index = BeautifulSoup(r.text, 'html')
```

In [4]:
```
sns.set()
```

In [5]:
```
OHCO = ["book_id", "chap_id", "para_num", "sent_num", "token_num"]
```

In [6]:
```
SENTS = OHCO[:4]
PARAS = OHCO[:3]
CHAPS = OHCO[:2]
BOOKS = OHCO[:1]
```

Preprocessing

Renaming Files

book_id	title
98-0	A Tale of Two Cities
pg564	The Mystery of Edwin Drood
580-0	The Pickwick Papers
588-0	Master Humphrey's Clock
644-0	The Haunted Man and the Ghost's Bargain
650-0	Pictures from Italy
653-0	The Chimes
675-0	American Notes
676-0	The Battle of Life
pg699	A Child's History of England
pg700	The Old Curiosity Shop
730-0	Oliver Twist
766-0	David Copperfield
786-0	Hard Times
807-0	Hunted Down
809-0	Holiday Romance
810-0	George Silverman's Explanation
pg821	Dombey and Son
824-0	Speeches of Charles Dickens
872-0	Reprinted Pieces
882-0	Sketches by Boz
883-0	Our Mutual Friend

book_id	title
888-0	The Lazy Tour of Two Idle Apprentices
912-0	The Mudfog and Other Sketches
914-0	The Uncommercial Traveller
916-0	Sketches of Young Couples
917-0	Barnaby Rudge
918-0	Sketches of Young Gentlemen
922-0	Sunday Under Three Heads
927-0	The Lamplighter
967-0	Nicholas Nickleby
968-0	Martin Chuzzlewit
pg1023	Bleak House
1289-0	Three Ghost Stories
pg1394	The Holly-Tree
1400-0	Great Expectations
pg1406	The Perils of Certain English Prisoners
pg1407	A Message from the Sea
pg1413	Tom Tiddler's Ground
pg1414	Somebody's Luggage
pg1415	Doctor Marigold
pg1416	Mrs. Lirriper's Lodgings
pg1421	Mrs. Lirriper's Legacy
1435-0	Miscellaneous Papers
1467-0	Some Christmas Stories
2324-0	A House to Let
pg19337	A Christmas Carol
pg20795	The Cricket on the Hearth
27924-0	Mugby Junction
pg35536	The Poems and Verses of Charles Dickens

In [7]:
```
# os.chdir('Dickens')
```

In [8]:
```
# !mv 98-0.txt 98-a_tale_of_two_cities.txt
# !mv pg564.txt 564-the_mystery_of_edwin_drood.txt
# !mv 580-0.txt 580-the_pickwick_papers.txt
# !mv 588-0.txt 588-master_humphreys_clock.txt
# !mv 644-0.txt 644-the_haunted_man_and_the_ghosts_bargain.txt
# !mv 650-0.txt 650-pictures_from_italy.txt
# !mv 653-0.txt 653-the_chimes.txt
```

```
# !mv 675-0.txt 675-american_notes.txt
# !mv 676-0.txt 676-the_battle_of_life.txt
# !mv pg699.txt 699-a_childs_history_of_england.txt
# !mv pg700.txt 700-the_old_curiosity_shop.txt
# !mv 730-0.txt 730-oliver_twist.txt
# !mv 766-0.txt 766-david_copperfield.txt
# !mv 786-0.txt 786-hard_times.txt
# !mv 807-0.txt 807-hunted_down.txt
# !mv 809-0.txt 809-holiday_romance.txt
# !mv 810-0.txt 810-george_silvermans_explanation.txt
# !mv pg821.txt 821-dombey_and_sons.txt
# !mv 824-0.txt 824-speeches_of_charles_dickens.txt
# !mv 872-0.txt 872-reprinted_pieces.txt
# !mv 882-0.txt 882-sketches_by_boz.txt
# !mv 883-0.txt 883-our_mutual_friend.txt
# !mv 888-0.txt 888-the_lazy_tour_of_two_idle_apprentices.txt
# !mv 912-0.txt 912-the_mudfog_and_other_sketches.txt
# !mv 914-0.txt 914-the_uncommerical_traveller.txt
# !mv 916-0.txt 916-sketches_of_young_couples.txt
# !mv 917-0.txt 917-barnaby_rudge.txt
# !mv 918-0.txt 918-sketches_of_young_gentlemen.txt
# !mv 922-0.txt 922-sunday_under_three_heads.txt
# !mv 927-0.txt 927-the_lamplighter.txt
# !mv 967-0.txt 967-nicholas_nickleby.txt
# !mv 968-0.txt 968-martin_chuzzlewit.txt
# !mv pg1023.txt 1023-bleak_house.txt
# !mv 1289-0.txt 1289-three_ghost_stories.txt
# !mv pg1394.txt 1394-the_holly_tree.txt
# !mv 1400-0.txt 1400-great_expectations.txt
# !mv pg1406.txt 1406-the_perils_of_certain_english_prisoners.txt
# !mv pg1407.txt 1407-a_message_from_the_sea.txt
# !mv pg1413.txt 1413-tom_tiddlers_ground.txt
# !mv pg1414.txt 1414-somebodys_luggage.txt
# !mv pg1415.txt 1415-doctor_marigold.txt
# !mv pg1416.txt 1416-mrs_lirripers_lodgings.txt
# !mv pg1421.txt 1421-mrs_lirripers_legacy.txt
# !mv 1435-0.txt 1435-miscellaneous_papers.txt
# !mv 1467-0.txt 1467-some_christmas_stories.txt
# !mv 2324-0.txt 2324-a_house_to_let.txt
# !mv pg19337.txt 19337-a_christmas_carol.txt
# !mv pg20795.txt 20795-the_cricket_on_the_hearth.txt
# !mv 27924-0.txt 27924-mugby_junction.txt
# mv pg35536.txt 35536-the_poems_and_verses_of_charles_dickens.txt
```

In [9]:

```
# os.chdir('..')
```

File *Sketches by Boz* (882) with some material contained in other files so want to delete that duplicated material → remove duplicated material from file and create new file

In [10]:

```
# %%bash

# # create copy of current full Sketches by Boz file and rename
# cp Dickens/882-sketches_by_boz.txt 882-sketches_by_boz_full.txt

# # get start line of text to delete (sketches of young gentlemen)
# start=$(grep -n "SKETCHES OF YOUNG GENTLEMEN" 882-sketches_by_boz_full.txt | c
```

```
# # get end line of text to delete (last line before "THE END OF THE PROJECT GUT
# end=$(grep -n "In its original form" 882-sketches_by_boz_full.txt | cut -f1 -d

# # create file where delete material included in other files in CORPUS
# sed -e "${start},${end}d" 882-sketches_by_boz_full.txt > 882-sketches_by_boz.t

# # move new file into Dickens directory and replace old file
# mv 882-sketches_by_boz.txt Dickens
```

Preprocessing Cases with Duplicate Chapter Headings

- **Modified `textparser.py` by Professor Raf Alvarado with the code below to remove duplicates (when chapter headings in the table of contents and the body of the work are exactly the same BUT prevent repeats of same chapter heading if the book has different sections, e.g., I. in Part 1 and I. in Part 2)**

```
# added self.dups in __int__ as a boolean for whether or not to
consider duplicate chapters (default False)
self.dups = dups

# then in parse_tokens() method added the following:
if dups == True:
    chap_duplicates = self.TOKENS.loc[self.TOKENS.duplicated(keep =
'last') & self.TOKENS.line_str.str.contains(div_pat, case =
False)].index.values
    self.TOKENS = self.TOKENS.drop(chap_duplicates)
```

In [11]:

```
# regex roman numeral pattern
roman = '[IVXLCM]+'
```

American Notes for General Circulation (675-0): duplicate chapter headings

In [12]:

```
# encoding argument for open() to strip out character associated with Project Gu

am_notes = 'Dickens/675-american_notes.txt'

# read lines of text file and convert to dataframe
LINES = pd.DataFrame(open(am_notes, 'r', encoding='utf-8-sig').readlines(), colu

# rename index
LINES.index.name = 'line_num'

# replace newline with space and strip whitespace at front and end
LINES.line_str = LINES.line_str.str.replace(r'\n+', ' ', regex=True).str.strip()
```

In [13]:

```
# lists of two regexs for start and end of texts

clip_pats = [
    r"\*\*\*\s*START OF (?:THE|THIS) PROJECT",
```

```
    r"\*\*\*\s*END OF (?:THE|THIS) PROJECT"
]
```

In [14]:
```
# match regexs using .match() method

# Series with boolean values for each line
# only one elt True for each list --> corresponds to line with regex

pat_a = LINES.line_str.str.match(clip_pats[0])
pat_b = LINES.line_str.str.match(clip_pats[1])
```

In [15]:
```
# use pat_a and pat_b as boolean masks for LINES df

# index (line number) of row with front matter and back matter
# increment or decrement by one to exclude the front and back matter

line_a = LINES.loc[pat_a].index[0] + 1
line_b = LINES.loc[pat_b].index[0] - 1
```

In [16]:
```
# slice df using index to remove front and back matter

LINES = LINES.loc[line_a : line_b]
```

In [17]:
```
LINES
```

Out[17]:

line_num	line_str
29	CIRCULATION***
30	
31	
32	Transcribed from the 1913 Chapman & Hall, Ltd....
33	email ccx074@pglaf.org
...	...
10029	been in America, but sufficiently striking to ...
10030	
10031	
10032	
10033	

10005 rows × 1 columns

In [18]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^CHAPTER\s{roman}$"
```

In [19]:
```
# get duplicated lines and keep the last occurrence (i.e., chapter headers withi
chapter_duplicates = LINES.loc[LINES.duplicated(keep = 'last') & LINES.line_str.

# filter out tables of contents chapter lines
LINES = LINES.drop(chapter_duplicates)
```

In [20]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

chap_LINES = LINES.line_str.str.match(chap_pat, case=False)
```

In [21]:
```
LINES.loc[chap_LINES]
```

Out[21]:

line_num	line_str
211	CHAPTER I
505	CHAPTER II
1054	CHAPTER III
2492	CHAPTER IV
2823	CHAPTER V
3187	CHAPTER VI
3898	CHAPTER VII
4499	CHAPTER VIII
5119	CHAPTER IX
5798	CHAPTER X
6197	CHAPTER XI
6533	CHAPTER XII
6973	CHAPTER XIII
7293	CHAPTER XIV
7954	CHAPTER XV
8653	CHAPTER XVI
8954	CHAPTER XVII
9617	CHAPTER XVIII

Hard Times (786-0): duplicate chapter headings

In [22]:
```
# encoding argument for open() to strip out character associated with Project Gu

hard_times = 'Dickens/786-hard_times.txt'

# read lines of text file and convert to dataframe
```

```
LINES = pd.DataFrame(open(hard_times, 'r', encoding='utf-8-sig').readlines(), co

# rename index
LINES.index.name = 'line_num'

# replace newline with space and strip whitespace at front and end
LINES.line_str = LINES.line_str.str.replace(r'\n+', ' ', regex=True).str.strip()
```

In [23]:

```
# lists of two regexs for start and end of texts

clip_pats = [
    r"\*\*\*\s*START OF (?:THE|THIS) PROJECT",
    r"\*\*\*\s*END OF (?:THE|THIS) PROJECT"
]
```

In [24]:

```
# match regexs using .match() method

# Series with boolean values for each line
# only one elt True for each list --> corresponds to line with regex

pat_a = LINES.line_str.str.match(clip_pats[0])
pat_b = LINES.line_str.str.match(clip_pats[1])
```

In [25]:

```
# use pat_a and pat_b as boolean masks for LINES df

# index (line number) of row with front matter and back matter
# increment or decrement by one to exclude the front and back matter

line_a = LINES.loc[pat_a].index[0] + 1
line_b = LINES.loc[pat_b].index[0] - 1
```

In [26]:

```
# slice df using index to remove front and back matter

LINES = LINES.loc[line_a : line_b]
```

In [27]:

```
LINES
```

Out[27]:

line_num	line_str
28	
29	
30	Transcribed from the 1905 Chapman and Hall edi...
31	ccx074@pglaf.org
32	
...	...
11674	Gutenberg, and is not included in this eText.

line_num	line_str
11675	
11676	
11677	
11678	

11651 rows × 1 columns

In [28]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

chap_LINES_duplicated = LINES.line_str.str.match(chap_pat, case=False)

# remove duplicates (first half of chap_LINES_duplicated) and get index values
chap_LINES = LINES.loc[chap_LINES_duplicated].iloc[int(LINES.loc[chap_LINES_dupl
```

In [29]:
```
LINES.loc[chap_LINES]
```

Out[29]:

line_num	line_str
157	CHAPTER I
194	CHAPTER II
439	CHAPTER III
628	CHAPTER IV
921	CHAPTER V
1150	CHAPTER VI
1693	CHAPTER VII
1965	CHAPTER VIII
2204	CHAPTER IX
2534	CHAPTER X
2757	CHAPTER XI
3068	CHAPTER XII
3269	CHAPTER XIII
3588	CHAPTER XIV
3807	CHAPTER XV
4124	CHAPTER XVI
4348	CHAPTER I
4899	CHAPTER II
5216	CHAPTER III

line_num	line_str
5434	CHAPTER IV
5728	CHAPTER V
6008	CHAPTER VI
6516	CHAPTER VII
7031	CHAPTER VIII
7552	CHAPTER IX
7886	CHAPTER X
8067	CHAPTER XI
8422	CHAPTER XII
8623	CHAPTER I
8887	CHAPTER II
9283	CHAPTER III
9629	CHAPTER IV
10008	CHAPTER V
10356	CHAPTER VI
10748	CHAPTER VII
11197	CHAPTER VIII
11456	CHAPTER IX

Great Expectations (1400-0): duplicate chapter headings

In [30]:
```
# encoding argument for open() to strip out character associated with Project Gu

ge = 'Dickens/1400-great_expectations.txt'

# read lines of text file and convert to dataframe
LINES = pd.DataFrame(open(ge, 'r', encoding='utf-8-sig').readlines(), columns=['

# rename index
LINES.index.name = 'line_num'

# replace newline with space and strip whitespace at front and end
LINES.line_str = LINES.line_str.str.replace(r'\n+', ' ', regex=True).str.strip()
```

In [31]:
```
# lists of two regexs for start and end of texts

clip_pats = [
    r"\*\*\*\s*START OF (?:THE|THIS) PROJECT",
    r"\*\*\*\s*END OF (?:THE|THIS) PROJECT"
]
```

In [32]:
```
# match regexs using .match() method

# Series with boolean values for each line
# only one elt True for each list --> corresponds to line with regex

pat_a = LINES.line_str.str.match(clip_pats[0])
pat_b = LINES.line_str.str.match(clip_pats[1])
```

In [33]:
```
# use pat_a and pat_b as boolean masks for LINES df

# index (line number) of row with front matter and back matter
# increment or decrement by one to exclude the front and back matter

line_a = LINES.loc[pat_a].index[0] + 1
line_b = LINES.loc[pat_b].index[0] - 1
```

In [34]:
```
# slice df using index to remove front and back matter

LINES = LINES.loc[line_a : line_b]
```

In [35]:
```
# get duplicated lines and keep the last occurrence (i.e., chapter headers withi
chapter_duplicates = LINES.loc[LINES.duplicated(keep = 'last') & LINES.line_str.

# filter out tables of contents chapter lines
LINES = LINES.drop(chapter_duplicates)
```

In [36]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^\s*Chapter\s*{roman}"
```

In [37]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

chap_LINES = LINES.line_str.str.match(chap_pat, case=False)
```

In [38]:
```
LINES.loc[chap_LINES]
```

Out[38]:

line_num	line_str
104	Chapter I.
314	Chapter II.
677	Chapter III.
901	Chapter IV.
1252	Chapter V.
1685	Chapter VI.
1757	Chapter VII.

line_num	line_str
2208	Chapter VIII.
2733	Chapter IX.
3049	Chapter X.
3335	Chapter XI.
3939	Chapter XII.
4159	Chapter XIII.
4489	Chapter XIV.
4568	Chapter XV.
5044	Chapter XVI.
5227	Chapter XVII.
5594	Chapter XVIII.
6183	Chapter XIX.
6789	Chapter XX.
7143	Chapter XXI.
7351	Chapter XXII.
7906	Chapter XXIII.
8240	Chapter XXIV.
8501	Chapter XXV.
8818	Chapter XXVI.
9121	Chapter XXVII.
9461	Chapter XXVIII.
9716	Chapter XXIX.
10282	Chapter XXX.
10659	Chapter XXXI.
10885	Chapter XXXII.
11134	Chapter XXXIII.
11442	Chapter XXXIV.
11678	Chapter XXXV.
12009	Chapter XXXVI.
12324	Chapter XXXVII.
12609	Chapter XXXVIII.
13168	Chapter XXXIX.
13693	Chapter XL.
14274	Chapter XLI.

line_num	line_str
14525	Chapter XLII.
14821	Chapter XLIII.
15065	Chapter XLIV.
15405	Chapter XLV.
15723	Chapter XLVI.
16025	Chapter XLVII.
16287	Chapter XLVIII.
16594	Chapter XLIX.
16992	Chapter L.
17181	Chapter LI.
17516	Chapter LII.
17742	Chapter LIII.
18263	Chapter LIV.
18826	Chapter LV.
19145	Chapter LVI.
19369	Chapter LVII.
19903	Chapter LVIII.
20240	Chapter LIX.

A Child's History of England (pg699): duplicate chapter heading (first one only)

In [39]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Dickens/699-a_childs_history_of_england.txt'

# read lines of text file and convert to dataframe
LINES = pd.DataFrame(open(text_file, 'r', encoding='utf-8-sig').readlines(), col

# rename index
LINES.index.name = 'line_num'

# replace newline with space and strip whitespace at front and end
LINES.line_str = LINES.line_str.str.replace(r'\n+', ' ', regex=True).str.strip()
```

In [40]:
```
# lists of two regexs for start and end of texts

clip_pats = [
    r"\*\*\*\s*START OF (?:THE|THIS) PROJECT",
    r"\*\*\*\s*END OF (?:THE|THIS) PROJECT"
]
```

In [41]:
```
# match regexs using .match() method

# Series with boolean values for each line
# only one elt True for each list --> corresponds to line with regex

pat_a = LINES.line_str.str.match(clip_pats[0])
pat_b = LINES.line_str.str.match(clip_pats[1])
```

In [42]:
```
# use pat_a and pat_b as boolean masks for LINES df

# index (line number) of row with front matter and back matter
# increment or decrement by one to exclude the front and back matter

line_a = LINES.loc[pat_a].index[0] + 1
line_b = LINES.loc[pat_b].index[0] - 1
```

In [43]:
```
# get duplicated lines and keep the last occurrence (i.e., chapter headers withi
chapter_duplicates = LINES.loc[LINES.duplicated(keep = 'last') & LINES.line_str.

# filter out tables of contents chapter lines
LINES = LINES.drop(chapter_duplicates)
```

In [44]:
```
# slice df using index to remove front and back matter

LINES = LINES.loc[line_a : line_b]
```

In [45]:
```
LINES
```

Out[45]:

line_num	line_str
23	
24	
25	
26	
27	A CHILD’S HISTORY OF ENGLAND
...	...
14888	God Save the Queen!
14889	
14890	
14891	
14892	

14869 rows × 1 columns

In [46]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^CHAPTER\s{roman}$"
```

In [47]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

chap_lines = LINES.line_str.str.match(chap_pat, case=False)
```

In [48]:
```
LINES.loc[chap_lines]
```

Out[48]:

line_num	line_str
81	CHAPTER I
421	CHAPTER II
599	CHAPTER III
807	CHAPTER IV
1221	CHAPTER V
1290	CHAPTER VI
1555	CHAPTER VII
1723	CHAPTER VIII
1987	CHAPTER IX
2247	CHAPTER X
2623	CHAPTER XI
2759	CHAPTER XII
3475	CHAPTER XIII
3823	CHAPTER XIV
4275	CHAPTER XV
4719	CHAPTER XVI
5326	CHAPTER XVII
5679	CHAPTER XVIII
6148	CHAPTER XIX
6550	CHAPTER XX
6757	CHAPTER XXI
7137	CHAPTER XXII
7845	CHAPTER XXIII
8132	CHAPTER XXIV
8313	CHAPTER XXV
8474	CHAPTER XXVI

line_num	line_str
8860	CHAPTER XXVII
9301	CHAPTER XXVIII
9670	CHAPTER XXIX
9951	CHAPTER XXX
10411	CHAPTER XXXI
11312	CHAPTER XXXII
11898	CHAPTER XXXIII
12954	CHAPTER XXXIV
13547	CHAPTER XXXV
14333	CHAPTER XXXVI
14813	CHAPTER XXXVII

Identifiers and Regexes

Specific Chapter Patterns

In [49]:
```
# project gutenberg
gutenberg_clip_pats = [
    r"\*\*\*\s*START OF (?:THE|THIS) PROJECT",
    r"\*\*\*\s*END OF (?:THE|THIS) PROJECT"
]
```

Chapter names for Pictures from Italy (650-0) scraped from Project Gutenberg Index of Charles Dickens works

In [50]:
```
# regex to identify lines in text that act as headers for chapters for Pictures

# first line in html source code for given book
ital_start_line = index.find('h1', text = re.compile(r'PICTURES FROM ITALY')).so

# last line in html source code for given book --> first occurrence of "LIST OF
ital_end_line = [i.sourceline for i in index.find_all('h2', text = re.compile(r'

# text of all p tags after start line and before end line
ital_chap_pats = [i.text.strip() for i in index.find_all('p') if i.sourceline >

# take out p tags with empty strings and numbers and make everything uppercase
ital_chap_pats = [i.replace('\r\n            ', ' ').upper() for i in ital_chap_

ital_chap_pat = '|'.join(ital_chap_pats)

ital_chap_pat = rf'{ital_chap_pat}$'
```

Chapter names for Reprinted Pieces (872-0) scraped from Project Gutenberg Index of Charles Dickens works

In [51]:
```
# regex to identify lines in text that act as headers for chapters for Reprinted

# first line in html source code for given book
reprint_start_line = index.find('h1', text = re.compile(r'REPRINTED PIECES')).so

# last line in html source code for given book --> first occurrence of "OUR MUTU
reprint_end_line = [i.sourceline for i in index.find_all('h1', text = re.compile

# text of all p tags after start line and before end line
reprint_chap_pats = [i.text.strip() for i in index.find_all('p') if i.sourceline

# take out p tags with empty strings and numbers and make everything uppercase a
reprint_chap_pats = [i.replace('\r\n            ', ' ').replace('“Births.\xa0 Mr
                     for i in reprint_chap_pats if i != '' and not re.match((rf'

reprint_chap_pat = '$|'.join(reprint_chap_pats)

reprint_chap_pat = rf'{reprint_chap_pat}$'
```

Chapter names for The Mudfog and Other Sketches (912-0) scraped from Project Gutenberg Index of Charles Dickens works

In [52]:
```
# regex to identify lines in text that act as headers for chapters for The Mudfo

# first line in html source code for given book
mudfog_start_line = index.find('h1', text = re.compile(r'MUDFOG')).sourceline

# last line in html source code for given book --> first occurrence of "THE UNCO
mudfog_end_line = [i.sourceline for i in index.find_all('h1', text = re.compile(

# text of all p tags after start line and before end line
mudfog_chap_pats = [i.text.strip() for i in index.find_all('p') if i.sourceline

# take out p tags with empty strings and numbers and make everything uppercase a
mudfog_chap_pats = [re.sub(r'[A-Z]+\sMEETING .*', '', i.replace('\r\n

mudfog_chap_pat = '$|'.join(mudfog_chap_pats)

mudfog_chap_pat = rf'{mudfog_chap_pat}$'
```

Chapter names for Sketches of Young Couples (916-0) scraped from Project Gutenberg Index of Charles Dickens works

In [53]:
```
# regex to identify lines in text that act as headers for chapters for Sketches

# first line in html source code for given book
young_start_line = index.find('h1', text = re.compile(r'YOUNG COUPLES')).sourcel

# last line in html source code for given book --> first occurrence of "BARNABY
young_end_line = [i.sourceline for i in index.find_all('h1', text = re.compile(r

# text of all p tags after start line and before end line
young_chap_pats = [i.text.strip() for i in index.find_all('p') if i.sourceline >

# take out p tags with empty strings and numbers and make everything uppercase a
young_chap_pats = [i.replace('\r\n            ', '').upper().strip() for i in yo
```

```
young_chap_pat = '$|'.join(young_chap_pats)

young_chap_pat = rf'{young_chap_pat}$'
```

Chapter names for Sketches of Young Gentlemen (918-0) scraped from Project Gutenberg Index of Charles Dickens works

In [54]:

```
# regex to identify lines in text that act as headers for chapters for Sketches

# first line in html source code for given book
gentlemen_start_line = index.find('h1', text = re.compile(r'YOUNG GENTLEMEN')).s

# last line in html source code for given book --> first occurrence of "THE LIFE
gentlemen_end_line = [i.sourceline for i in index.find_all('h1', text = re.compi

# text of all p tags after start line and before end line
gentlemen_chap_pats = [i.text.strip() for i in index.find_all('p') if i.sourceli

# take out p tags with empty strings and numbers and make everything uppercase a
gentlemen_chap_pats = [i.replace('\r\n            ', '').upper().strip() for i i

gentlemen_chap_pat = '$|'.join(gentlemen_chap_pats)

gentlemen_chap_pat = rf'{gentlemen_chap_pat}$'
```

Chapter names for Three Ghost Stories (1289-0) scraped from Project Gutenberg Index of Charles Dickens works

In [55]:

```
# regex to identify lines in text that act as headers for chapters for Three Gho

# first line in html source code for given book
ghost_start_line = index.find('h1', text = re.compile(r'GHOST STORIES')).sourcel

# last line in html source code for given book --> first occurrence of "GREAT EX
ghost_end_line = [i.sourceline for i in index.find_all('h1', text = re.compile(r

# text of all p tags after start line and before end line
ghost_chap_pats = [i.text.strip() for i in index.find_all('p') if i.sourceline >

# take out p tags with empty strings and numbers and make everything uppercase a
ghost_chap_pats = [i.replace('\r\n            ', '').upper().strip() for i in gh

ghost_chap_pat = '\.|'.join(ghost_chap_pats)

ghost_chap_pat = rf'{ghost_chap_pat}\.'
```

Chapter names for Some Christmas Stories (1467-0) scraped from Project Gutenberg Index of Charles Dickens works

In [56]:

```
# regex to identify lines in text that act as headers for chapters for Some Chri

# first line in html source code for given book
christmas_start_line = index.find('h1', text = re.compile(r'CHRISTMAS STORIES'))

# last line in html source code for given book --> first occurrence of "THE CRIC
```

```
christmas_end_line = [i.sourceline for i in index.find_all('h1', text = re.compi

# text of all p tags after start line and before end line
christmas_chap_pats = [i.text.strip() for i in index.find_all('p') if i.sourceli

# take out p tags with empty strings and numbers and make everything uppercase a
christmas_chap_pats = [i.replace('\r\n            ', '').upper().strip() for i i

christmas_chap_pat = '[\.]?|'.join(christmas_chap_pats)

christmas_chap_pat = rf'{christmas_chap_pat}[\.]?$'
```

Chapter names for Mugby Junction (27924-0) but NOT scraped from Project Gutenberg Index of Charles Dickens works (easier to manually define given contents in index)

In [57]:

```
# regex to identify lines in text that act as headers for chapters for Mugby Jun

mugby_chap_pats = ['BARBOX BROTHERS$',
                   'BARBOX BROTHERS AND CO\.$',
                   'MAIN LINE: THE BOY AT MUGBY$',
                   'NO. [0-9] BRANCH LINE$'
]

mugby_chap_pat = '|'.join(mugby_chap_pats)

mugby_chap_pat = rf'{mugby_chap_pat}'
```

Chapter names for Poems and Verses (pg35536) but NOT scraped from Project Gutenberg Index of Charles Dickens works (easier to manually define ggiven idiosyncrasies in text)

In [58]:

```
# regex to identify lines in text that act as headers for chapters for Poems and

poem_chap_pats = ['THE VILLAGE COQUETTES$',
                  'THE LAMPLIGHTER$',
                  'SONGS FROM \'THE PICKWICK PAPERS\'$',
                  'POLITICAL SQUIBS FROM \'THE EXAMINER\'',
                  'PROLOGUE TO \'THE PATRICIAN\'S DAUGHTER\'$',
                  'A WORD IN SEASON FROM THE \'KEEPSAKE\'$',
                  'VERSES FROM THE \'DAILY NEWS\'$',
                  'NEW SONG LINES ADDRESSED TO MARK LEMON$',
                  'WILKIE COLLINS\'S PLAY \'THE LIGHTHOUSE\'$',
                  'PROLOGUE TO WILKIE COLLINS\'S PLAY \'THE FROZEN DEEP\'$',
                  'A CHILD\'S HYMN FROM \'THE WRECK OF THE GOLDEN MARY\'$'
]

poem_chap_pat = '|'.join(poem_chap_pats)

poem_chap_pat = rf'{poem_chap_pat}'
```

Chapter names for Miscellaneous Papers (1435-0) but NOT scraped from Project Gutenberg Index of Charles Dickens works (no table of contents provided in index for this work)

In [59]:

```
# regex to identify lines in text that act as headers for chapters for Miscellan

paper_chap_pats = ['THE AGRICULTURAL INTEREST$',
                   'THREATENING LETTER TO THOMAS HOOD$',
                   'CRIME AND EDUCATION$',
                   'CAPITAL PUNISHMENT$',
                   'THE SPIRIT OF CHIVALRY IN WESTMINSTER HALL$',
                   'IN MEMORIAM$',
                   'ADELAIDE ANNE PROCTER$',
                   '^CHAUNCEY HARE TOWNSHEND$',
                   'ON MR. FECHTER'S ACTING$'
]

paper_chap_pat = '|'.join(paper_chap_pats)

paper_chap_pat = rf'{paper_chap_pat}'
```

Chapter names for A House to Let (2324-0) but NOT scraped from Project Gutenberg Index of Charles Dickens works (no table of contents provided in index for this work)

In [60]:

```
# regex to identify lines in text that act as headers for chapters for A House t

house_chap_pats = ['OVER THE WAY$',
                   'THE MANCHESTER MARRIAGE$',
                   'GOING INTO SOCIETY$',
                   'THREE EVENINGS IN THE HOUSE$',
                   'TROTTLE'S REPORT$',
                   'LET AT LAST$'
]

house_chap_pat = '|'.join(house_chap_pats)

house_chap_pat = rf'{house_chap_pat}'
```

In [61]:

```
# All are 'chap'and 'm' (milestone)
ohco_pat_list = [
    (98, rf"^\s*CHAPTER\s*{roman}\.$", False),
    (564, rf"^CHAPTER\s{roman}\.$", False),
    (580, rf"^CHAPTER\s{roman}\.\s[A-Z]+", False),
    (588, rf"^(?:{roman}$|TO THE READERS OF)", False),
    (644, rf"^CHAPTER\s{roman}$", False),
    (650, ital_chap_pat, False),
    (653, rf"^CHAPTER\s{roman}", False),
    (675, rf"^CHAPTER\s{roman}$", True),
    (676, rf"^Part the [A-Z][a-z]+$", False),
    (699, rf"^CHAPTER\s{roman}$", True),
    (700, rf"^CHAPTER\s", False),
    (730, rf"^\s*CHAPTER\s*{roman}\.$", False),
    (766, rf"\s*(PREFACE\sTO|CHAPTER\s*[0-9]*)", False),
    (786, rf"CHAPTER\s{roman}", True),
    (807, rf"^{roman}\.$", False),
    (809, rf"^PART\s{roman}\.$", False),
    (810, rf"[A-Z]+\sCHAPTER$", False),
    (821, rf"^\s*CHAPTER\s*{roman}\.$", False),
    (824, rf"{roman}\.$", False),
    (872, reprint_chap_pat, False),
```

```
    (882, rf"^(PREFACE|CHAPTER\s{roman})", False),
    (883, rf"^\s*Chapter\s*", False),
    (888, rf"CHAPTER\s{roman}$", False),
    (912, mudfog_chap_pat, False),
    (914, rf"^{roman}$", False),
    (916, young_chap_pat, False),
    (917, rf"^Chapter\s([0-9]+|the Last)", False),
    (918, gentlemen_chap_pat, False),
    (922, rf"^{roman}$", False),
    (927, rf"^‘IF", False), # no chapters so use regex that matches first line
    (967, rf"^(AUTHOR’S PREFACE|CHAPTER\s[0-9]+|Conclusion$)", False),
    (968, rf"^(PREFACE|CHAPTER\s[A-Z]+[-]?[A-Z]+$)", False),
    (1023, rf"^\s*(PREFACE|CHAPTER\s*{roman})$", False),
    (1289, ghost_chap_pat, False),
    (1394, rf"^[A-Z]+\sBRANCH", False),
    (1400, rf"^\s*Chapter\s*{roman}", True),
    (1406, rf"^CHAPTER\s{roman}", False), # Dickens did not write the second cha
    (1407, rf"^CHAPTER\s{roman}", False), # Dickens did not write the third and
    (1413, rf"^CHAPTER\s{roman}", False), # DIckens did not write the second and
    (1414, rf"^CHAPTER\s{roman}", False),
    (1415, rf"\* \* \* \* \*", False),
    (1416, rf"^CHAPTER\s{roman}", False),
    (1421, rf"^CHAPTER\s{roman}", False),
    (1435, paper_chap_pat, True),
    (1467, christmas_chap_pat, False),
    (2324, house_chap_pat, False),
    (19337, rf"^\s*STAVE\s[A-Z]+$", False),
    (20795, rf"^CHIRP\sTHE", False),
    (27924, mugby_chap_pat, False),
    (35536, poem_chap_pat, False)
]
```

In [62]:
```
source_files = f'Dickens'
```

In [63]:
```
source_file_list = sorted(glob(f"{source_files}/*.*"))
```

In [64]:
```
len(source_file_list)
```

Out[64]: 50

LIB, CORPUS, and VOCAB Tables

In [65]:
```
book_data = []
for source_file_path in source_file_list:
    book_id = int(source_file_path.split("/")[-1].split('-')[0])
    book_title = source_file_path.split('/')[-1].split('-')[-1].split('.')[0].re
    book_data.append((book_id, source_file_path, book_title))
```

In [66]:
```
LIB = pd.DataFrame(book_data, columns=['book_id','source_file_path','title']) \
        .set_index('book_id').sort_index()
```

In [67]:
```
LIB
```

Out[67]:

book_id	source_file_path	title
98	Dickens/98-a_tale_of_two_cities.txt	a tale of two cities
564	Dickens/564-the_mystery_of_edwin_drood.txt	the mystery of edwin drood
580	Dickens/580-the_pickwick_papers.txt	the pickwick papers
588	Dickens/588-master_humphreys_clock.txt	master humphreys clock
644	Dickens/644-the_haunted_man_and_the_ghosts_bar...	the haunted man and the ghosts bargain
650	Dickens/650-pictures_from_italy.txt	pictures from italy
653	Dickens/653-the_chimes.txt	the chimes
675	Dickens/675-american_notes.txt	american notes
676	Dickens/676-the_battle_of_life.txt	the battle of life
699	Dickens/699-a_childs_history_of_england.txt	a childs history of england
700	Dickens/700-the_old_curiosity_shop.txt	the old curiosity shop
730	Dickens/730-oliver_twist.txt	oliver twist
766	Dickens/766-david_copperfield.txt	david copperfield
786	Dickens/786-hard_times.txt	hard times
807	Dickens/807-hunted_down.txt	hunted down
809	Dickens/809-holiday_romance.txt	holiday romance
810	Dickens/810-george_silvermans_explanation.txt	george silvermans explanation
821	Dickens/821-dombey_and_sons.txt	dombey and sons
824	Dickens/824-speeches_of_charles_dickens.txt	speeches of charles dickens
872	Dickens/872-reprinted_pieces.txt	reprinted pieces
882	Dickens/882-sketches_by_boz.txt	sketches by boz
883	Dickens/883-our_mutual_friend.txt	our mutual friend
888	Dickens/888-the_lazy_tour_of_two_idle_apprenti...	the lazy tour of two idle apprentices
912	Dickens/912-the_mudfog_and_other_sketches.txt	the mudfog and other sketches
914	Dickens/914-the_uncommerical_traveller.txt	the uncommerical traveller
916	Dickens/916-sketches_of_young_couples.txt	sketches of young couples
917	Dickens/917-barnaby_rudge.txt	barnaby rudge
918	Dickens/918-sketches_of_young_gentlemen.txt	sketches of young gentlemen
922	Dickens/922-sunday_under_three_heads.txt	sunday under three heads
927	Dickens/927-the_lamplighter.txt	the lamplighter
967	Dickens/967-nicholas_nickleby.txt	nicholas nickleby
968	Dickens/968-martin_chuzzlewit.txt	martin chuzzlewit

book_id	source_file_path	title
1023	Dickens/1023-bleak_house.txt	bleak house
1289	Dickens/1289-three_ghost_stories.txt	three ghost stories
1394	Dickens/1394-the_holly_tree.txt	the holly tree
1400	Dickens/1400-great_expectations.txt	great expectations
1406	Dickens/1406-the_perils_of_certain_english_pri...	the perils of certain english prisoners
1407	Dickens/1407-a_message_from_the_sea.txt	a message from the sea
1413	Dickens/1413-tom_tiddlers_ground.txt	tom tiddlers ground
1414	Dickens/1414-somebodys_luggage.txt	somebodys luggage
1415	Dickens/1415-doctor_marigold.txt	doctor marigold
1416	Dickens/1416-mrs_lirripers_lodgings.txt	mrs lirripers lodgings
1421	Dickens/1421-mrs_lirripers_legacy.txt	mrs lirripers legacy
1435	Dickens/1435-miscellaneous_papers.txt	miscellaneous papers
1467	Dickens/1467-some_christmas_stories.txt	some christmas stories
2324	Dickens/2324-a_house_to_let.txt	a house to let
19337	Dickens/19337-a_christmas_carol.txt	a christmas carol
20795	Dickens/20795-the_cricket_on_the_hearth.txt	the cricket on the hearth
27924	Dickens/27924-mugby_junction.txt	mugby junction
35536	Dickens/35536-the_poems_and_verses_of_charles_...	the poems and verses of charles dickens

In [68]:
```
LIB['chap_regex'] = LIB.index.map(pd.Series({x[0]:x[1] for x in ohco_pat_list}))
```

In [69]:
```
LIB['author'] = 'dickens'
```

In [70]:
```
# type ids --> dict with keys: types, values: book ids

type_ids =  {'novel': [98, 564, 580, 653, 676, 700, 730, 766, 786, 821, 883, 967
             'stories': [588, 644, 807, 809, 810, 872, 882, 888, 912, 916, 917,
             'non-fiction': [650, 675, 699, 824, 914, 922, 1435]
}
```

In [71]:
```
# create dict with each key a book id, genres the values
id_types = {k: og_key for (og_key, og_value) in type_ids.items() for k in og_val
```

In [72]:
```
# map to create new col with types for each work
LIB['type'] = LIB.index.map(id_types)
```

In [73]:
```
# add year when book published

book_year = ((98, 1859),
             (564, 1870),
             (580, 1836),
             (588, 1840),
             (644, 1848),
             (650, 1846),
             (653, 1844),
             (675, 1842),
             (676, 1846),
             (699, 1853),
             (700, 1840),
             (730, 1837),
             (766, 1849),
             (786, 1854),
             (807, 1859),
             (809, 1868),
             (810, 1868),
             (821, 1846),
             (824, 1870),
             (872, 1861),
             (882, 1836),
             (883, 1864),
             (888, 1857),
             (912, 1837),
             (914, 1860),
             (916, 1840),
             (917, 1841),
             (918, 1838),
             (922, 1836),
             (927, 1838),
             (967, 1838),
             (968, 1842),
             (1023, 1852),
             (1289, 1860),
             (1394, 1855),
             (1400, 1860),
             (1406, 1857),
             (1407, 1860),
             (1413, 1861),
             (1414, 1862),
             (1415, 1865),
             (1416, 1863),
             (1421, 1864),
             (1435, 1840),
             (1467, 1850),
             (2324, 1858),
             (19337, 1843),
             (20795, 1845),
             (27924, 1866),
             (35536, 1885)
)
```

In [74]:
```
LIB['year'] = LIB.index.map(pd.Series({x[0]: x[1] for x in book_year}))
```

In [75]:
```
bins = [1830, 1840, 1850, 1860, 1870, 1880, 1890]
```

```
LIB['decade'] = pd.cut(LIB['year'], bins = bins, labels = bins[:-1], right = Fal
```

In [76]:

```
LIB.head()
```

Out[76]:

book_id	source_file_path	title	chap_regex	author	
98	Dickens/98-a_tale_of_two_cities.txt	a tale of two cities	^\s*CHAPTER\s*[IVXLCM]+\.$	dickens	
564	Dickens/564-the_mystery_of_edwin_drood.txt	the mystery of edwin drood	^CHAPTER\s[IVXLCM]+\.$	dickens	
580	Dickens/580-the_pickwick_papers.txt	the pickwick papers	^CHAPTER\s[IVXLCM]+\.\s[A-Z]+	dickens	
588	Dickens/588-master_humphreys_clock.txt	master humphreys clock	^(?:[IVXLCM]+$|TO THE READERS OF)	dickens	s
644	Dickens/644-the_haunted_man_and_the_ghosts_bar...	the haunted man and the ghosts bargain	^CHAPTER\s[IVXLCM]+$	dickens	s

In [77]:

```
books = []
for pat in ohco_pat_list:

    book_id, chap_regex = pat[0], pat[1]
    print("Tokenizing", book_id, LIB.loc[book_id].title)
    ohco_pats = [('chap', chap_regex, 'm')]
    src_file_path = LIB.loc[book_id].source_file_path
    dups = pat[2]

    text = TextParser(src_file_path, ohco_pats=ohco_pats, clip_pats=clip_pats, u
    text.verbose = False
    text.strip_hyphens = True
    text.strip_whitespace = True
    text.import_source().parse_tokens();
    text.TOKENS['book_id'] = book_id
    text.TOKENS = text.TOKENS.reset_index().set_index(['book_id'] + text.OHCO)

    books.append(text.TOKENS)
```

```
Tokenizing 98 a tale of two cities
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 564 the mystery of edwin drood
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 580 the pickwick papers
line_str chap_str
Index(['chap_str'], dtype='object')
```

```
Tokenizing 588 master humphreys clock
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 644 the haunted man and the ghosts bargain
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 650 pictures from italy
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 653 the chimes
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 675 american notes
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 676 the battle of life
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 699 a childs history of england
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 700 the old curiosity shop
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 730 oliver twist
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 766 david copperfield
/Users/cecilyestherwolfe/Desktop/Spring_2022/DS5001/Project/textparser.py:136: U
serWarning: This pattern has match groups. To actually get the groups, use str.e
xtract.
  div_lines = self.TOKENS[src_col].str.contains(div_pat, regex=True, case=True)
# TODO: Parametize case
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 786 hard times
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 807 hunted down
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 809 holiday romance
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 810 george silvermans explanation
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 821 dombey and sons
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 824 speeches of charles dickens
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 872 reprinted pieces
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 882 sketches by boz
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 883 our mutual friend
```

```
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 888 the lazy tour of two idle apprentices
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 912 the mudfog and other sketches
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 914 the uncommerical traveller
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 916 sketches of young couples
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 917 barnaby rudge
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 918 sketches of young gentlemen
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 922 sunday under three heads
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 927 the lamplighter
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 967 nicholas nickleby
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 968 martin chuzzlewit
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1023 bleak house
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1289 three ghost stories
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1394 the holly tree
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1400 great expectations
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1406 the perils of certain english prisoners
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1407 a message from the sea
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1413 tom tiddlers ground
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1414 somebodys luggage
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1415 doctor marigold
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1416 mrs lirripers lodgings
```

```
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1421 mrs lirripers legacy
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1435 miscellaneous papers
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1467 some christmas stories
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 2324 a house to let
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 19337 a christmas carol
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 20795 the cricket on the hearth
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 27924 mugby junction
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 35536 the poems and verses of charles dickens
line_str chap_str
Index(['chap_str'], dtype='object')
```

In [78]:
```
CORPUS = pd.concat(books).sort_index()
```

In [79]:
```
CORPUS = CORPUS[CORPUS.term_str != '']

CORPUS = CORPUS.loc[~CORPUS.term_str.isna()]

CORPUS = CORPUS.loc[~CORPUS.term_str.str.contains('jpg', case = False, regex = T
```

In [80]:
```
CORPUS
```

Out[80]:

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
98	1	0	0	0	(The, DT)	DT	The	the
				1	(Period, NN)	NN	Period	period
		1	0	0	(It, PRP)	PRP	It	it
				1	(was, VBD)	VBD	was	was
				2	(the, DT)	DT	the	the
...	...	...	...	...	...	...	...	...
35536	13	16	0	12	(Charles, NNP)	NNP	Charles	charles
				13	(Dickens,, NNP)	NNP	Dickens,	dickens

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
				14	(by, IN)	IN	by	by
				15	(Charles, NNP)	NNP	Charles	charles
				16	(Dickens, NNP)	NNP	Dickens	dickens

4969969 rows × 4 columns

In [81]:
```
# number of chapters in book
LIB['n_chaps'] = CORPUS.reset_index()[['book_id', 'chap_id']] \
                  .drop_duplicates() \
                  .groupby('book_id').chap_id.count()
```

In [82]:
```
# length of each book (number of tokens)
LIB['book_len'] = CORPUS.groupby('book_id').agg({'token_str': 'count'})
```

In [83]:
```
LIB
```

Out[83]:

book_id	source_file_path	title	chap_r
98	Dickens/98-a_tale_of_two_cities.txt	a tale of two cities	^\s*CHAPTER\s*[IVXLCM
564	Dickens/564-the_mystery_of_edwin_drood.txt	the mystery of edwin drood	^CHAPTER\s[IVXLCM
580	Dickens/580-the_pickwick_papers.txt	the pickwick papers	^CHAPTER\s[IVXLCM]+\.\s[A
588	Dickens/588-master_humphreys_clock.txt	master humphreys clock	^(?:[IVXLCM]+$|TO THE READER
644	Dickens/644-the_haunted_man_and_the_ghosts_bar...	the haunted man and the ghosts bargain	^CHAPTER\s[IVXLC
650	Dickens/650-pictures_from_italy.txt	pictures from italy	THE READER'S PASSPORT|GOING THRC FRANCE|I
653	Dickens/653-the_chimes.txt	the chimes	^CHAPTER\s[IVXL
675	Dickens/675-american_notes.txt	american notes	^CHAPTER\s[IVXLC
676	Dickens/676-the_battle_of_life.txt	the battle of life	^Part the [A-Z][a

book_id	source_file_path	title	chap_r
699	Dickens/699-a_childs_history_of_england.txt	a childs history of england	^CHAPTER\s[IVXLC
700	Dickens/700-the_old_curiosity_shop.txt	the old curiosity shop	^CHAP
730	Dickens/730-oliver_twist.txt	oliver twist	^\s*CHAPTER\s*[IVXLCM
766	Dickens/766-david_copperfield.txt	david copperfield	\s*(PREFACE\sTO|CHAPTER\s*[C
786	Dickens/786-hard_times.txt	hard times	CHAPTER\s[IVXL
807	Dickens/807-hunted_down.txt	hunted down	^[IVXLCM
809	Dickens/809-holiday_romance.txt	holiday romance	^PART\s[IVXLCM
810	Dickens/810-george_silvermans_explanation.txt	george silvermans explanation	[A-Z]+\sCHAP
821	Dickens/821-dombey_and_sons.txt	dombey and sons	^\s*CHAPTER\s*[IVXLCM
824	Dickens/824-speeches_of_charles_dickens.txt	speeches of charles dickens	[IVXLCM
872	Dickens/872-reprinted_pieces.txt	reprinted pieces	THE LONG VO $\|THEBEGGING - LETTERWRI$
882	Dickens/882-sketches_by_boz.txt	sketches by boz	^(PREFACE|CHAPTER\s[IVXLC
883	Dickens/883-our_mutual_friend.txt	our mutual friend	^\s*Chap
888	Dickens/888-the_lazy_tour_of_two_idle_apprenti...	the lazy tour of two idle apprentices	CHAPTER\s[IVXLC
912	Dickens/912-the_mudfog_and_other_sketches.txt	the mudfog and other sketches	PUBLIC LIFE OF MR. TULRUMBLE$| REPORT C
914	Dickens/914-the_uncommerical_traveller.txt	the uncommerical traveller	^[IVXLC
916	Dickens/916-sketches_of_young_couples.txt	sketches of young couples	AN URGENT REMONSTRANCE $\|THEYOUNGCOUP$
917	Dickens/917-barnaby_rudge.txt	barnaby rudge	^Chapter\s([0-9]+|the
918	Dickens/918-sketches_of_young_gentlemen.txt	sketches of young gentlemen	THE BASHFUL YOUNG GENTLEMAN$ OUT-AND-OU

book_id	source_file_path	title	chap_r
922	Dickens/922-sunday_under_three_heads.txt	sunday under three heads	^[IVXLC
927	Dickens/927-the_lamplighter.txt	the lamplighter	
967	Dickens/967-nicholas_nickleby.txt	nicholas nickleby	^(AUTHOR'S PREFACE|CHAPTER 9]+|Conclus
968	Dickens/968-martin_chuzzlewit.txt	martin chuzzlewit	^(PREFACE|CHAPTER\s[A-Z]+[-]?[A-Z
1023	Dickens/1023-bleak_house.txt	bleak house	^\s*(PREFACE|CHAPTER\s*[IVXLCM
1289	Dickens/1289-three_ghost_stories.txt	three ghost stories	THE HAUNTED HOUSE\.|THE TRIAL MURDER\.|T
1394	Dickens/1394-the_holly_tree.txt	the holly tree	^[A-Z]+\sBRA
1400	Dickens/1400-great_expectations.txt	great expectations	^\s*Chapter\s*[IVXL
1406	Dickens/1406-the_perils_of_certain_english_pri...	the perils of certain english prisoners	^CHAPTER\s[IVXL
1407	Dickens/1407-a_message_from_the_sea.txt	a message from the sea	^CHAPTER\s[IVXL
1413	Dickens/1413-tom_tiddlers_ground.txt	tom tiddlers ground	^CHAPTER\s[IVXL
1414	Dickens/1414-somebodys_luggage.txt	somebodys luggage	^CHAPTER\s[IVXL
1415	Dickens/1415-doctor_marigold.txt	doctor marigold	* * \
1416	Dickens/1416-mrs_lirripers_lodgings.txt	mrs lirripers lodgings	^CHAPTER\s[IVXL
1421	Dickens/1421-mrs_lirripers_legacy.txt	mrs lirripers legacy	^CHAPTER\s[IVXL
1435	Dickens/1435-miscellaneous_papers.txt	miscellaneous papers	THE AGRICULT INTEREST$|THREATENING LETT
1467	Dickens/1467-some_christmas_stories.txt	some christmas stories	A CHRISTMAS TREE[\.]?|WHAT CHRIST IS AS \
2324	Dickens/2324-a_house_to_let.txt	a house to let	OVER THE |*THEMANCHESTERMARRI.* |GOIN
19337	Dickens/19337-a_christmas_carol.txt	a christmas carol	^\s*STAVE\s[A-
20795	Dickens/20795-the_cricket_on_the_hearth.txt	the cricket on the hearth	^CHIRP\

book_id	source_file_path	title	chap_r
27924	Dickens/27924-mugby_junction.txt	mugby junction	BARBOX BROTI |*BARBOXBROTHERSAND(* |I
35536	Dickens/35536-the_poems_and_verses_of_charles_...	the poems and verses of charles dickens	THE VILLAGE COQUE |*THELAMPLIGHTER*|SON

In [84]:
```
# df with NLTK's English stopwords
stopwords = pd.DataFrame(nltk.corpus.stopwords.words('english'), columns = ['ter

# make term the index and previous (numeric) index a column
stopwords = stopwords.reset_index().set_index('term_str')

# replace index col with dummy col of 1's
stopwords.columns = ['dummy']
stopwords.dummy = 1
```

In [85]:
```
def create_vocab(corpus, i = CORPUS.index.get_level_values(0).unique()):

    # subset corpus to include only defined book(s) (default is to include all o
    corpus = corpus.loc[i]

    # create term table
    vocab = corpus.term_str.value_counts().to_frame('n').sort_index()

    # rename index
    vocab.index.name = 'term_str'

    # number of characters in each term
    vocab['n_chars'] = vocab.index.str.len()

    # probability of term
    vocab['p'] = vocab.n / vocab.n.sum()

    # log2 prob of term
    vocab['i'] = - np.log2(vocab.p)

    # most common POS associated with term
    vocab['max_pos'] = corpus[['term_str', 'pos']].value_counts().unstack(fill_v

    # term, POS matrix
    TPM = corpus[['term_str', 'pos']].value_counts().unstack()

    # col with number of non-NA cells for each row (i.e., along the columns) = n
    vocab['n_pos'] = TPM.count(axis = 1)

    vocab['cat_pos'] = corpus[['term_str', 'pos']].value_counts().to_frame('n').
                   .groupby('term_str').pos.apply(lambda x: set(x))

    # map stopwords dummy col to VOCAB df based on shared index
    vocab['stop'] = vocab.index.map(stopwords.dummy)
```

```
    # fill non-stopword rows with value 0 in stop col
    vocab['stop'] = vocab['stop'].fillna(0).astype('int')

    # Porter stemmer
    stemmer1 = PorterStemmer()
    vocab['stem_porter'] = vocab.apply(lambda x: stemmer1.stem(x.name), 1)

    # Snowball stemmer
    stemmer2 = SnowballStemmer("english")
    vocab['stem_snowball'] = vocab.apply(lambda x: stemmer2.stem(x.name), 1)

    # Lancaster stemmer
    stemmer3 = LancasterStemmer()
    vocab['stem_lancaster'] = vocab.apply(lambda x: stemmer3.stem(x.name), 1)

    return vocab
```

In [86]:
```
VOCAB = create_vocab(CORPUS)
```

In [87]:
```
VOCAB
```

Out[87]:

term_str	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	stem_porter	st
0	60	1	1.207251e-05	16.337915	CD	4	{RB, CD, NN, JJ}	0	0	
1	38	1	7.645923e-06	16.996878	CD	5	{NNP, CD, VB, NN, JJ}	0	1	
10	8	2	1.609668e-06	19.244805	CD	4	{NNP, IN, CD, NN}	0	10	
100	4	3	8.048340e-07	20.244805	CD	4	{JJ, IN, CD, NN}	0	100	
1000	1	4	2.012085e-07	22.244805	JJ	1	{JJ}	0	1000	
...	...	...	...	...	...	...	...	...	...	
æolian	2	6	4.024170e-07	21.244805	JJ	1	{JJ}	0	æolian	
æsop	1	4	2.012085e-07	22.244805	NN	1	{NN}	0	æsop	
éclat	1	5	2.012085e-07	22.244805	NN	1	{NN}	0	éclat	
élite	1	5	2.012085e-07	22.244805	NN	1	{NN}	0	élite	
ĕngine	1	6	2.012085e-07	22.244805	NNP	1	{NNP}	0	ĕngine	

55272 rows × 11 columns

In [88]:
```
prefix = "dickens_pre_"

LIB.to_csv(f'{prefix}LIB.csv')
CORPUS.to_csv(f'{prefix}CORPUS.csv')
VOCAB.to_csv(f'{prefix}VOCAB.csv')
```

In [ ]:

In [ ]: