Mark Twain 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]:
```
sns.set()
```

In [3]:
```
headers = {'user-agent': 'UVA example (cew4pf@virginia.edu)'}
r = requests.get("https://www.gutenberg.org/files/28803/28803-h/28803-h.htm#link
r
```

Out[3]:
```
<Response [200]>
```

In [4]:
```
index = BeautifulSoup(r.text, 'html')
```

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]
```

In [7]:
```
# regex roman numeral pattern
roman = '[IVXLCM]+'
```

Preprocessing

Renaming Files

book_id	title
70-0	What is Man? And Other Stories
74-0	The Adventures of Tom Sawyer
76-0	The Adventures of Huckleberry Finn
86-0	A Connecticut Yankee in King Arthur's Court
91-0	Tom Sawyer Abroad
93-0	Tom Sawyer, Detective
102-0	The Tragedy of Pudd'nhead Wilson
119-0	A Tramp Abroad
142-0	The $30,000 Bequest and Other Stories
245-0	Life On The Mississippi
1044-0	Extract from Captain Stormfield's Visit to Heaven
1837-0	The Prince and the Pauper
2874-0	Personal Recollections of Joan of Arc - Vol. 1
2875-0	Personal Recollections of Joan of Arc - Vol. 2
2895-0	Following the Equator
3171-0	In Defense of Harriet Shelley
3172-0	Fenimore Cooper's Literary Offences
3173-0	Essays on Paul Bourget
3176-0	The Innocents Abroad
3178-0	The Gilded Age

book_id	title
3179-0	The American Claimant
3180-0	A Double Barrelled Detective Story
3181-0	The Stolen White Elephant
3182-0	Some Rambling Notes of an Idle Excursion
3183-0	The Facts Concerning The Recent Carnival Of Crime In Connecticut
3184-0	Alonzo Fitz and Other Stories
3185-0	Those Extraordinary Twins
3186-0	The Mysterious Stranger and Other Stories
3188-0	Mark Twain's Speeches
3189-0	Sketches New and Old
3190-0	1601 -- Conversation as it was by the Social Fireside in the the Time of the Tudors
3191-0	Goldsmith's Friend Abroad Again
3192-0	The Curious Republic of Gondour and Other Whimsical Sketches
3199-0	The Letters of Mark Twain
3250-0	How to Tell a Story and Other Essays
3251-0	The Man That Corrupted Hadleyburg and Other Stories
33077-0	The Treaty With China, its Provisions Explained
60900-0	Merry Tales
61522-0	The £1,000,000 Bank-Note and Other New Stories
62636-0	To The Person Sitting in Darkness
62739-0	King Leopold's Soliloquy
pg1086	A Horse's Tale
pg3177	Roughing It
pg19484	Editorial Wild Oats
pg19987	Chapters from My Autobiography

In [8]:
```
# os.chdir('Twain')
```

In [9]:
```
# !mv 70-0.txt 70-what_is_man.txt
# !mv 74-0.txt 74-the_adventures_of_tom_sawyer.txt
# !mv 76-0.txt 76-the_adventures_of_huckleberry_finn.txt
# !mv 86-0.txt 86-a_connecticut_yankee_in_king_arthurs_court.txt
# !mv 91-0.txt 91-tom_sawyer_abroad.txt
# !mv 93-0.txt 93-tom_sawyer_detective.txt
# !mv 102-0.txt 102-the_tragedy_of_puddnhead_wilson.txt
# !mv 119-0.txt 119-a_tramp_abroad.txt
# !mv 142-0.txt 142-the_30000_bequest_and_other_stories.txt
# !mv 245-0.txt 245-life_on_the_mississippi.txt
# !mv 1044-0.txt 1044-extract_from_captain_stormfields_visit_to_Heaven.txt
# !mv 1837-0.txt 1837-the_prince_and_the_pauper.txt
```

```
# !mv 2874-0.txt 2874-personal_recollections_of_joan_of_arc_vol_1.txt
# !mv 2875-0.txt 2875-personal_recollections_of_joan_of_arc_vol_2.txt
# !mv 2895-0.txt 2895-following_the_equator.txt
# !mv 3171-0.txt 3171-in_defense_of_harriet_shelley.txt
# !mv 3172-0.txt 3172-fenimore_coopers_literary_offences.txt
# !mv 3173-0.txt 3173-essays_on_paul_bourget.txt
# !mv 3176-0.txt 3176-the_innocents_abroad.txt
# !mv 3178-0.txt 3178-the_gilded_age.txt
# !mv 3179-0.txt 3179-the_american_claimant.txt
# !mv 3180-0.txt 3180-a_double_barrelled_detective_story.txt
# !mv 3181-0.txt 3181-the_stolen_white_elephant.txt
# !mv 3182-0.txt 3182-some_rambling_notes_of_an_idle_excursion.txt
# !mv 3183-0.txt 3183-the_facts_concerning_the_recent_carnival_of_crime_in_conne
# !mv 3184-0.txt 3184-alonzo_fitz_and_other_stories.txt
# !mv 3185-0.txt 3185-those_extraordinary_twins.txt
# !mv 3186-0.txt 3186-the_mysterious_stranger_and_other_stories.txt
# !mv 3188-0.txt 3188-mark_twain_speeches.txt
# !mv 3189-0.txt 3189-sketches_new_and_old.txt
# !mv 3190-0.txt 3190-1601_conversation_as_it_was_by_the_social_fireside_in_the_
# !mv 3191-0.txt 3191-goldsmiths_friend_abroad_again.txt
# !mv 3192-0.txt 3192-the_curious_republic_of_gondour_and_other_whimsical_sketch
# !mv 3199-0.txt 3199-the_letters_of_mark_twain.txt
# !mv 3250-0.txt 3250-how_to_tell_a_story_and_other_essays.txt
# !mv 3251-0.txt 3251-the_man_that_corrupted_hadleyburg_and_other_stories.txt
# !mv 33077-0.txt 33077-the_treaty_with_china_its_provisions_explained.txt
# !mv 60900-0.txt 60900-merry_tales.txt
# !mv 61522-0.txt 61522-the_1000000_bank_note.txt
# !mv 62636-0.txt 62636-to_the_person_sitting_in_darkness.txt
# !mv 62739-0.txt 62739-king_leopolds_soliloquy.txt
# !mv pg1086.txt 1086-a_horses_tale.txt
# !mv pg3177.txt 3177-roughing_it.txt
# !mv pg19484.txt 19484-editorial_wild_oats.txt
# !mv pg19987.txt 19987-chapters_from_my_autobiography.txt
```

In [10]:

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

File *The Man That Corrupted Hadleyburg and Other Stories* (3251) with some material contained in other files (*The Merry Tales* (60900)) so want to delete that duplicated material → remove duplicated material from file and create new file

In [11]:

```
# %%bash

# # create copy of current full The Man Who Corrupted Hadleyburg and Other Stori
# cp Twain/3251-the_man_that_corrupted_hadleyburg_and_other_stories.txt full_325

# # get start line of first section of text to delete (luck, the captain's story
# start=$(grep -n -m2 "LUCK" full_3251-the_man_that_corrupted_hadleyburg_and_oth

# # get end of first section of text to delete
# end=$(grep -n -m2 "This is the captain's own mistake" 3251-the_man_that_corrup

# # get start of second section of text to delete (meisterschaft)
# start2=$(grep -n -m2 "MEISTERSCHAFT" 3251-the_man_that_corrupted_hadleyburg_an

# # get end line of text to delete (last line before "THE END OF THE PROJECT GUT
# end2=$(grep -n "Anybody can do it" 3251-the_man_that_corrupted_hadleyburg_and_
```

```
# # create file where delete material included in other files in CORPUS
# sed -e "${start},${end}d;${start2},${end2}d" full_3251-the_man_that_corrupted_

# # move new file into Twain directory and replace old file
# mv 3251-the_man_that_corrupted_hadleyburg_and_other_stories.txt Twain
```

File *The $30,000 Bequest and Other Stories* (142) with some material contained in other files (*How to Tell a Story and Other Essays* (3250)) so want to delete that duplicated material → remove duplicated material from file and create new file

In [12]:

```
# %%bash

# # create copy of current full The $30,000 Bequest and Other Stories
# cp Twain/142-the_30000_bequest_and_other_stories.txt full_142-the_30000_beques

# # get start line of section of text to delete (how to tell a story)
# start=$(grep -n -m2 "HOW TO TELL A STORY" full_142-the_30000_bequest_and_other

# # get end line of text to delete (beginning of next story)
# end=$( grep -n -m2 "GENERAL WASHINGTON'S NEGRO BODY-SERVANT" full_142-the_3000

# # create file where delete material included in other files in CORPUS
# sed -e "${start},${end}d" full_142-the_30000_bequest_and_other_stories.txt > 1

# # move new file into Twain directory and replace old file
# mv 142-the_30000_bequest_and_other_stories.txt Twain
```

File *Sketches New and Old* (3189) with some material contained in other files (*Mark Twain's Speeches* (3188)) so want to delete that duplicated material → remove duplicated material from file and create new file

In [13]:

```
# %%bash

# # create copy of current full Sketches New and Old
# cp Twain/3189-sketches_new_and_old.txt full_3189-sketches_new_and_old.txt

# # get start line of first section of text to delete (Speech at the Scottish Ba
# start=$(grep -n -m2 "for the rest of the speakers" full_3189-sketches_new_and_

# # get end of first section of text to delete
# end=$(grep -n -m2 "rise to a world of discussion.]" full_3189-sketches_new_and

# # get start of second section of text to delete (Speech on Accident Insurance
# start2=$(grep -n -m2 "SPEECH ON ACCIDENT INSURANCE" full_3189-sketches_new_and

# # get end line of text to delete
# end2=$(grep -n -m2 "for the rest of the speakers" full_3189-sketches_new_and_o

# # create file where delete material included in other files in CORPUS
# sed -e "${start},${end}d;${start2},${end2}d" full_3189-sketches_new_and_old.tx

# # move new file into Twain directory and replace old file
# mv 3189-sketches_new_and_old.txt Twain
```

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)
```

The Innocents Abroad (3176): duplicate chapter headings

In [14]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/3176-the_innocents_abroad.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 [15]:
```
# 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 [16]:
```
# 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 [17]:
```
# 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 [18]:
```
# slice df using index to remove front and back matter

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

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

chap_pat = rf"^\s*(CHAPTER\s*{roman}\.$|CONCLUSION)"
```

In [21]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [22]:
```
LINES.loc[chap_lines]
```

Out[22]:

line_num	line_str
482	CHAPTER I.
785	CHAPTER II.
925	CHAPTER III.
1084	CHAPTER IV.
1340	CHAPTER V.
...	...
17705	CHAPTER LVIII.
18169	CHAPTER LIX.
18284	CHAPTER LX.
18398	CHAPTER LXI.
18639	CONCLUSION.

62 rows × 1 columns

Personal Recollections of Joan of Arc - Vol 1. (2874-0): duplicate chapter headings

In [23]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/2874-personal_recollections_of_joan_of_arc_vol_1.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 [24]:
```
# 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 [25]:
```
# 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 [26]:
```
# 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 [27]:
```
# slice df using index to remove front and back matter

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

In [28]:
```
# 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 [29]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^Chapter\s[0-9]+"
```

In [30]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [31]:
```
LINES.loc[chap_lines]
```

Out[31]:

line_num	line_str
324	Chapter 1 When Wolves Ran Free in Paris
416	Chapter 2 The Fairy Tree of Domremy
930	Chapter 3 All Aflame with Love of France
1198	Chapter 4 Joan Tames the Mad Man
1557	Chapter 5 Domremy Pillaged and Burned
1878	Chapter 6 Joan and Archangel Michael
2146	Chapter 7 She Delivers the Divine Command
2475	Chapter 8 Why the Scorners Relented
2550	Chapter 1 Joan Says Good-By
2638	Chapter 2 The Governor Speeds Joan
2906	Chapter 3 The Paladin Groans and Boasts
3165	Chapter 4 Joan Leads Us Through the Enemy
3492	Chapter 5 We Pierce the Last Ambuscades
3942	Chapter 6 Joan Convinces the King
4222	Chapter 7 Our Paladin in His Glory
4458	Chapter 8 Joan Persuades Her Inquisitors
4765	Chapter 9 She Is Made General-in-Chief
4869	Chapter 10 The Maid's Sword and Banner
5054	Chapter 11 The War March Is Begun
5194	Chapter 12 Joan Puts Heart in Her Army
5430	Chapter 13 Checked by the Folly of the Wise
5686	Chapter 14 What the English Answered
5774	Chapter 15 My Exquisite Poem Goes to Smash
6053	Chapter 16 The Finding of the Dwarf
6416	Chapter 17 Sweet Fruit of Bitter Truth
6519	Chapter 18 Joan's First Battle-Field
6723	Chapter 19 We Burst In Upon Ghosts
6818	Chapter 20 Joan Makes Cowards Brave Victors
7025	Chapter 21 She Gently Reproves Her Dear Friend

line_num	line_str
7237	Chapter 22 The Fate of France Decided
7520	Chapter 23 Joan Inspires the Tawdry King
7822	Chapter 24 Tinsel Trappings of Nobility
7961	Chapter 25 At Last—Forward!
8155	Chapter 26 The Last Doubts Scattered
8293	Chapter 27 How Joan Took Jargeau

Personal Recollections of Joan of Arc - Vol 2. (2875-0): duplicate chapter headings

In [32]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/2875-personal_recollections_of_joan_of_arc_vol_2.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 [33]:
```
# 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 [34]:
```
# 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 [35]:
```
# 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 [36]:
```
# slice df using index to remove front and back matter

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

In [37]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^[0-9]+\s[A-Z]+"
```

In [38]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [39]:
```
LINES.loc[chap_lines]
```

Out[39]:

line_num	line_str
66	28 Joan Foretells Her Doom
68	29 Fierce Talbot Reconsiders
70	30 The Red Field of Patay
72	31 France Begins to Live Again
74	32 The Joyous News Flies Fast
...	...
6737	20 The Betrayal
7074	21 Respited Only for Torture
7176	22 Joan Gives the Fatal Answer
7413	23 The Time Is at Hand
7663	24 Joan the Martyr

76 rows × 1 columns

Following the Equator (2895-0): duplicate chapter headings (such a mess...)

In [40]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/2895-following_the_equator.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 [41]:
```
# 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 [42]:
```
# 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 [43]:
```
# 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 [44]:
```
# slice df using index to remove front and back matter

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

In [45]:
```
# get duplicated lines and keep the last occurrence (i.e., chapter headers withi
chapter_duplicates = LINES.loc[LINES.duplicated(keep = 'last') & LINES.line_str.

conclusion_duplicates = LINES.loc[LINES.duplicated(keep = 'last') & LINES.line_s

#chapter_duplicates = chapter_duplicates.append(conclusion_duplicates)

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

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

chap_pat = rf"^(CHAPTER[,]?\s{roman}|CONCLUSION)\.$"
```

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_str

line_num	line_str
line_num	
71	CHAPTER I.
78	CHAPTER II.
83	CHAPTER III.
89	CHAPTER IV.
96	CHAPTER V.
...	...
17804	CHAPTER LXVI.
18077	CHAPTER LXVII.
18528	CHAPTER LXVIII.
18838	CHAPTER LXIX.
19149	CONCLUSION.

134 rows × 1 columns

Essays on Paul Bourget (3173-0): duplicate chapter headings

In [49]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/3173-essays_on_paul_bourget.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 [50]:
```
# 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 [51]:
```
# 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 [52]:
```
# 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 [53]:
```
# slice df using index to remove front and back matter

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

In [54]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"(WHAT PAUL BOURGET|A LITTLE NOTE TO)"
```

In [55]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [56]:
```
LINES.loc[chap_lines].drop_duplicates(keep = 'last')
```

Out[56]:

line_num	line_str
44	WHAT PAUL BOURGET THINKS OF US
642	A LITTLE NOTE TO M. PAUL BOURGET

The American Claimant (3179-0): duplicate chapter headings

In [57]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/3179-the_american_claimant.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 [58]:
```
# 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 [59]:
```
# 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 [60]:
```
# 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 [61]:
```
# slice df using index to remove front and back matter

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

In [62]:
```
LINES
```

Out[62]:

line_num	line_str
20	
21	
22	
23	
24	Produced by David Widger. Additional proofing ...
...	...
7884	
7885	
7886	End of the Project Gutenberg EBook of The Amer...
7887	Twain (Samuel Clemens)
7888	

7869 rows × 1 columns

In [63]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^(CHAPTER\s{roman}|APPENDIX)\.$"
```

In [64]:
```
# Series with boolean values for each line --> True where matches pattern (chapt
```

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

In [65]:
```
LINES.loc[chap_lines].drop_duplicates(keep = 'last')
```

Out[65]:

line_num	line_str
298	CHAPTER I.
546	CHAPTER II.
829	CHAPTER III.
1269	CHAPTER IV.
1480	CHAPTER V.
1735	CHAPTER VI.
1885	CHAPTER VII.
2024	CHAPTER VIII.
2279	CHAPTER IX.
2593	CHAPTER X.
2859	CHAPTER XI.
3269	CHAPTER XII.
3581	CHAPTER XIII.
3949	CHAPTER XIV.
4220	CHAPTER XV.
4536	CHAPTER XVI.
4807	CHAPTER XVII.
5129	CHAPTER XVIII.
5425	CHAPTER XIX.
5737	CHAPTER XX.
5946	CHAPTER XXI.
6380	CHAPTER XXII.
6691	CHAPTER XXIII.
7050	CHAPTER XXIV.
7451	CHAPTER XXV.
7793	APPENDIX.

Alonzo Fitz and Other Stories (3184-0): encoding and duplicate chapter headings

In [66]:
```
# encoding argument for open() to strip out character associated with Project Gu
```

```
text_file = 'Twain/3184-alonzo_fitz_and_other_stories.txt'

# read lines of text file and convert to dataframe
LINES = pd.DataFrame(open(text_file, 'r', encoding='latin-1').readlines(), colum

# 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 [67]:
```
# 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 [68]:
```
# 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 [69]:
```
# 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 [70]:
```
# slice df using index to remove front and back matter

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

In [71]:
```
LINES
```

Out[71]:

line_num	line_str
23	
24	
25	
26	Produced by David Widger
27	

line_num	line_str
...	...
3380	
3381	
3382	End of the Project Gutenberg EBook of Alonzo F...
3383	by Mark Twain (Samuel Clemens)
3384	

3362 rows × 1 columns

In [72]:
```
# capture chapter (speech) names (and remove whitespace, replace " with differen
alonzo_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).replac
                    for i in index.find_all('a', {'href': re.compile(r'https://w

# add -- for one story title that contains those extra characters for in-text ti
alonzo_chap_pats = [i + ' --' if 'CONCERNING' in i else i for i in alonzo_chap_p

alonzo_chap_pat = '|'.join(alonzo_chap_pats)

alonzo_chap_pat = rf'{alonzo_chap_pat}$'
```

In [73]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [74]:
```
LINES.loc[chap_lines].drop_duplicates(keep = 'last')
```

Out[74]:

line_num	line_str
72	THE LOVES OF ALONZO FITZ CLARENCE AND ROSANNAH...
947	ON THE DECAY OF THE ART OF LYING
1157	ABOUT MAGNANIMOUS-INCIDENT LITERATURE
1399	PUNCH, BROTHERS, PUNCH
1615	THE GREAT REVOLUTION IN PITCAIRN
2059	THE CANVASSER'S TALE
2282	AN ENCOUNTER WITH AN INTERVIEWER
2512	PARIS NOTES
2612	LEGEND OF SAGENFELD, IN GERMANY
2847	SPEECH ON THE BABIES
2947	SPEECH ON THE WEATHER

line_num	line_str
3058	CONCERNING THE AMERICAN LANGUAGE --
3183	ROGERS

Those Extraordinary Twins (3185-0): encoding

In [75]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/3185-those_extraordinary_twins.txt'

# read lines of text file and convert to dataframe
LINES = pd.DataFrame(open(text_file, 'r', encoding='latin-1').readlines(), colum

# 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 [76]:
```
# 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 [77]:
```
# 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 [78]:
```
# 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 [79]:
```
# slice df using index to remove front and back matter

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

In [80]:
```
LINES
```

Out[80]:

line_num	line_str
23	
24	
25	
26	Produced by David Widger
27	
...	...
2624	
2625	
2626	End of the Project Gutenberg EBook of Those Ex...
2627	Twain (Samuel Clemens)
2628	

2606 rows × 1 columns

In [81]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^CHAPTER\s{roman}\.\s[A-Z]+"
```

In [82]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [83]:
```
LINES.loc[chap_lines].drop_duplicates(keep = 'last')
```

Out[83]:

line_num	line_str
194	CHAPTER I. THE TWINS AS THEY REALLY WERE
500	CHAPTER II. MA COOPER GETS ALL MIXED UP
817	CHAPTER III. ANGELO IS BLUE
927	CHAPTER IV. SUPERNATURAL CHRONOMETRY
1246	CHAPTER V. GUILT AND INNOCENCE FINELY BLENT
1888	CHAPTER VI. THE AMAZING DUEL
2102	CHAPTER VII. LUIGI DEFIES GALEN
2350	CHAPTER VIII. BAPTISM OF THE BETTER HALF
2465	CHAPTER IX. THE DRINKLESS DRUNK
2542	CHAPTER X. SO THEY HANGED LUIGI

Goldsmith's Friend Abroad Again (3191-0): duplicate chapter headings

In [84]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/3191-goldsmiths_friend_abroad_again.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 [85]:
```
# 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 [86]:
```
# 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 [87]:
```
# 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 [88]:
```
# slice df using index to remove front and back matter

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

In [89]:
```
LINES
```

Out[89]:

line_num	line_str
24	
25	

line_num	line_str
26	
27	Produced by David Widger
28	
...	...
271	
272	
273	End of the Project Gutenberg EBook of Goldsmit...
274	by Mark Twain (Samuel Clemens)
275	

252 rows × 1 columns

In [90]:
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"LETTER\s{roman}"
```

In [91]:
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [92]:
```
LINES.loc[chap_lines].drop_duplicates(keep = 'last')
```

Out[92]:

line_num	line_str
71	LETTER I
84	LETTER II
107	LETTER III
122	LETTER IV
165	LETTER V
182	LETTER VI
225	LETTER VII

The Curious Republic of Gondour and Other Whimsical Sketches (3192-0): duplicate chapter headings

In [93]:
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/3192-the_curious_republic_of_gondour_and_other_whimsical_sket
```

```
# 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 [94]:
```
# 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 [95]:
```
# 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 [96]:
```
# 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 [97]:
```
# slice df using index to remove front and back matter

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

In [98]:
```
LINES
```

Out[98]:

line_num	line_str
24	
25	
26	
27	Produced by David Widger
28	
...	...

line_num	line_str
820	
821	
822	End of the Project Gutenberg EBook of The Curi...
823	Other Whimsical Sketches, by Mark Twain (Samue...
824	

801 rows × 1 columns

In [99]:
```
# capture chapter (speech) names (and remove whitespace, replace " with differen
gondour_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).repla
                     for i in index.find_all('a', {'href': re.compile(r'https://

gondour_chap_pat = '|'.join(gondour_chap_pats)

gondour_chap_pat = rf'{gondour_chap_pat}$'
```

In [100…
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"LETTER\s{roman}"
```

In [101…
```
# Series with boolean values for each line --> True where matches pattern (chapt

chap_lines = LINES.line_str.str.match(gondour_chap_pat, case=True)
```

In [102…
```
LINES.loc[chap_lines].drop_duplicates(keep = 'last')
```

Out[102…

line_num	line_str
36	THE CURIOUS REPUBLIC OF GONDOUR AND OTHER WHIM...
94	THE CURIOUS REPUBLIC OF GONDOUR
156	A MEMORY
264	INTRODUCTORY TO “MEMORANDA”
296	ABOUT SMELLS
330	A COUPLE OF SAD EXPERIENCES
340	DAN MURPHY
350	THE “TOURNAMENT” IN A. D. 1870
377	CURIOUS RELIC FOR SALE
419	A REMINISCENCE OF THE BACK SETTLEMENTS

line_num	line_str
429	A ROYAL COMPLIMENT
475	THE APPROACHING EPIDEMIC
501	THE TONE-IMPARTING COMMITTEE
509	OUR PRECIOUS LUNATIC

Identifiers and Regexes

Specific Chapter Patterns

In [103…

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

Chapter names for What is Man? and Other Stories (70-0) scraped from Project Gutenberg Index of Mark Twain works

In [104…

```
# capture chapter (speech) names (and remove whitespace, replace " with differen
man_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).replace('
                     for i in index.find_all('a', {'href': re.compile(r'https:/


man_chap_pat = '|'.join(man_chap_pats)

man_chap_pat = rf'{man_chap_pat}$'
```

Chapter names for The $30,000 Bequest and Other Stories (142-0) scraped from Project Gutenberg Index of Mark Twain works

In [105…

```
# capture chapter (speech) names (and remove whitespace, replace " with differen
beq_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).replace('
                     for i in index.find_all('a', {'href': re.compile(r'https:/

# use to escape $ in regex for story with $ in title
beq_chap_pats = [i[:i.index('$')] + '\\' + i[i.index('$'):] if '$' in i else i f

# remove names of stories taken out since duplicates in other works
beq_chap_pats = [i for i in beq_chap_pats if re.search('(HOW TO TELL A|GENERAL W


beq_chap_pat = '$|'.join(beq_chap_pats)

beq_chap_pat = rf'{beq_chap_pat}$'
```

Chapter names for Alonzo Fitz and Other Stories (3184-0) scraped from Project Gutenberg Index of Mark Twain works

In [106…
```
# capture chapter (speech) names (and remove whitespace, replace " with differen
alonzo_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).replac
                    for i in index.find_all('a', {'href': re.compile(r'https://w

# add -- for one story title that contains those extra characters for in-text ti
alonzo_chap_pats = [i + ' --' if 'CONCERNING' in i else i for i in alonzo_chap_p

alonzo_chap_pat = '|'.join(alonzo_chap_pats)

alonzo_chap_pat = rf'{alonzo_chap_pat}$'
```

Chapter names for Mark Twain's Speeches (3188-0) scraped from Project Gutenberg Index of Mark Twain works

In [107…
```
# capture chapter (speech) names (and remove whitespace, replace " with differen
speeches_chap_pats = [i.text.strip().replace('\\ ', '').replace('"', '“', 1).rep
                      for i in index.find_all('a', {'href': re.compile(r'https:/

# add this for one speech since chapter name in text different for this speech
speeches_chap_pats = [i + ' \[THE HORRORS OF THE GERMAN LANGUAGE\]' if 'SCHRECKE

speeches_chap_pat = '$|'.join(speeches_chap_pats)

speeches_chap_pat = rf'{speeches_chap_pat}$'
```

Chapter names for Sketches New and Old (3189-0) scraped from Project Gutenberg Index of Mark Twain works

In [108…
```
# capture chapter (speech) names (and remove whitespace, replace " with differen
sketch_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).replac
                    for i in index.find_all('a', {'href': re.compile(r'https://w

sketch_chap_pat = '|'.join(sketch_chap_pats)

sketch_chap_pat = rf'{sketch_chap_pat}$'
```

Chapter names for The Curious Republic of Gondour and Other Whimsical Sketches (3192-0) scraped from Project Gutenberg Index of Mark Twain works

In [109…
```
# capture chapter (speech) names (and remove whitespace, replace " with differen
gondour_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).repla
                     for i in index.find_all('a', {'href': re.compile(r'https://


gondour_chap_pat = '|'.join(gondour_chap_pats)

gondour_chap_pat = rf'{gondour_chap_pat}$'
```

Chapter names for How to Tell a Story and Others (3250-0) scraped from Project Gutenberg Index of Mark Twain works

In [110…
```
# capture chapter (speech) names (and remove whitespace, replace " with differen
story_chap_pats = [i.text.strip().replace('\\', '').replace('"', '“', 1).replace
```

```
                    for i in index.find_all('a', {'href': re.compile(r'https://ww

story_chap_pat = '$|'.join(story_chap_pats)

story_chap_pat = rf'{story_chap_pat}$'
```

In [111…

```
story_chap_pat
```

Out[111…

```
"HOW TO TELL A STORY$|THE WOUNDED SOLDIER.$|THE GOLDEN ARM.$|MENTAL TELEGRAPHY A
GAIN$|THE INVALID'S STORY$"
```

Chapter names for The Man That Corrupted Hadleyburg and Other Stories (3251-0) NOT scraped from Project Gutenberg Index of Mark Twain works

In [112…

```
hadleyburg_chap_pats = ['^THE MAN THAT CORRUPTED HADLEYBURG$',
                        '^MY FIRST LIE, AND HOW I GOT OUT OF IT$',
                        '^THE ESQUIMAUX MAIDEN\'S ROMANCE$',
                        '^CHRISTIAN SCIENCE AND THE BOOK OF MRS\. EDDY$',
                        '^IS HE LIVING OR IS HE DEAD\?$',
                        '^MY DEBUT AS A LITERARY PERSON$',
                        '^AT THE APPETITE-CURE$',
                        '^CONCERNING THE JEWS$',
                        '^FROM THE \'LONDON TIMES\' OF 1904$',
                        '^ABOUT PLAY-ACTING$',
                        '^TRAVELLING WITH A REFORMER$',
                        '^STIRRING TIMES IN AUSTRIA$',
                        '^MY BOYHOOD DREAMS$',
                        '^TO THE ABOVE OLD PEOPLE$',
                        '^IN MEMORIAM$'

]

hadleyburg_chap_pat = '|'.join(hadleyburg_chap_pats)

hadleyburg_chap_pat = rf'{hadleyburg_chap_pat}'
```

Chapter names for Merry Tales (60900-0) but NOT scraped from Project Gutenberg Index of Mark Twain works since not included there for some reason

In [113…

```
merry_chap_pats = ['^THE PRIVATE HISTORY OF A CAMPAIGN THAT FAILED\.',
                   '^THE INVALID’S STORY\.',
                   '^LUCK\.',
                   '^THE CAPTAIN\'S STORY\.',
                   '^A CURIOUS EXPERIENCE\.',
                   '^MRS\. McWILLIAMS AND THE LIGHTNING\.',
                   'MEISTERSCHAFT:'
]

merry_chap_pat = '|'.join(merry_chap_pats)

merry_chap_pat = rf'{merry_chap_pat}'
```

Chapter names for THE £1,000,000 BANK-NOTE (61522-0) but NOT scraped

from Project Gutenberg Index of Mark Twain works since not included there for some reason

In [114…

```
bank_chap_pats = ['^_THE £1,000,000 BANK-NOTE_$',
                  '^_METNAL TELEGRAPHY_$',
                  '^_A CURE FOR THE BLUES_$',
                  '^_THE ENEMY CONQUERED; OR, LOVE TRIUMPHANT_$',
                  '^_ALMOST ALL KINDS OF SHIPS_$',
                  '^_ABOUT ALL KINDS OF SHIPS_$',
                  '^_PLAYING COURIER_$',
                  '^_THE GERMAN CHICAGO_$'
]

bank_chap_pat = '|'.join(bank_chap_pats)

bank_chap_pat = rf'{bank_chap_pat}'
```

Chapter names for Editorial Wild Oats (pg19484) but NOT scraped from Project Gutenberg Index of Mark Twain works since not included there for some reason

In [115…

```
oats_chap_pats = ['^My First Literary Venture$',
                  '^Journalism in Tennessee$',
                  '^Nicodemus Dodge--Printer$',
                  '^Mr\. Bloke\'s Item$',
                  '^How I Edited an Agricultural Paper$',
                  '^The Killing of Julius'
]

oats_chap_pat = '|'.join(oats_chap_pats)

oats_chap_pat = rf'{oats_chap_pat}$'
```

In [116…

```
# All are 'chap'and 'm' (milestone)
ohco_pat_list = [
    (70, man_chap_pat, False),
    (74, rf"^\s*CHAPTER\s*{roman}$", False),
    (76, rf"^\s*CHAPTER\s*(?:{roman}\.|THE LAST)$", True),
    (86, rf"^\s*(?:PREFACE|A WORD OF EXPLANATION|THE STRANGER'S HISTORY|CHAPTER\
    (91, rf"CHAPTER\s{roman}\.", False),
    (93, rf"^CHAPTER\s{roman}\.\s[A-Z]", True),
    (102, rf"^(?:A Whisper|CHAPTER\s{roman}\.|CONCLUSION)$", True),
    (119, rf"^(?:CHAPTER\s{roman}|APPENDIX\s[A-Z]\.)$", True),
    (142, beq_chap_pat, True),
    (245, rf"^(THE 'BODY OF THE NATION'|CHAPTER\s[0-9]+|APPENDIX\s[A-Z])$", Fals
    (1044, rf"CHAPTER\s{roman}$", False),
    (1086, rf"^{roman}$", False),
    (1837, rf"^\s*CHAPTER\s*{roman}", False),
    (2874, rf"^Chapter\s[0-9]+", True),
    (2875, rf"^[0-9]+\s[A-Z]+", True),
    (2895, rf"^(CHAPTER[,]?\s{roman}|CONCLUSION)\.$", True),
    (3171, rf"^{roman}$", False),
    (3172, rf"The Pathfinder and The Deerslayer", False), # no chapters so use r
    (3173, rf"(WHAT PAUL BOURGET|A LITTLE NOTE TO)", True),
    (3176, rf"^\s*(CHAPTER\s*{roman}\.$|CONCLUSION)", True),
    (3177, rf"^(CHAPTER\s{roman}|APPENDIX)\.$", False),
```

```
    (3178, rf"^(CHAPTER\s{roman}|APPENDIX)\.$", False),
    (3179, rf"^(CHAPTER\s{roman}|APPENDIX)\.$", True),
    (3180, rf"^{roman}[\.]?$", False),
    (3181, rf"^{roman}[\.]?$", False),
    (3182, rf"^{roman}\.$", False),
    (3183, rf"I was feeling blithe", False), # no chapters so use regex for firs
    (3184, alonzo_chap_pat, True),
    (3185, rf"^CHAPTER\s{roman}\.\s[A-Z]+", False),
    (3186, rf"^(Chapter\s[0-9]+|A FABLE|HUNTING THE DECEITFUL TURKEY|THE McWILLI
    (3188, speeches_chap_pat, True),
    (3189, sketch_chap_pat, False),
    (3190, rf"^(INTRODUCTION|THE FIRST PRINTING|FOOTNOTES|PARTIAL BIBLIOGRAPHY)"
    (3191, rf"LETTER\s{roman}", True),
    (3192, gondour_chap_pat, True),
    (3199, rf"^{roman}\. [A-Z]+\s", False),
    (3250, story_chap_pat, True),
    (3251, hadleyburg_chap_pat, True),
    (19484, oats_chap_pat, False),
    (19987, rf"^(INTRODUCTION|{roman}|CHAPTERS FROM MY AUTOBIOGRAPHY\.--{roman})
    (33077, rf"^New York Tribune", False), # no chapters so use regex for first
    (60900, merry_chap_pat, False),
    (61522, bank_chap_pat, False),
    (62636, rf"^Extending the Blessings", False), # no chapters so use regex for
    (62739, rf"^(\[_Throws down pamphlets which he has|Footnote)", False)
]
```

In [117…
```
source_files = f'Twain'
```

In [118…
```
source_file_list = sorted(glob(f"{source_files}/*.*"))
```

In [119…
```
len(source_file_list)
```

Out[119… 45

LIB, CORPUS, and VOCAB Tables

In [120…
```
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 [121…
```
LIB = pd.DataFrame(book_data, columns=['book_id','source_file_path','title']) \
        .set_index('book_id').sort_index()
```

In [122…
```
LIB
```

Out[122…

book_id	source_file_path	title

book_id	source_file_path	title
70	Twain/70-what_is_man.txt	what is man
74	Twain/74-the_adventures_of_tom_sawyer.txt	the adventures of tom sawyer
76	Twain/76-the_adventures_of_huckleberry_finn.txt	the adventures of huckleberry finn
86	Twain/86-a_connecticut_yankee_in_king_arthurs_...	a connecticut yankee in king arthurs court
91	Twain/91-tom_sawyer_abroad.txt	tom sawyer abroad
93	Twain/93-tom_sawyer_detective.txt	tom sawyer detective
102	Twain/102-the_tragedy_of_puddnhead_wilson.txt	the tragedy of puddnhead wilson
119	Twain/119-a_tramp_abroad.txt	a tramp abroad
142	Twain/142-the_30000_bequest_and_other_stories.txt	the 30000 bequest and other stories
245	Twain/245-life_on_the_mississippi.txt	life on the mississippi
1044	Twain/1044-extract_from_captain_stormfields_vi...	extract from captain stormfields visit to Heaven
1086	Twain/1086-a_horses_tale.txt	a horses tale
1837	Twain/1837-the_prince_and_the_pauper.txt	the prince and the pauper
2874	Twain/2874-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 1
2875	Twain/2875-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 2
2895	Twain/2895-following_the_equator.txt	following the equator
3171	Twain/3171-in_defense_of_harriet_shelley.txt	in defense of harriet shelley
3172	Twain/3172-fenimore_coopers_literary_offences.txt	fenimore coopers literary offences
3173	Twain/3173-essays_on_paul_bourget.txt	essays on paul bourget
3176	Twain/3176-the_innocents_abroad.txt	the innocents abroad
3177	Twain/3177-roughing_it.txt	roughing it
3178	Twain/3178-the_gilded_age.txt	the gilded age
3179	Twain/3179-the_american_claimant.txt	the american claimant
3180	Twain/3180-a_double_barrelled_detective_story.txt	a double barrelled detective story
3181	Twain/3181-the_stolen_white_elephant.txt	the stolen white elephant
3182	Twain/3182-some_rambling_notes_of_an_idle_excu...	some rambling notes of an idle excursion
3183	Twain/3183-the_facts_concerning_the_recent_car...	the facts concerning the recent carnival of cr...
3184	Twain/3184-alonzo_fitz_and_other_stories.txt	alonzo fitz and other stories

book_id	source_file_path	title
3185	Twain/3185-those_extraordinary_twins.txt	those extraordinary twins
3186	Twain/3186-the_mysterious_stranger_and_other_s...	the mysterious stranger and other stories
3188	Twain/3188-mark_twain_speeches.txt	mark twain speeches
3189	Twain/3189-sketches_new_and_old.txt	sketches new and old
3190	Twain/3190-1601_conversation_as_it_was_by_the_...	1601 conversation as it was by the social fire...
3191	Twain/3191-goldsmiths_friend_abroad_again.txt	goldsmiths friend abroad again
3192	Twain/3192-the_curious_republic_of_gondour_and...	the curious republic of gondour and other whim...
3199	Twain/3199-the_letters_of_mark_twain.txt	the letters of mark twain
3250	Twain/3250-how_to_tell_a_story_and_other_essay...	how to tell a story and other essays
3251	Twain/3251-the_man_that_corrupted_hadleyburg_a...	the man that corrupted hadleyburg and other st...
19484	Twain/19484-editorial_wild_oats.txt	editorial wild oats
19987	Twain/19987-chapters_from_my_autobiography.txt	chapters from my autobiography
33077	Twain/33077-the_treaty_with_china_its_provisio...	the treaty with china its provisions explained
60900	Twain/60900-merry_tales.txt	merry tales
61522	Twain/61522-the_1000000_bank_note.txt	the 1000000 bank note
62636	Twain/62636-to_the_person_sitting_in_darkness.txt	to the person sitting in darkness
62739	Twain/62739-king_leopolds_soliloquy.txt	king leopolds soliloquy

In [123…
```
LIB['chap_regex'] = LIB.index.map(pd.Series({x[0]:x[1] for x in ohco_pat_list}))
```

In [124…
```
LIB['author'] = 'twain'
```

In [125…
```
# type ids --> dict with keys: types, values: book ids

type_ids =  {'novel': [74, 76, 86, 91, 93, 102, 1837, 3178, 3177, 1086, 3179],
             'stories': [142, 1044, 3180, 3181, 3183, 3184, 3185, 3186, 3189, 31
             'non-fiction': [70, 119, 245, 2874, 2875, 2895, 3171, 3172, 3173, 3
}
```

In [126…
```
# 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 [127…
```
# map to create new col with types for each work
LIB['type'] = LIB.index.map(id_types)
```

In [128…
```
# add year when book published

book_year = ((74, 1876),
             (76, 1884),
             (86, 1889),
             (91, 1894),
             (93, 1896),
             (102, 1894),
             (1837, 1881),
             (3178, 1873),
             (3177, 1872),
             (1086, 1907),
             (3179, 1892),
             (142, 1906),
             (1044, 1909),
             (3180, 1902),
             (3181, 1882),
             (3183, 1877),
             (3184, 1878),
             (3185, 1892),
             (3186, 1916),
             (3189, 1916),
             (3190, 1880),
             (3191, 1870),
             (3192, 1919),
             (3251, 1900),
             (60900, 1892),
             (61522, 1893),
             (62739, 1905),
             (19484, 1875),
             (245, 1883),
             (2874, 1896),
             (2875, 1896),
             (2895, 1897),
             (3171, 1918),
             (3172, 1895),
             (3173, 1890),
             (3176, 1869),
             (3182, 1877),
             (3188, 1880),
             (3199, 1853),
             (3250, 1897),
             (33077, 1868),
             (62636, 1901),
             (19987, 1906),
             (119, 1880),
             (70, 1906)
)
```

In [129…
```
LIB['year'] = LIB.index.map(pd.Series({x[0]: x[1] for x in book_year}))
```

In [130…
```
bins = [1850, 1860, 1870, 1880, 1890, 1900, 1910, 1920]
```

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

In [131…

```
LIB.head()
```

Out[131…

book_id	source_file_path	title	chap_regex	author	type
70	Twain/70-what_is_man.txt	what is man	WHAT IS MAN?|THE DEATH OF JEAN|THE TURNING-POI...	twain	non-fiction
74	Twain/74-the_adventures_of_tom_sawyer.txt	the adventures of tom sawyer	^\s*CHAPTER\s*[IVXLCM]+$	twain	novel
76	Twain/76-the_adventures_of_huckleberry_finn.txt	the adventures of huckleberry finn	^\s*CHAPTER\s*(?:[IVXLCM]+\.|THE LAST)$	twain	novel
86	Twain/86-a_connecticut_yankee_in_king_arthurs_...	a connecticut yankee in king arthurs court	^\s*(?:PREFACE|A WORD OF EXPLANATION|THE STRAN...	twain	novel
91	Twain/91-tom_sawyer_abroad.txt	tom sawyer abroad	CHAPTER\s[IVXLCM]+\.	twain	novel

In [132…

```
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
    try:
        text.import_source().parse_tokens();
    except:
        text.import_source(char_encoding = 'latin-1').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 70 what is man
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 74 the adventures of tom sawyer
line_str chap_str
```

```
Index(['chap_str'], dtype='object')
Tokenizing 76 the adventures of huckleberry finn
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 86 a connecticut yankee in king arthurs court
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 91 tom sawyer abroad
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 93 tom sawyer detective
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 102 the tragedy of puddnhead wilson
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 119 a tramp abroad
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 142 the 30000 bequest and other stories
/Users/cecilyestherwolfe/Desktop/Spring_2022/DS5001/Project/textparser.py:133: U
serWarning: This pattern has match groups. To actually get the groups, use str.e
xtract.
  chap_duplicates = self.TOKENS.loc[self.TOKENS.duplicated(keep = 'last') & sel
f.TOKENS.line_str.str.contains(div_pat, case = False)].index.values
/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 245 life on the mississippi
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1044 extract from captain stormfields visit to Heaven
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1086 a horses tale
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 1837 the prince and the pauper
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 2874 personal recollections of joan of arc vol 1
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 2875 personal recollections of joan of arc vol 2
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 2895 following the equator
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3171 in defense of harriet shelley
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3172 fenimore coopers literary offences
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3173 essays on paul bourget
```

```
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3176 the innocents abroad
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3177 roughing it
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3178 the gilded age
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3179 the american claimant
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3180 a double barrelled detective story
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3181 the stolen white elephant
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3182 some rambling notes of an idle excursion
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3183 the facts concerning the recent carnival of crime in connecticut
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3184 alonzo fitz and other stories
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3185 those extraordinary twins
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3186 the mysterious stranger and other stories
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3188 mark twain speeches
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3189 sketches new and old
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3190 1601 conversation as it was by the social fireside in the time o
f the tudors
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3191 goldsmiths friend abroad again
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3192 the curious republic of gondour and other whimsical sketches
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3199 the letters of mark twain
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3250 how to tell a story and other essays
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 3251 the man that corrupted hadleyburg and other stories
line_str chap_str
Index(['chap_str'], dtype='object')
```

```
Tokenizing 19484 editorial wild oats
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 19987 chapters from my autobiography
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 33077 the treaty with china its provisions explained
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 60900 merry tales
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 61522 the 1000000 bank note
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 62636 to the person sitting in darkness
line_str chap_str
Index(['chap_str'], dtype='object')
Tokenizing 62739 king leopolds soliloquy
line_str chap_str
Index(['chap_str'], dtype='object')
```

In [133…

```
CORPUS = pd.concat(books).sort_index()
```

In [134…

```
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 [135…

```
CORPUS
```

Out[135…

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
70	1	1	0	0	(By, IN)	IN	By	by
				1	(Mark, NNP)	NNP	Mark	mark
				2	(Twain, NNP)	NNP	Twain	twain
		2	0	0	((Samuel, JJ)	JJ	(Samuel	samuel
				1	(Langhorne, NNP)	NNP	Langhorne	langhorne
...	...	...	...	...	...	...	...	...
62739	6	13	0	8	(Leopold's, NNP)	NNP	Leopold's	leopolds
				9	(Soliloquy,, NNP)	NNP	Soliloquy,	soliloquy
				10	(by, IN)	IN	by	by
				11	(Mark, NNP)	NNP	Mark	mark
				12	(Twain, NNP)	NNP	Twain	twain

2970374 rows × 4 columns

In [136…
```
# number of chapters in book
LIB['n_chaps'] = CORPUS.reset_index()[['book_id', 'chap_id']] \
                 .drop_duplicates() \
                 .groupby('book_id').chap_id.count()
```

In [137…
```
# length of each book (number of tokens)
LIB['book_len'] = CORPUS.groupby('book_id').agg({'token_str': 'count'})
```

In [138…
```
LIB
```

Out[138…

book_id	source_file_path	title	ch
70	Twain/70-what_is_man.txt	what is man	WHAT IS MAN?|THE DEATH OF TURN
74	Twain/74-the_adventures_of_tom_sawyer.txt	the adventures of tom sawyer	^\s*CHAPTER\s*[I
76	Twain/76-the_adventures_of_huckleberry_finn.txt	the adventures of huckleberry finn	^\s*CHAPTER\s*(?:[IVXLCM]+\.|TI
86	Twain/86-a_connecticut_yankee_in_king_arthurs_...	a connecticut yankee in king arthurs court	^\s*(?:PREFACE|A WORD OF EXPLAN/
91	Twain/91-tom_sawyer_abroad.txt	tom sawyer abroad	CHAPTER\s[I
93	Twain/93-tom_sawyer_detective.txt	tom sawyer detective	^CHAPTER\s[IVXLCM
102	Twain/102-the_tragedy_of_puddnhead_wilson.txt	the tragedy of puddnhead wilson	Whisper|CHAPTER\s[IVXLCM]+\.|CONC
119	Twain/119-a_tramp_abroad.txt	a tramp abroad	^(?:CHAPTER\s[IVXLCM]+|APPENDIX
142	Twain/142-the_30000_bequest_and_other_stories.txt	the 30000 bequest and other stories	THE $30,000 BEQUEST\|$ADOG'ST$
245	Twain/245-life_on_the_mississippi.txt	life on the mississippi	^(THE 'BODY OF THE NATION'|CHA
1044	Twain/1044-extract_from_captain_stormfields_vi...	extract from captain stormfields visit to Heaven	CHAPTER\s[I
1086	Twain/1086-a_horses_tale.txt	a horses tale	^[I

book_id	source_file_path	title	ch
1837	Twain/1837-the_prince_and_the_pauper.txt	the prince and the pauper	^\s*CHAPTER\s*
2874	Twain/2874-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 1	^Chapt
2875	Twain/2875-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 2	^[0-9
2895	Twain/2895-following_the_equator.txt	following the equator	^(CHAPTER[,]?\s[IVXLCM]+|CONCL
3171	Twain/3171-in_defense_of_harriet_shelley.txt	in defense of harriet shelley	^[I
3172	Twain/3172-fenimore_coopers_literary_offences.txt	fenimore coopers literary offences	The Pathfinder and The
3173	Twain/3173-essays_on_paul_bourget.txt	essays on paul bourget	(WHAT PAUL BOURGET|A LITTLE
3176	Twain/3176-the_innocents_abroad.txt	the innocents abroad	^\s*(CHAPTER\s*[IVXLCM]+\.$|CON
3177	Twain/3177-roughing_it.txt	roughing it	^(CHAPTER\s[IVXLCM]+|APP
3178	Twain/3178-the_gilded_age.txt	the gilded age	^(CHAPTER\s[IVXLCM]+|APP
3179	Twain/3179-the_american_claimant.txt	the american claimant	^(CHAPTER\s[IVXLCM]+|APP
3180	Twain/3180-a_double_barrelled_detective_story.txt	a double barrelled detective story	^[IVXL
3181	Twain/3181-the_stolen_white_elephant.txt	the stolen white elephant	^[IVXL
3182	Twain/3182-some_rambling_notes_of_an_idle_excu...	some rambling notes of an idle excursion	^[IV
3183	Twain/3183-the_facts_concerning_the_recent_car...	the facts concerning the recent carnival of cr...	I was fe
3184	Twain/3184-alonzo_fitz_and_other_stories.txt	alonzo fitz and other stories	THE LOVES OF ALONZO FITZ CLAR RO
3185	Twain/3185-those_extraordinary_twins.txt	those extraordinary twins	^CHAPTER\s[IVXLCM]-

book_id	source_file_path	title	ch
3186	Twain/3186-the_mysterious_stranger_and_other_s...	the mysterious stranger and other stories	^(Chapter\s[0-9]+|A FABLE|HUM D
3188	Twain/3188-mark_twain_speeches.txt	mark twain speeches	INTRODUCTION|*PREFACE*|THE ST S
3189	Twain/3189-sketches_new_and_old.txt	sketches new and old	MY WATCH|POLITICAL ECONOMY|THE I
3190	Twain/3190-1601_conversation_as_it_was_by_the_...	1601 conversation as it was by the social fire...	^(INTRODUCTION| PRINTING|FOOTN
3191	Twain/3191-goldsmiths_friend_abroad_again.txt	goldsmiths friend abroad again	LETTER\s|
3192	Twain/3192-the_curious_republic_of_gondour_and...	the curious republic of gondour and other whim...	THE CURIOUS REPUBLIC OF GO MEMOR
3199	Twain/3199-the_letters_of_mark_twain.txt	the letters of mark twain	^[IVXLCM]+
3250	Twain/3250-how_to_tell_a_story_and_other_essay...	how to tell a story and other essays	HOW TO TEL| |*THEWOUNDEDSOLDI*
3251	Twain/3251-the_man_that_corrupted_hadleyburg_a...	the man that corrupted hadleyburg and other st...	^THE MAN THAT CO HADLEYBURG$|^M
19484	Twain/19484-editorial_wild_oats.txt	editorial wild oats	^My First Literary Venture$|^Journalis
19987	Twain/19987-chapters_from_my_autobiography.txt	chapters from my autobiography	^(INTRODUCTION|[IVXLCM]+|CHAPT M
33077	Twain/33077-the_treaty_with_china_its_provisio...	the treaty with china its provisions explained	^New Yo
60900	Twain/60900-merry_tales.txt	merry tales	^THE PRIVATE HISTORY OF A CAMPA
61522	Twain/61522-the_1000000_bank_note.txt	the 1000000 bank note	^_THE £1,000,000 BANK-NOTE_$|^ TE
62636	Twain/62636-to_the_person_sitting_in_darkness.txt	to the person sitting in darkness	^Extending the
62739	Twain/62739-king_leopolds_soliloquy.txt	king leopolds soliloquy	^(\[_Throws down pamphlet has

In [139…
```
# 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 [140…
```
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 [141…

```
VOCAB = create_vocab(CORPUS)
```

In [142…

```
VOCAB
```

Out[142…

term_str	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	stem_po
0	5	1	1.683290e-06	19.180285	CD	1	{CD}	0	
00	3	2	1.009974e-06	19.917251	NN	2	{NN, NNS}	0	
01	3	2	1.009974e-06	19.917251	NNS	2	{NN, NNS}	0	
02	4	2	1.346632e-06	19.502213	NN	3	{POS, NN, NNP}	0	
03	6	2	2.019948e-06	18.917251	NN	3	{POS, NN, NNS}	0	
…	…	…	…	…	…	…	…	…	
êtes	1	4	3.366579e-07	21.502213	NNS	1	{NNS}	0	
être	3	4	1.009974e-06	19.917251	NNP	2	{JJ, NNP}	0	
öffnen	1	6	3.366579e-07	21.502213	NN	1	{NN}	0	öf
über	1	4	3.366579e-07	21.502213	NNP	1	{NNP}	0	ü
übergeschlagen	1	14	3.366579e-07	21.502213	NN	1	{NN}	0	übergeschla

53854 rows × 11 columns

In [143…

```
prefix = "twain_pre_"

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

In [ ]:

In [ ]:

In [ ]:

In [144…
```
# encoding argument for open() to strip out character associated with Project Gu

text_file = 'Twain/142-the_30000_bequest_and_other_stories.txt'

# read lines of text file and convert to dataframe
LINES = pd.DataFrame(open(text_file, 'r', encoding='utf-8').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 [145…
```
# 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 [146…
```
# 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 [147…
```
# 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 [148…
```
# slice df using index to remove front and back matter

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

In [149…
```
LINES
```

Out[149…

line_num	line_str
20	
21	

line_num	line_str
22	
23	Produced by An Anonymous Volunteer, and David ...
24	
...	...
9944	
9945	
9946	End of the Project Gutenberg EBook of The $30,...
9947	Stories, by Mark Twain
9948	

9929 rows × 1 columns

In [150…
```
# regex to identify lines in text that act as headers for chapters

chap_pat = rf"^(?:A Whisper|CHAPTER\s{roman}\.|Conclusion)$"
```

In [151…
```
# Series with boolean values for each line --> True where matches pattern (chapt

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

In [152…
```
LINES.loc[chap_lines].drop_duplicates(keep = 'last')
```

Out[152…

line_num	line_str
163	THE $30,000 BEQUEST
1418	A DOG'S TALE
2697	A CURE FOR THE BLUES
3393	THE CURIOUS BOOK
5030	THE CALIFORNIAN'S TALE
5345	A HELPLESS SITUATION
5594	A TELEPHONIC CONVERSATION
5797	EDWARD MILLS AND GEORGE BENTON: A TALE
6018	THE FIVE BOONS OF LIFE
6132	THE FIRST WRITING-MACHINES
6246	ITALIAN WITHOUT A MASTER
6497	ITALIAN WITH GRAMMAR

line_num	line_str
6821	A BURLESQUE BIOGRAPHY
7166	WIT INSPIRATIONS OF THE "TWO-YEAR-OLDS"
7302	AN ENTERTAINING ARTICLE
7620	A LETTER TO THE SECRETARY OF THE TREASURY
7648	AMENDED OBITUARIES
7725	A MONUMENT TO ADAM
7799	A HUMANE WORD FROM SATAN
7985	ADVICE TO LITTLE GIRLS
8270	THE DANGER OF LYING IN BED
8391	PORTRAIT OF KING WILLIAM III
8954	EXTRACTS FROM ADAM'S DIARY
9329	EVE'S DIARY
9701	EXTRACT FROM ADAM'S DIARY

In [153…

```
t_books = []
for pat in ohco_pat_list[6:7]:

    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
    try:
        text.import_source().parse_tokens();
    except:
        text.import_source(char_encoding = 'latin-1').parse_tokens();
    text.TOKENS['book_id'] = book_id
    text.TOKENS = text.TOKENS.reset_index().set_index(['book_id'] + text.OHCO)

    t_books.append(text.TOKENS)
```

```
Tokenizing 102 the tragedy of puddnhead wilson
line_str chap_str
Index(['chap_str'], dtype='object')
```

In [154…

```
t_CORPUS = pd.concat(t_books).sort_index()
```

In [155…

```
t_CORPUS = t_CORPUS[t_CORPUS.term_str != '']
```

In [156…
```
t_CORPUS
```

Out[156…

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
102	1	0	0	0	(to, TO)	TO	to	to
				1	(the, DT)	DT	the	the
				2	(Reader., NNP)	NNP	Reader.	reader
		1	0	0	(There, EX)	EX	There	there
				1	(is, VBZ)	VBZ	is	is
	...	...	...	...	...	...	...	...
	22	145	0	18	(would, MD)	MD	would	would
				19	(be, VB)	VB	be	be
				20	(a, DT)	DT	a	a
				21	(long, JJ)	JJ	long	long
				22	(story.", NN)	NN	story."	story

53953 rows × 4 columns

In [157…
```
t_CORPUS.loc[(102, 4)]
```

Out[157…

para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
0	0	0	(Roxy, NNP)	NNP	Roxy	roxy
		1	(Plays, VBZ)	VBZ	Plays	plays
		2	(a, DT)	DT	a	a
		3	(Shrewd, NNP)	NNP	Shrewd	shrewd
		4	(Trick., NNP)	NNP	Trick.	trick
...	...	...	...	...	...	...
30	1	15	(permanently, RB)	RB	permanently	permanently
		16	(out, IN)	IN	out	out
		17	(of, IN)	IN	of	of
		18	(her, PRP$)	PRP$	her	her
		19	(mind., NN)	NN	mind.	mind

2060 rows × 4 columns

In [158…
```
CORPUS.loc[(142, 14)]
```

Out [158…

para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
0	0	0	(All, DT)	DT	All	all
		1	(infants, NNS)	NNS	infants	infants
		2	(appear, VBP)	VBP	appear	appear
		3	(to, TO)	TO	to	to
		4	(have, VB)	VB	have	have
...	...	...	...	...	...	...
29	3	9	(of, IN)	IN	of	of
		10	(infanticide, NN)	NN	infanticide	infanticide
		11	(in, IN)	IN	in	in
		12	(our, PRP$)	PRP$	our	our
		13	(family., NN)	NN	family.	family

1145 rows × 4 columns

In [159…

```
LIB
```

Out [159…

book_id	source_file_path	title	ch
70	Twain/70-what_is_man.txt	what is man	WHAT IS MAN?|THE DEATH OF TURN
74	Twain/74-the_adventures_of_tom_sawyer.txt	the adventures of tom sawyer	^\s*CHAPTER\s*[I
76	Twain/76-the_adventures_of_huckleberry_finn.txt	the adventures of huckleberry finn	^\s*CHAPTER\s*(?:[IVXLCM]+\.|TI
86	Twain/86-a_connecticut_yankee_in_king_arthurs_...	a connecticut yankee in king arthurs court	^\s*(?:PREFACE|A WORD OF EXPLANA
91	Twain/91-tom_sawyer_abroad.txt	tom sawyer abroad	CHAPTER\s[I
93	Twain/93-tom_sawyer_detective.txt	tom sawyer detective	^CHAPTER\s[IVXLCM
102	Twain/102-the_tragedy_of_puddnhead_wilson.txt	the tragedy of puddnhead wilson	Whisper|CHAPTER\s[IVXLCM]+\.|CONC
119	Twain/119-a_tramp_abroad.txt	a tramp abroad	^(?:CHAPTER\s[IVXLCM]+|APPENDIX

book_id	source_file_path	title	ch
142	Twain/142-the_30000_bequest_and_other_stories.txt	the 30000 bequest and other stories	THE $30,000 BEQUEST\|$ADOG'ST$
245	Twain/245-life_on_the_mississippi.txt	life on the mississippi	^(THE 'BODY OF THE NATION'|CHA
1044	Twain/1044-extract_from_captain_stormfields_vi...	extract from captain stormfields visit to Heaven	CHAPTER\s[I'
1086	Twain/1086-a_horses_tale.txt	a horses tale	^[I'
1837	Twain/1837-the_prince_and_the_pauper.txt	the prince and the pauper	^\s*CHAPTER\s*
2874	Twain/2874-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 1	^Chapt
2875	Twain/2875-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 2	^[0-9
2895	Twain/2895-following_the_equator.txt	following the equator	^(CHAPTER[,]?\s[IVXLCM]+|CONCL
3171	Twain/3171-in_defense_of_harriet_shelley.txt	in defense of harriet shelley	^[I'
3172	Twain/3172-fenimore_coopers_literary_offences.txt	fenimore coopers literary offences	The Pathfinder and The
3173	Twain/3173-essays_on_paul_bourget.txt	essays on paul bourget	(WHAT PAUL BOURGET|A LITTLE
3176	Twain/3176-the_innocents_abroad.txt	the innocents abroad	^\s*(CHAPTER\s*[IVXLCM]+\.$|CON
3177	Twain/3177-roughing_it.txt	roughing it	^(CHAPTER\s[IVXLCM]+|APP
3178	Twain/3178-the_gilded_age.txt	the gilded age	^(CHAPTER\s[IVXLCM]+|APP
3179	Twain/3179-the_american_claimant.txt	the american claimant	^(CHAPTER\s[IVXLCM]+|APP
3180	Twain/3180-a_double_barrelled_detective_story.txt	a double barrelled detective story	^[IVXL(
3181	Twain/3181-the_stolen_white_elephant.txt	the stolen white elephant	^[IVXL(

book_id	source_file_path	title	ch
3182	Twain/3182-some_rambling_notes_of_an_idle_excu...	some rambling notes of an idle excursion	^[IV
3183	Twain/3183-the_facts_concerning_the_recent_car...	the facts concerning the recent carnival of cr...	I was fe
3184	Twain/3184-alonzo_fitz_and_other_stories.txt	alonzo fitz and other stories	THE LOVES OF ALONZO FITZ CLAR RO
3185	Twain/3185-those_extraordinary_twins.txt	those extraordinary twins	^CHAPTER\s[IVXLCM]-
3186	Twain/3186-the_mysterious_stranger_and_other_s...	the mysterious stranger and other stories	^(Chapter\s[0-9]+|A FABLE|HU D
3188	Twain/3188-mark_twain_speeches.txt	mark twain speeches	INTRODUCTION|*PREFACE*|THE S S
3189	Twain/3189-sketches_new_and_old.txt	sketches new and old	MY WATCH|POLITICAL ECONOMY|THE I
3190	Twain/3190-1601_conversation_as_it_was_by_the_...	1601 conversation as it was by the social fire...	^(INTRODUCTION| PRINTING|FOOTN
3191	Twain/3191-goldsmiths_friend_abroad_again.txt	goldsmiths friend abroad again	LETTER\s|
3192	Twain/3192-the_curious_republic_of_gondour_and...	the curious republic of gondour and other whim...	THE CURIOUS REPUBLIC OF GO MEMOR
3199	Twain/3199-the_letters_of_mark_twain.txt	the letters of mark twain	^[IVXLCM]+
3250	Twain/3250-how_to_tell_a_story_and_other_essay...	how to tell a story and other essays	HOW TO TEL| |*THEWOUNDEDSOLDI*
3251	Twain/3251-the_man_that_corrupted_hadleyburg_a...	the man that corrupted hadleyburg and other st...	^THE MAN THAT CO HADLEYBURG$|^M
19484	Twain/19484-editorial_wild_oats.txt	editorial wild oats	^My First Literary Venture$|^Journalis
19987	Twain/19987-chapters_from_my_autobiography.txt	chapters from my autobiography	^(INTRODUCTION|[IVXLCM]+|CHAPT M

book_id	source_file_path	title	ch
33077	Twain/33077-the_treaty_with_china_its_provisio...	the treaty with china its provisions explained	^New Yo
60900	Twain/60900-merry_tales.txt	merry tales	^THE PRIVATE HISTORY OF A CAMPA
61522	Twain/61522-the_1000000_bank_note.txt	the 1000000 bank note	^_THE £1,000,000 BANK-NOTE_$|' TE
62636	Twain/62636-to_the_person_sitting_in_darkness.txt	to the person sitting in darkness	^Extending the
62739	Twain/62739-king_leopolds_soliloquy.txt	king leopolds soliloquy	^(\[_Throws down pamphlet has

In [ ]: