Analysis for Twain Works through Module 7 (Language Models, NLP, Vector Space Models, Similarity and Clustering, PCA)

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 sklearn.metrics import silhouette_score

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

In [2]:
```
sns.set()
```

In [3]:
```
OHCO = ["book_id", "chap_id", "para_num", "sent_num", "token_num"]
```

In [4]:
```
SENTS = OHCO[:4]
PARAS = OHCO[:3]
```

```
CHAPS = OHCO[:2]
BOOKS = OHCO[:1]
```

In [5]:
```
LIB = pd.read_csv("twain_pre_LIB.csv").set_index(BOOKS).sort_index()
```

In [6]:
```
CORPUS = pd.read_csv("twain_pre_CORPUS.csv").set_index(OHCO)

# remove NaN values
CORPUS = CORPUS[~CORPUS.term_str.isna()]
```

In [7]:
```
VOCAB = pd.read_csv("twain_pre_VOCAB.csv")

VOCAB['term_str'] = VOCAB['term_str'].astype('str')

VOCAB = VOCAB.set_index('term_str')
```

In [8]:
```
VOCAB
```

Out[8]:

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 [9]:
```
LIB
```

Out[9]:

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'STA$
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

book_id	source_file_path	title	ch
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]-
3186	Twain/3186-the_mysterious_stranger_and_other_s...	the mysterious stranger and other stories	^(Chapter\s[0-9]+|A FABLE|HUN 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

book_id	source_file_path	title	ch
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 G 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 TELL A STORY | *THEWOUNDE*
3251	Twain/3251-the_man_that_corrupted_hadleyburg_a...	the man that corrupted hadleyburg and other st...	^THE MAN THAT C 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 Y
60900	Twain/60900-merry_tales.txt	merry tales	^THE PRIVATE HISTORY OF A CAMP
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 [10]:
```
CORPUS
```

Out[10]:

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

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
				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

2970356 rows × 4 columns

M03: Language Models

Create Training Vocab (V_{train})

In [11]:
```
V_TRAIN = sorted(list(set(VOCAB.index)))
```

In [12]:
```
len(V_TRAIN)
```

Out[12]:
```
53853
```

Generate Training Sentences

In [13]:
```
# convert col type to str otherwise errors when generating training sentences
CORPUS['term_str'] = CORPUS.term_str.astype('str')
CORPUS['token_str'] = CORPUS.term_str.astype('str')
```

In [14]:
```
S_TRAIN = list(CORPUS.groupby(OHCO[:-1]).term_str.apply(lambda x: ' '.join(x)).v
```

In [15]:
```
len(S_TRAIN)
```

Out[15]: 155621

In [16]:
```
S_TRAIN[:5]
```

Out[16]:
```
['by mark twain',
 'samuel langhorne clemens 1835 1910',
 'contents',
 'what is man',
 'the death of jean']
```

Generate and Count Ngrams

In [17]:
```
train = NgramCounter(S_TRAIN, V_TRAIN)
train.generate()
```

n-gram token table

In [18]:
```
train.I
```

Out[18]:

		w0	w1	w2
sent_num	**token_num**			
0	**0**	<s>	<s>	by
	1	<s>	by	mark
	2	by	mark	twain
	3	mark	twain	</s>
	4	twain	</s>	<s>
...	**...**	...	...	...
155620	**11**	soliloquy	by	mark
	12	by	mark	twain
	13	mark	twain	</s>
	14	twain	</s>	NaN
	15	</s>	NaN	NaN

3437219 rows × 3 columns

Unigram table

In [129…
```
stop_words = VOCAB.loc[VOCAB.stop == 1].index.values
```

In [130…
```
unigram_df = train.LM[0].sort_values('n', ascending = False)
unigram_df
```

Out[130…

	n	mle	p	log_p

w0	n	mle	p	log_p
<s>	311242	9.055053e-02	9.055053e-02	-3.465133
the	161902	4.710261e-02	4.710261e-02	-4.408049
</s>	155621	4.527526e-02	4.527526e-02	-4.465133
and	128130	3.727723e-02	3.727723e-02	-4.745562
of	82827	2.409710e-02	2.409710e-02	-5.374997
...	...	...	...	...
jumieges	1	2.909329e-07	2.909329e-07	-21.712810
jument	1	2.909329e-07	2.909329e-07	-21.712810
jumblings	1	2.909329e-07	2.909329e-07	-21.712810
jumbling	1	2.909329e-07	2.909329e-07	-21.712810
übergeschlagen	1	2.909329e-07	2.909329e-07	-21.712810

53854 rows × 4 columns

In [131…
```
unigram_df.filter(regex = '^[^<]', axis = 0)
```

Out[131…

w0	n	mle	p	log_p
the	161902	4.710261e-02	4.710261e-02	-4.408049
and	128130	3.727723e-02	3.727723e-02	-4.745562
of	82827	2.409710e-02	2.409710e-02	-5.374997
a	76650	2.230000e-02	2.230000e-02	-5.486812
to	75666	2.201373e-02	2.201373e-02	-5.505453
...	...	...	...	...
jumieges	1	2.909329e-07	2.909329e-07	-21.712810
jument	1	2.909329e-07	2.909329e-07	-21.712810
jumblings	1	2.909329e-07	2.909329e-07	-21.712810
jumbling	1	2.909329e-07	2.909329e-07	-21.712810
übergeschlagen	1	2.909329e-07	2.909329e-07	-21.712810

53852 rows × 4 columns

Bigram table

In [132…
```
bigram_df = train.LM[1].sort_values('n', ascending = False)
bigram_df
```

Out[132…

w0	w1	n	mle	mle2	p	log_p
<s>	<s>	155621	4.527528e-02	0.500000	0.426250	-1.230230
</s>	<s>	155620	4.527499e-02	0.999994	0.742910	-0.428741
of	the	18933	5.508234e-03	0.228585	0.138527	-2.851762
<s>	i	14638	4.258677e-03	0.047031	0.040096	-4.640387
in	the	14055	4.089063e-03	0.277954	0.134610	-2.893140
…	…	…	…	…	…	…
hillfor	just	1	2.909330e-07	1.000000	0.000037	-14.716793
hillock	at	1	2.909330e-07	0.333333	0.000037	-14.716846
	five	1	2.909330e-07	0.333333	0.000037	-14.716846
	he	1	2.909330e-07	0.333333	0.000037	-14.716846
übergeschlagen	</s>	1	2.909330e-07	1.000000	0.000037	-14.716793

764796 rows × 5 columns

In [133…

```
reduced_bigram = bigram_df.reset_index()

# remove spaces
reduced_bigram = reduced_bigram.loc[~((reduced_bigram.w0.str.contains('<')) | (r

# remove stop words
reduced_bigram = reduced_bigram.loc[~((reduced_bigram.w0.isin(stop_words)) | (re

reduced_bigram
```

Out[133…

w0	w1	n	mle	mle2	p	log_p
new	york	738	2.147085e-04	0.244128	0.012993	-6.266127
mark	twain	608	1.768872e-04	0.480253	0.011049	-6.499990
years	ago	597	1.736870e-04	0.197095	0.010513	-6.571708
dont	know	506	1.472121e-04	0.158571	0.008888	-6.813971
good	deal	404	1.175369e-04	0.084149	0.006905	-7.178188
…	…	…	…	…	…	…
hill	west	1	2.909330e-07	0.002618	0.000037	-14.726963
	wheeling	1	2.909330e-07	0.002618	0.000037	-14.726963
	wherewith	1	2.909330e-07	0.002618	0.000037	-14.726963
	work	1	2.909330e-07	0.002618	0.000037	-14.726963
hillock	five	1	2.909330e-07	0.333333	0.000037	-14.716846

325618 rows × 5 columns

Trigram table

In [134…]
```
trigram_df = train.LM[2].sort_values('n', ascending = False)
trigram_df
```

Out[134…]

			n	mle	mle2	p	log_p
w0	w1	w2					
</s>	<s>	<s>	155620	4.527500e-02	1.000000	5.365485e-05	-14.185932
<s>	<s>	i	14638	4.258678e-03	0.094062	5.047219e-06	-17.596080
		the	12480	3.630844e-03	0.080195	4.303186e-06	-17.826163
		it	8015	2.331828e-03	0.051503	2.763748e-06	-18.464942
		he	7764	2.258804e-03	0.049890	2.677209e-06	-18.510839
…	…	…	…	…	…	…	…
harte	and	mr	1	2.909330e-07	0.250000	6.895949e-10	-30.433532
		i	1	2.909330e-07	0.250000	6.895949e-10	-30.433532
		getting	1	2.909330e-07	0.250000	6.895949e-10	-30.433532
		clemens	1	2.909330e-07	0.250000	6.895949e-10	-30.433532
übergeschlagen	</s>	<s>	1	2.909330e-07	1.000000	6.895949e-10	-30.433532

1914877 rows × 5 columns

In [135…]
```
reduced_trigram = trigram_df.reset_index()

# remove spaces
reduced_trigram = reduced_trigram.loc[~((reduced_trigram.w0.str.contains('<')) |

# remove stop words
reduced_trigram = reduced_trigram.loc[~((reduced_trigram.w0.isin(stop_words)) |

reduced_trigram
```

Out[135…]

			n	mle	mle2	p	log_p
w0	w1	w2					
hundred	years	ago	76	2.211091e-05	0.288973	2.654940e-08	-25.166745
wilsons	new	calendar	74	2.152905e-05	1.000000	2.585981e-08	-25.204713
puddnhead	wilsons	new	74	2.152905e-05	0.632479	2.585981e-08	-25.204713
twenty	four	hours	70	2.036531e-05	0.569106	2.448062e-08	-25.283785
yrs	ever	mark	55	1.600132e-05	0.833333	1.930866e-08	-25.626177
…	…	…	…	…	…	…	…

			n	mle	mle2	p	log_p
w0	w1	w2					
harsher	methods	would	1	2.909330e-07	1.000000	6.895949e-10	-30.433532
harsh	words	troubled	1	2.909330e-07	1.000000	6.895949e-10	-30.433532
hart	came	along	1	2.909330e-07	1.000000	6.895949e-10	-30.433532
harte	remarked	upon	1	2.909330e-07	1.000000	6.895949e-10	-30.433532
	became	editor	1	2.909330e-07	1.000000	6.895949e-10	-30.433532

157792 rows × 5 columns

In [136…
```
reduced_trigram.head(10)
```

Out[136…

			n	mle	mle2	p	log_p
w0	w1	w2					
hundred	years	ago	76	0.000022	0.288973	2.654940e-08	-25.166745
wilsons	new	calendar	74	0.000022	1.000000	2.585981e-08	-25.204713
puddnhead	wilsons	new	74	0.000022	0.632479	2.585981e-08	-25.204713
twenty	four	hours	70	0.000020	0.569106	2.448062e-08	-25.283785
yrs	ever	mark	55	0.000016	0.833333	1.930866e-08	-25.626177
mrs	jane	clemens	41	0.000012	1.000000	1.448149e-08	-26.041214
fifty	years	ago	38	0.000011	0.351852	1.344710e-08	-26.148130
puddnhead	wilsons	calendar	38	0.000011	0.324786	1.344710e-08	-26.148130
five	years	ago	36	0.000010	0.251748	1.275751e-08	-26.224078
	hundred	years	35	0.000010	0.204678	1.241271e-08	-26.263607

In [137…
```
LIB.type.value_counts()
```

Out[137…
```
non-fiction    17
stories        17
novel          11
Name: type, dtype: int64
```

Create language model

In [22]:
```
# generate n-gram model using n-gram counter
model = NgramLanguageModel(train)

# implement smoothing (with k = 1) to prevent zero prob. for n-grams unseen in t
model.k = 1
model.apply_smoothing()
```

In [23]:
```
# list of ngram token tables for unigram, bigram, trigram
```

```
ngram = model.NG

# list of unigram, bigram, trigram dfs with count, max likelihood estimate (MLE)
LM = model.LM

Z1 = model.Z1

Z2 = model.Z2
```

Generate a text with the `.generate_text()` method of the `langmod.NgramLanguageModel` object (`model`)

In [24]:
```
model.generate_text()
```

```
01. COMFORTABLE GOOD SERVICE OF ANY KIND OF FANCY PERSIAN STUFF FILLED WITH HUMA
N ASPECTS REMOVED OH VERY WELL FORTIFIED WITH RESPONSIBLE APPETITES AND GODLESS
ATTITUDES OF THE LITTLE CREATURES AS ALL THOSE PEOPLE A LIMPID TORRENT GOES WHIS
TLING DOWN THE LIGHTNING HAS STRUCK A FINAL STRUGGLE FOR INDEPENDENCE AND THAT I
S YOU CANNOT SEE THE BOIL SCAR UNDER ARMPIT.

02. AND THERE STOOD FATHER PETER HAD BEEN A DEAL FOR HE HAS OPENED HIS BOX AND P
UT HIS HAND ON A HALF AND WOULD LIKE TO SAMPLE IT I TELL YOU IT WAS A POOR LITTL
E CHAPAND WONDERING WHAT WAS IN OXFORD STREET JUST THENIT HAD BEEN ASLEEP.

03. DO YOU REMEMBER THAT HE IS IRRELEVANT AND ALSO FOR A CRIPPLED OCTOGENARIAN W
HO COULD HAVE QUALIFIED AS A MANS HEAD ON THE CURBSTONES WERE STILL THE END OF T
HAT OLD MAN BECAME EXHAUSTED AND SPIRITUALLY RECONCILED.

04. BUT I KNOW THAT NOR THE LIE HE WOULD HE ASK THE INDIANS.

05. IN TIME FOR A LITTLE STRONGER INCLINATION LIKE THAT ONCE STOOD A MOMENT BEFO
RE AND AFTER THE DATE AND WE LOOKED DOWN UPON THE JEW HAS REAL NEED FOR HE PRESE
NTLY DIED A QUICK ESCAPE WHILE THE COST OF ADVERTISING THE LECTURE FIELD.

06. THEY HAVE NOT SUFFERED FROM CORPULENCE AND HAD NOT BEFORE HE ASKED.

07. WELL WE WERE COMPROMISED BY IT ALWAYS MAKES ME FEEL MUCH TROUBLED AND DESPON
DENT MOREOVER HE WAS JUST WHAT I REGARDED THAT AS A MATTER OF HUGE WILD BOARS HE
ADS PRESERVED AND PERPETUATED AND WITH THEM AND TOOK HER TO REARRANGE THEM TO SH
OW WHO WAS PRESENT HALF OF IT.

08. THIS COULD NOT CONSENT TO A DISTANCE BECAUSE ALTHOUGH HE DID NOT KNOW HOW TO
SNATCH THE OLD TESTAMENT FOR A TRUNK AND GAVE MAMMA HER TICKET AND WITH A PATHET
IC CONFUSION THEN GOT UP AND PUT IT WHERE IT CAN DO THAT BUT YOU WILL FIND IT VA
LUABLE.

09. SHE SAYS.

10. OH IT WAS THOSE THUGS HAVE BUILT MONUMENTS ON THEIR PRECIOUS ACCUMULATIONSAN
D AFTER ALL.

11. AFTER HIS RECOVERY FROM AN ENGLISH TOURIST RAILROAD UP THE INDIGNANT BRITISH
LION ROSE WITH STEM THORN LEAVES PETALS COMPLETE AND THE QUIET REPLY GET YOUR OW
N DEPUTIES AND NOT ONLY GLAD.

12. HE SAID SHE WOULD DECEIVE THE UNWARY.
```

```
13. AS IT IS DANGEROUS TO REFER TO THE RESPONSIBILITIES ON MYSELF.

14. NO.

15. HE SAID THE PUNISHMENT UPON THE PASSER BY WITH THE COLD AND HE ENJOYS RIDING
ON A COMET.

16. THAT OFFICER MUST SUFFER IF WE LIKED WHETHER IT IS DELIVERED AND DAMNABLE PO
RTRAITS OF THE PILOT HOUSE AND I SHALL BE HELD BY THE LOAD UPON MY CONSCIENCE BE
GAN TO PLAN THINGS AND NOT NEEDLESSLY ELABORATED YOUR REPORT OF THE EVASION WORK
ED ALL RIGHT ALL RIGHTHAVE IT YOUR SUBTLE PERSUASIONS THAT HAVE BEEN INNUMERABLE
TEMPORARY SEEKERS OF 49 OR MAYBE HALF A MILE ALWAYS FOLLOWING PATHS WHICH HAD BE
FALLEN TOM HAD NEVER SUFFERED.

17. WHO SAID PROCRASTINATION IS THE ENTIRE NATIONAL CHARACTER THAT IS THAT WHAT
HIS HISTORY CREATING REVOLVER.

18. THE FOLLOWING LETTER.

19. I WAS AWAY FROM THE PARIS AND EVERYWHERE.

20. BUT WHEN IT IS TEN DOLLARS AND I JUMPED IN AND YEAR OUT OF WORK IN HER COUNT
ENANCE SHOWED UNCOMMON VIVACITY WITH A CHORUS FROM ALL THE HOUSES HAVE VANISHED.
```

Examining redundancy for unigrams, bigrams, trigrams → redundancy increases

- Entropy equation for n-grams (ng): $H = \sum p(ng) log_2(\frac{1}{p(ng)})$, where p is the `.mle` in the unigram, bigram, and trigram tables in the language model
- Redundancy: $R = 1 - \frac{H}{H_{max}}$, where H is the actual model entropy and $H_{max} = \frac{1}{n} * log_2(\frac{1}{n}) * n = log_2(\frac{1}{n})$ is the max entropy

In [25]:
```
V = len(VOCAB)
```

In [26]:
```
R = []
for i in range(3):
    N = V**(i+1)
    H = (train.LM[i]['mle'] * np.log2(1/train.LM[i]['mle'])).sum()
    Hmax = np.log2(N)
    R.append(int(round(1 - H/Hmax, 2) * 100))
```

In [27]:
```
R
```

Out[27]:
```
[41, 50, 60]
```

Using the bigram model represented as a matrix (too large to use `BGX = model.LM[1].n.unstack()` so use method below), explore the relationship between bigram pairs using the following lists for the first and second words of the bigrams of interest

In [28]:
```
w0 = ['he', 'she']

w1 = ['said', 'heard']

bigram_pairs = [i for i in itertools.combinations(w0 + w1, 2) if i[0] in w0 and
```

In [29]:
```
LM[1].loc[bigram_pairs].n.unstack()
```

Out[29]:

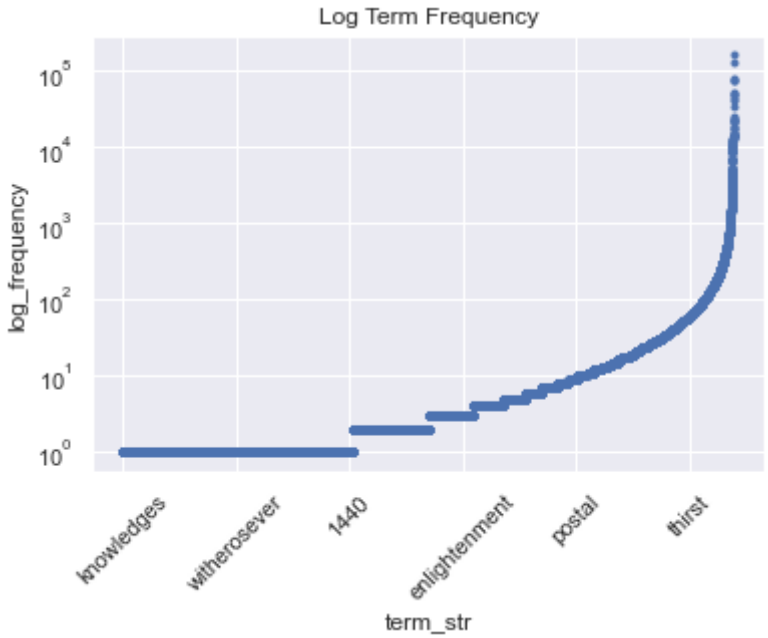
w0	w1 said	heard
he	1644	66
she	533	23

M05: Vector Space Models

Zipf's Law:

- **Equation**: $f \propto \frac{1}{r} \rightarrow f = \frac{k}{r}$, where $k = fr$
  - f: token frequency
  - r: token rank $\rightarrow$ okens ordered in terms of frequency s.t. rank $r = 1, 2, 3, \ldots, N$, where N is the number of tokens and the token with rank $r = 1$ is the most frequent token

In [30]:
```
VOCAB.n.sort_values().plot(ylabel = "log_frequency", logy=True, style = '.', rot
```

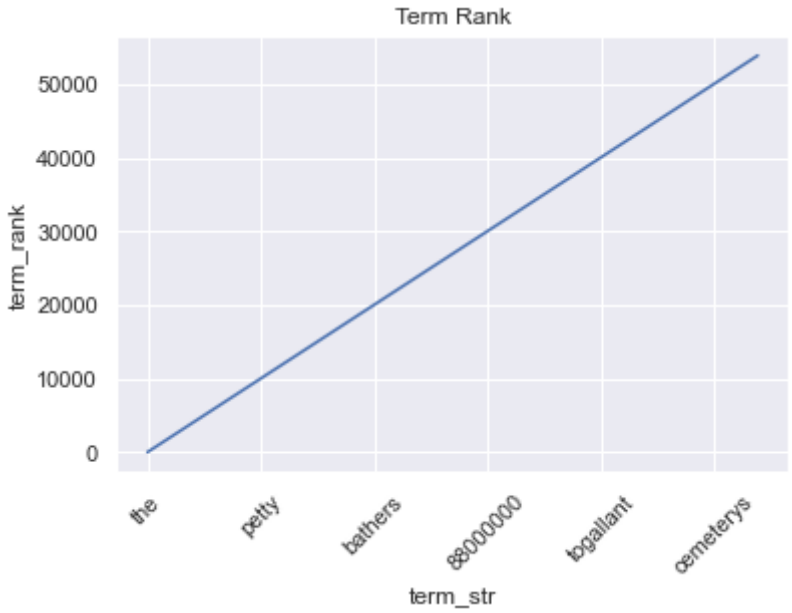


Add Term Rank r to VOCAB

In [31]:
```
if 'term_rank' not in VOCAB.columns:
```

```
VOCAB = VOCAB.sort_values('n', ascending = False).reset_index()
VOCAB.index.name = 'term_rank'
VOCAB = VOCAB.reset_index()
VOCAB['term_rank'] = VOCAB['term_rank'] + 1
VOCAB = VOCAB.set_index('term_str')
```

In [32]:
```
VOCAB.term_rank.plot(ylabel = "term_rank", logx = False, rot = 45, title = "Term
```



In [33]:
```
VOCAB.head()
```

Out[33]:

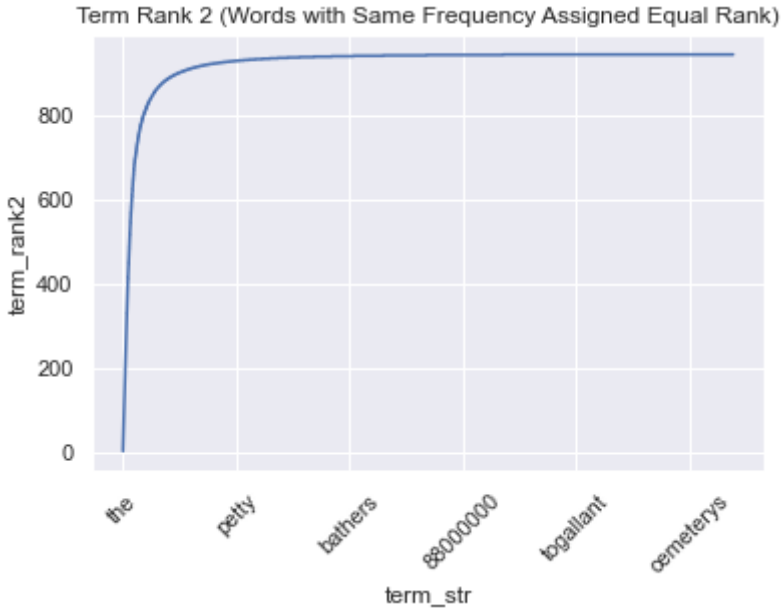
term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
the	1	161902	3	0.054506	4.197452	DT	20	{'RP', 'NNS', 'FW', 'VBG', 'RB', 'JJ', 'PRP', ...	1	
and	2	128130	3	0.043136	4.534964	CC	17	{'RP', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'POS'...	1	
of	3	82827	2	0.027884	5.164400	IN	17	{'PDT', 'RP', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP'...	1	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
a	4	76650	1	0.025805	5.276215	DT	19	{'RP', 'NNS', 'FW', 'RB', 'JJ', 'NNPS', 'PRP',...	1	
to	5	75666	2	0.025474	5.294856	TO	18	{'PDT', 'RP', 'VBZ', 'JJR', 'NNS', 'TO', 'VB',...	1	

Alternate Rank: words that appear the same number of times given the same rank

In [34]:
```
# times each num of times a term appears (e.g., 18273 terms appear 1 time)
# sort in descending order
# reset indices and rename cols --> nn: times each num of times a term appears

new_rank = VOCAB.n.value_counts()\
                  .sort_index(ascending = False).reset_index().reset_index()\
                  .rename(columns={'level_0': 'term_rank2', 'index': 'n', 'n':
                   .set_index('n')

VOCAB['term_rank2'] = VOCAB.n.map(new_rank.term_rank2) + 1
```

In [35]:
```
VOCAB.term_rank2.plot(ylabel = 'term_rank2', logx = False, rot = 45, title = "Te
```

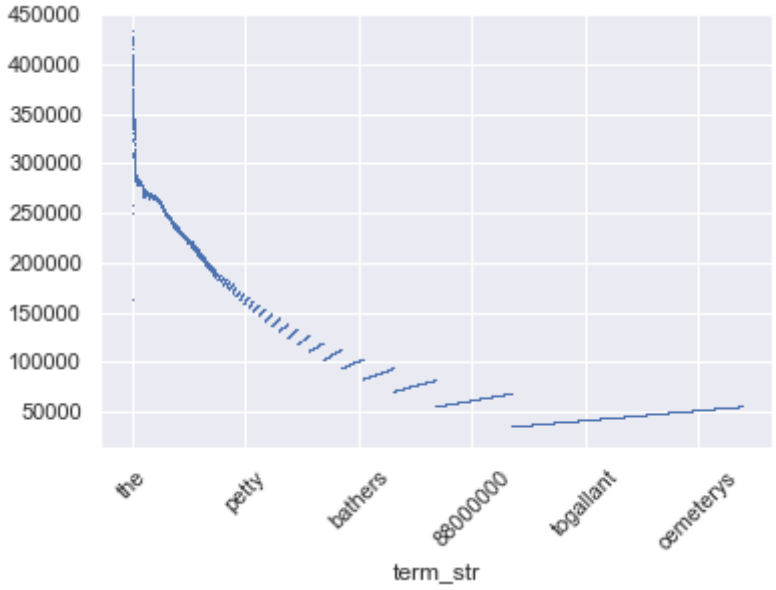


Compute Zipf's k using `term_rank` and `term_rank2`

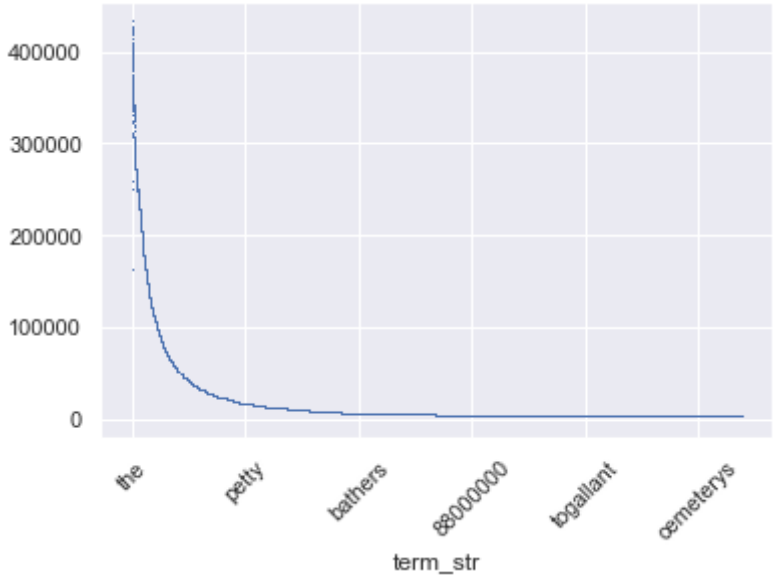
- **Equation**: $k = fr$

In [36]:
```
VOCAB['zipf_k'] = VOCAB.n * VOCAB.term_rank
VOCAB['zipf_k2'] = VOCAB.n * VOCAB.term_rank2
```

In [37]:
```
VOCAB.zipf_k.plot(style = ',', rot = 45);
```



In [38]:
```
VOCAB.zipf_k2.plot(style = ',', rot = 45);
```

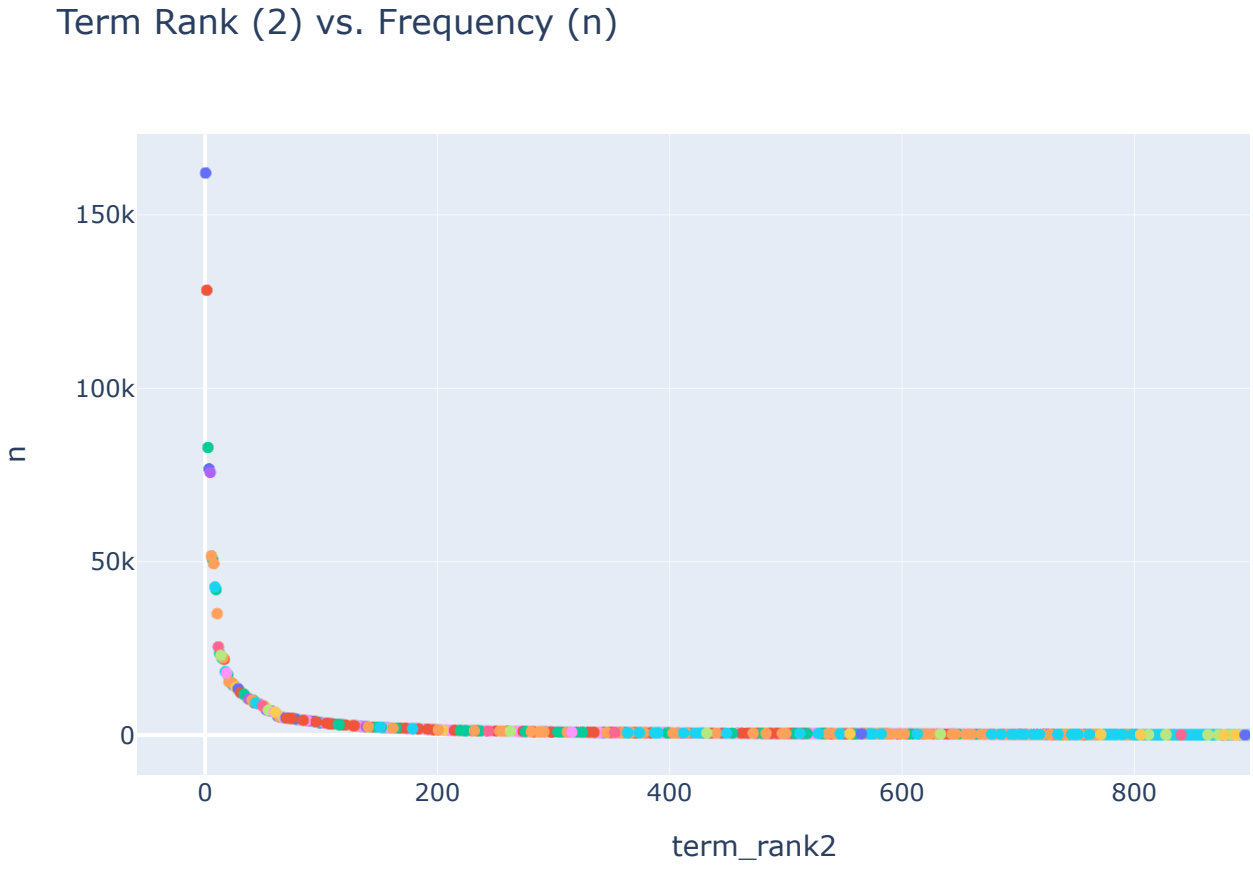


Rank vs. N (frequency n)

As rank (`term_rank2`) increases, frequnecy (`n`) decreases

In [39]:
```
# scatter plot of term_rank2 vs. n color coded by part of speech (POS)

px.scatter(VOCAB.reset_index(),
```

```
        x = 'term_rank2', y = 'n',
        title = 'Term Rank (2) vs. Frequency (n)',
        log_y = False, log_x = False,
        hover_name = 'term_str',
        color = 'max_pos',
        height = 500, width = 800)
```



BOW (Bag of Words) and TFIDF (Term Frequency - Inverse Document Frequency)

In [40]:
```
BOW = create_bow(CORPUS, CHAPS)
```

In [41]:
```
DTCM, TFIDF, BOW, DFIDF, VOCAB = get_tfidf(BOW, VOCAB, tf_method = 'max', idf_me
```

In [42]:
```
VOCAB
```

Out[42]:

	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos
term_str								

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos
the	1	161902	3	5.450559e-02	4.197452	DT	20	{'RP', 'NNS', 'FW', 'VBG', 'RB', 'JJ', 'PRP', ...
and	2	128130	3	4.313598e-02	4.534964	CC	17	{'RP', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'POS'...
of	3	82827	2	2.788437e-02	5.164400	IN	17	{'PDT', 'RP', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP'...
a	4	76650	1	2.580483e-02	5.276215	DT	19	{'RP', 'NNS', 'FW', 'RB', 'JJ', 'NNPS', 'PRP',...
to	5	75666	2	2.547356e-02	5.294856	TO	18	{'PDT', 'RP', 'VBZ', 'JJR', 'NNS', 'TO', 'VB',...
...	...	...	...	...	...	...	...	...
ouvre	53850	1	5	3.366579e-07	21.502213	NN	1	{'NN'}
outworks	53851	1	8	3.366579e-07	21.502213	NNS	1	{'NNS'}
outwitted	53852	1	9	3.366579e-07	21.502213	JJ	1	{'JJ'}
outvoted	53853	1	8	3.366579e-07	21.502213	VBD	1	{'VBD'}
übergeschlagen	53854	1	14	3.366579e-07	21.502213	NN	1	{'NN'}

53854 rows × 20 columns

Document-Term Count Matrix `DTCM`

In [43]:
```
DTCM
```

Out[43]:

	term_str	0	00	01	02	03	04	05	06	07	08	...	étant	éternumens	étouffan
book_id	chap_id														
70	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
62739	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0

1108 rows × 53852 columns

Reduce number of features in VOCAB , TFIDF matrix to the 1000 most significant terms

- #### DFIDF: the significance measure
- #### Only include terms whose maximum part-of-speech belongs to this set:
  - #### NN NNS VB VBD VBG VBN VBP VBZ JJ JJR JJS RB RBR RBS.
  - #### Note, these are all open categories, excluding proper nouns.

In [44]:
```
# open POS categories
open_cats = ['NN', 'NNS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'JJ',
             'JJR', 'JJS', 'RB', 'RBR', 'RBS']

# reduce VOCAB to significant terms --> filter POS, sort , take top 1000
SIGS = VOCAB.loc[VOCAB.max_pos.isin(open_cats)] \
            .sort_values('dfidf', ascending = False) \
            .iloc[:1000,]
```

In [45]:
```
SIGS
```

Out[45]:

	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	st
term_str										

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	st
saying	401	705	6	0.000237	12.040734	VBG	12	{'VBG', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'IN'...	0	
seem	378	752	4	0.000253	11.947624	VB	7	{'VBZ', 'VB', 'NNP', 'JJ', 'VBP', 'CC', 'NN'}	0	
indeed	373	758	6	0.000255	11.936159	NN	12	{'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'POS', 'IN'...	0	
couldnt	246	1219	7	0.000410	11.250731	VBP	11	{'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'RB', 'MD',...	0	
door	319	885	4	0.000298	11.712680	NN	8	{'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'FW', 'JJ',...	0	
...	...	...	...	...	...	...	...	...	...	
record	1249	215	6	0.000072	13.754020	NN	6	{'VBZ', 'VB', 'NNP', 'VBP', 'JJ', 'NN'}	0	
watched	1423	188	7	0.000063	13.947624	VBD	8	{'VB', 'VBD', 'IN', 'VBN', 'JJR', 'VBP', 'JJ',...	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	st
laws	1231	219	4	0.000074	13.727426	NNS	8	{'VBZ', 'NNS', 'JJS', 'VB', 'NNP', 'IN', 'JJ',...	0	
information	1484	180	11	0.000061	14.010360	NN	4	{'JJ', 'VB', 'NN', 'NNS'}	0	
questions	1276	212	9	0.000071	13.774293	NNS	7	{'NNS', 'VB', 'NNP', 'RB', 'VBP', 'JJ', 'NN'}	0	

1000 rows × 20 columns

In [139…
```
SIGS.head(10).index.values
```

Out[139…
```
array(['saying', 'seem', 'indeed', 'couldnt', 'door', 'taken', 'deal',
       'fifty', 'getting', 'perhaps'], dtype=object)
```

"Collapse" the TFIDF matrix so that it contains mean TFIDF of each term by book.

- #### Result: matrix with book IDs as rows, significant terms as cols

In [46]:
```
TFIDF
```

Out[46]:

book_id	chap_id	term_str	0	00	01	02	03	04	05	06	07	08	...	étant	éternumens	étouffan
70	1		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	2		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	3		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	4		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	5		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	
62739	2		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	3		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	4		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0

book_id	chap_id	0	00	01	02	03	04	05	06	07	08	...	étant	éternumens	étouffan
	5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0
	6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0

1108 rows × 53852 columns

In [47]:
```
TFIDF_sigs = TFIDF[SIGS.index]
```

In [48]:
```
TFIDF_sigs
```

Out[48]:

book_id	chap_id	saying	seem	indeed	couldnt	door	taken	deal	
70	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	2	0.005272	0.005285	0.005246	0.010622	0.001062	0.004187	0.002093	0.00
	3	0.006831	0.000000	0.000000	0.006881	0.055051	0.000000	0.000000	0.00
	4	0.005859	0.000000	0.005830	0.005902	0.000000	0.000000	0.011632	0.01
	5	0.008429	0.012674	0.004194	0.012737	0.004246	0.004183	0.004183	0.01
...	...	...	...	...	...	...	...	...	
62739	2	0.016015	0.008027	0.003984	0.004033	0.000000	0.003974	0.003974	0.000
	3	0.045041	0.045152	0.000000	0.000000	0.000000	0.000000	0.000000	0.04
	4	0.005093	0.005106	0.005068	0.010261	0.005131	0.000000	0.000000	0.005
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.013626	0.013626	0.000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.043356	0.000000	0.000

1108 rows × 1000 columns

In [49]:
```
print('max term TFIDF value:', max(TFIDF_sigs.sum(axis=0)))

print('max TFIDF term:', TFIDF_sigs.sum(axis = 0).idxmax())
```

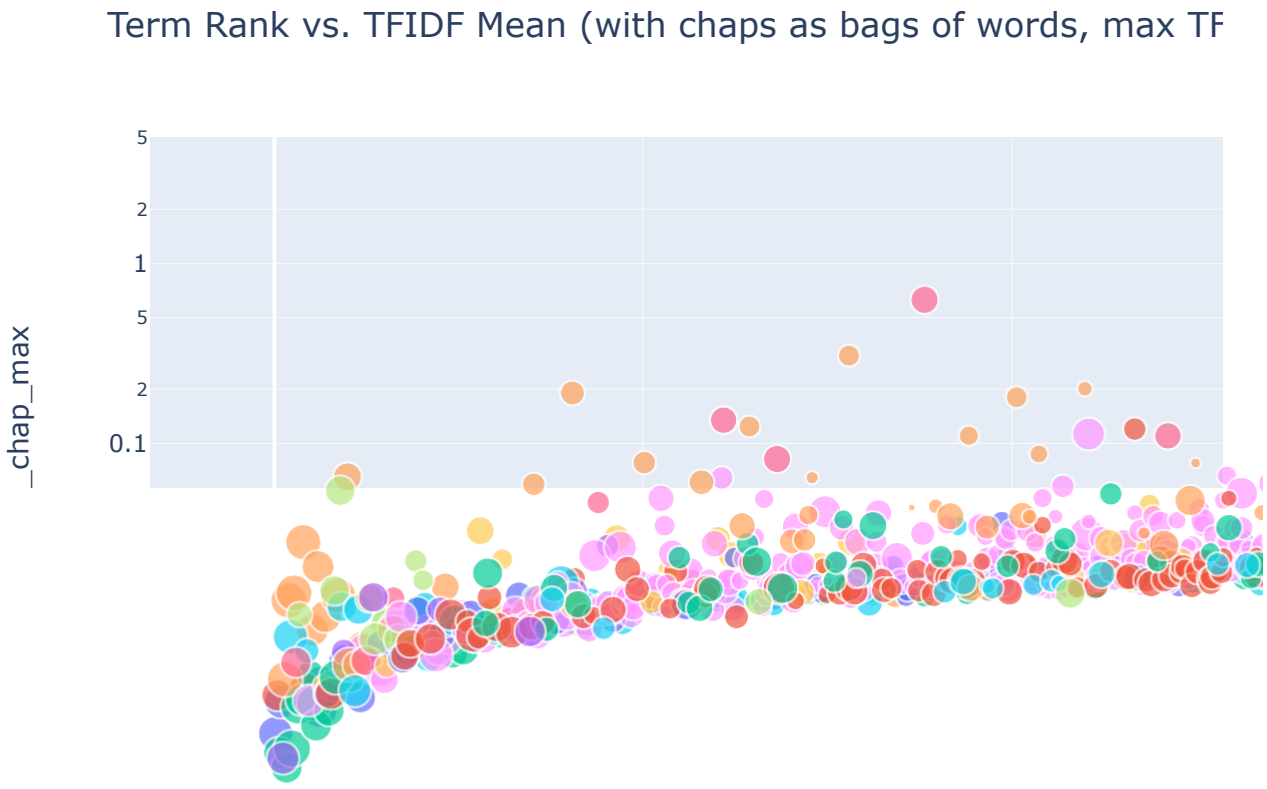
```
max term TFIDF value: 20.442455786925322
max TFIDF term: dont
```

In [50]:
```
print('Max total TFIDF value by book and chapter:', max(TFIDF_sigs.sum(axis=1)))

print('(Book_id, chap_id) with max total TFIDF: ', TFIDF_sigs.sum(axis = 1).idxm

print('Title of book with max total TFIDF:', LIB.loc[TFIDF_sigs.sum(axis = 1).id
```

```
Max total TFIDF value by book and chapter: 10.103244296859762
(Book_id, chap_id) with max total TFIDF:  (102, 4)
Title of book with max total TFIDF: The Tragedy Of Puddnhead Wilson
```

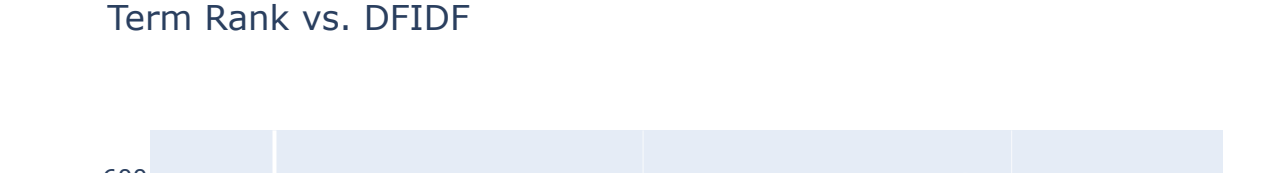
Rank and TFIDF Mean

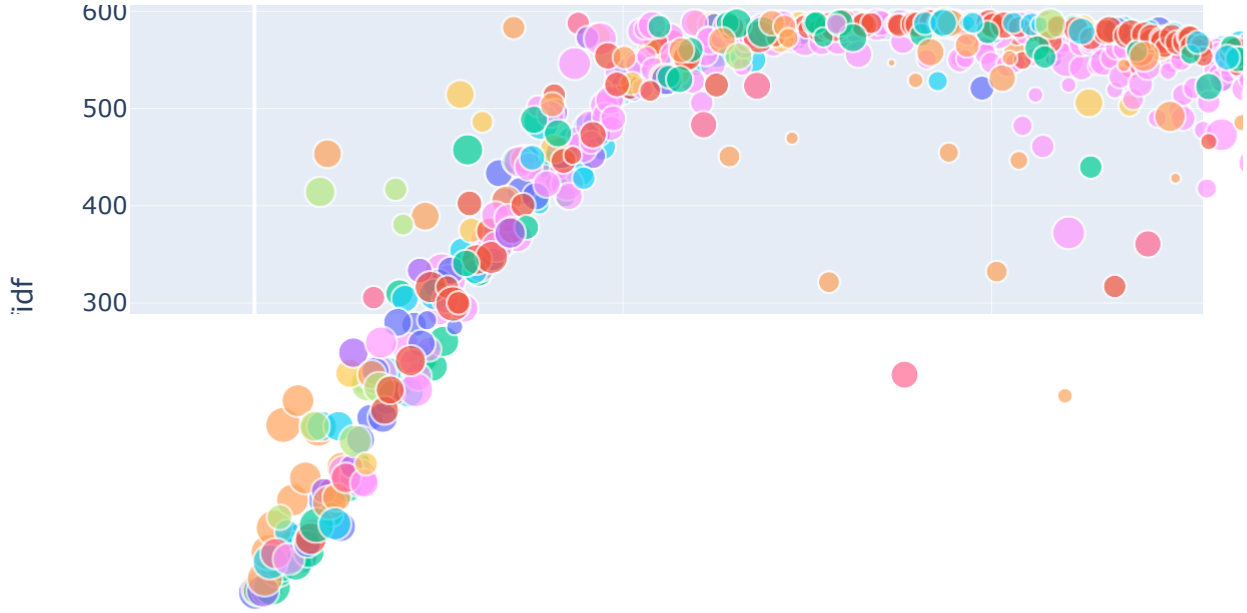
In [51]:
```
px.scatter(VOCAB.reset_index(),
           x = 'term_rank2', y = 'tfidf_mean_chap_max',
           title = 'Term Rank vs. TFIDF Mean (with chaps as bags of words, max T
           color = 'max_pos', size = 'n_pos',
           hover_name = 'term_str', hover_data = ['n', 'i'],
           log_y = True, log_x = False)
```



Rank and DFIDF

In [52]:
```
px.scatter(VOCAB.reset_index(),
           x = 'term_rank2', y = 'dfidf',
           title = 'Term Rank vs. DFIDF',
           color = 'max_pos', size = 'n_pos',
           hover_name = 'term_str', hover_data = ['n', 'i'])
```





M06: Similarity and Clustering

Collapse Bags (to use for clustering)

- #### Note: not the same as computing TFIDF with larger bags

Mean TFIDF for each book for all terms

In [53]:
```
# group by book_id (BOOKS = OHCO[:1])
mean_TFIDF = TFIDF.groupby(BOOKS).mean()

mean_TFIDF
```

Out[53]:

term_str	0	00	01	02	03	04	05	06	
book_id									
70	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
74	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
76	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
86	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
91	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
93	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
102	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
119	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
142	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

term_str	0	00	01	02	03	04	05	06	
book_id									
245	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1044	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1086	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1837	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
2874	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
2875	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
2895	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3171	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3172	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3173	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3176	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3177	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3178	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3179	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3180	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3181	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3182	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3183	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3184	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3185	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3186	0.000000	0.000000	0.000000	0.003557	0.000000	0.000000	0.000000	0.000000	0.0
3188	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3189	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3190	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3191	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3192	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3199	0.002481	0.004061	0.004469	0.004240	0.008474	0.011872	0.003477	0.002089	0.0
3250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
3251	0.002787	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
19484	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
19987	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000899	0.0
33077	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
60900	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
61522	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

term_str	0	00	01	02	03	04	05	06	
book_id									
62636	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
62739	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

45 rows × 53852 columns

Mean TFIDF for all book for 1000 most significant terms only

In [54]:
```
mean_TFIDF_sigs = TFIDF_sigs.groupby(BOOKS).mean()

mean_TFIDF_sigs
```

Out[54]:

term_str	saying	seem	indeed	couldnt	door	taken	deal	fifty	g
book_id									
70	0.002214	0.005691	0.003876	0.005907	0.007669	0.005358	0.004758	0.009586	0.0
74	0.006860	0.005358	0.005679	0.010177	0.018570	0.003364	0.001466	0.002715	0.0
76	0.011842	0.008150	0.000312	0.051429	0.017287	0.000202	0.003498	0.002909	0.0
86	0.004200	0.009052	0.008176	0.028338	0.009332	0.005638	0.007524	0.005537	0.0
91	0.009653	0.012530	0.000000	0.073216	0.001092	0.000000	0.005023	0.005192	0.0
93	0.005316	0.006406	0.000000	0.049434	0.016613	0.000000	0.005417	0.001492	0.0
102	0.014221	0.001967	0.006542	0.020268	0.018873	0.006466	0.004435	0.002955	0.0
119	0.004359	0.008560	0.005958	0.003688	0.006101	0.003824	0.006648	0.008816	0.0
142	0.007268	0.004428	0.005972	0.003653	0.004329	0.004804	0.003849	0.008807	0.0
245	0.003502	0.004068	0.002828	0.006774	0.006710	0.005372	0.006845	0.011155	0.0
1044	0.010437	0.020477	0.004029	0.062185	0.002039	0.000000	0.019964	0.010987	0.0
1086	0.011302	0.003689	0.009535	0.029505	0.002482	0.000000	0.004162	0.001445	0.0
1837	0.012585	0.007149	0.012676	0.000000	0.012856	0.006845	0.003559	0.001468	0.0
2874	0.015162	0.006606	0.010654	0.006011	0.005050	0.005317	0.004128	0.002257	0.0
2875	0.012666	0.003254	0.010652	0.002297	0.001937	0.012761	0.001903	0.004802	0.0
2895	0.001588	0.007035	0.007057	0.004420	0.005550	0.006149	0.005118	0.007155	0.0
3171	0.003821	0.017271	0.027987	0.000000	0.007740	0.013207	0.003190	0.002218	0.0
3172	0.004252	0.008524	0.004231	0.000000	0.000000	0.004220	0.004220	0.021102	0.0
3173	0.015125	0.009352	0.017372	0.000000	0.004296	0.000000	0.002315	0.006945	0.0
3176	0.001588	0.004203	0.000659	0.000562	0.004121	0.004841	0.005115	0.006773	0.0
3177	0.003114	0.005003	0.006568	0.002544	0.005879	0.007228	0.005472	0.011755	0.0
3178	0.005395	0.008850	0.019107	0.004711	0.010841	0.006513	0.009886	0.002842	0.0
3179	0.007909	0.009067	0.014630	0.023126	0.007769	0.003826	0.009298	0.004420	0.0

term_str	saying	seem	indeed	couldnt	door	taken	deal	fifty	g
book_id									
3180	0.009681	0.002094	0.004293	0.004316	0.006482	0.004786	0.001616	0.005246	0.0
3181	0.008897	0.003345	0.020116	0.000000	0.003559	0.011429	0.003507	0.016560	0.0
3182	0.007100	0.004571	0.002464	0.007197	0.003550	0.005521	0.015412	0.009644	0.0
3183	0.000000	0.005688	0.005647	0.005716	0.068597	0.000000	0.011266	0.011266	0.0
3184	0.009640	0.006315	0.010998	0.001801	0.008365	0.005465	0.006711	0.016163	0.0
3185	0.004623	0.008226	0.011253	0.015807	0.003437	0.004922	0.008367	0.000000	0.0
3186	0.012936	0.005536	0.011350	0.011731	0.006083	0.004225	0.000426	0.001164	0.0
3188	0.008052	0.007669	0.006826	0.009173	0.007591	0.005700	0.005608	0.009260	0.0
3189	0.003429	0.005742	0.002021	0.005877	0.010844	0.005124	0.007386	0.006111	0.0
3190	0.005380	0.002985	0.013965	0.000000	0.003403	0.001987	0.002236	0.007621	0.0
3191	0.000000	0.006880	0.005692	0.000000	0.003457	0.005241	0.006813	0.000000	0.0
3192	0.003366	0.000000	0.004816	0.004609	0.004700	0.015009	0.005170	0.006597	0.0
3199	0.004374	0.007408	0.005917	0.007903	0.002941	0.010699	0.008695	0.005646	0.0
3250	0.004133	0.018123	0.002079	0.022892	0.008372	0.002074	0.019889	0.002074	0.0
3251	0.006054	0.010133	0.008176	0.005880	0.005100	0.005336	0.004149	0.004723	0.0
19484	0.000000	0.011282	0.006236	0.000000	0.020720	0.003393	0.006878	0.010598	0.0
19987	0.004113	0.003789	0.002411	0.015717	0.010104	0.005772	0.008337	0.007960	0.0
33077	0.000000	0.011559	0.000000	0.002904	0.000000	0.005723	0.000000	0.000000	0.0
60900	0.010154	0.011673	0.005251	0.015166	0.016750	0.004565	0.015933	0.011696	0.0
61522	0.002880	0.008933	0.008071	0.014109	0.009320	0.005471	0.007913	0.003794	0.0
62636	0.022174	0.000000	0.011033	0.005584	0.000000	0.000000	0.005503	0.005503	0.0
62739	0.011025	0.009714	0.001509	0.002382	0.000855	0.010159	0.002933	0.008294	0.0

45 rows × 1000 columns

DOC Table

- #### Same as LIB table when books are docs

In [55]:
```
book_DOC = pd.DataFrame(index = mean_TFIDF.index)
```

In [56]:
```
book_DOC = book_DOC.join(LIB[['author', 'title']])

book_DOC['label'] = book_DOC.apply(lambda x: f"{x.author.split(',')[0]} {x.name}
```

In [57]:
```
book_DOC
```

Out[57]:

book_id	author	title	label
70	twain	what is man	twain 70: what is man
74	twain	the adventures of tom sawyer	twain 74: the adventures of tom sawyer
76	twain	the adventures of huckleberry finn	twain 76: the adventures of huckleberry finn
86	twain	a connecticut yankee in king arthurs court	twain 86: a connecticut yankee in king arthurs...
91	twain	tom sawyer abroad	twain 91: tom sawyer abroad
93	twain	tom sawyer detective	twain 93: tom sawyer detective
102	twain	the tragedy of puddnhead wilson	twain 102: the tragedy of puddnhead wilson
119	twain	a tramp abroad	twain 119: a tramp abroad
142	twain	the 30000 bequest and other stories	twain 142: the 30000 bequest and other stories
245	twain	life on the mississippi	twain 245: life on the mississippi
1044	twain	extract from captain stormfields visit to Heaven	twain 1044: extract from captain stormfields v...
1086	twain	a horses tale	twain 1086: a horses tale
1837	twain	the prince and the pauper	twain 1837: the prince and the pauper
2874	twain	personal recollections of joan of arc vol 1	twain 2874: personal recollections of joan of ...
2875	twain	personal recollections of joan of arc vol 2	twain 2875: personal recollections of joan of ...
2895	twain	following the equator	twain 2895: following the equator
3171	twain	in defense of harriet shelley	twain 3171: in defense of harriet shelley
3172	twain	fenimore coopers literary offences	twain 3172: fenimore coopers literary offences
3173	twain	essays on paul bourget	twain 3173: essays on paul bourget
3176	twain	the innocents abroad	twain 3176: the innocents abroad
3177	twain	roughing it	twain 3177: roughing it
3178	twain	the gilded age	twain 3178: the gilded age
3179	twain	the american claimant	twain 3179: the american claimant
3180	twain	a double barrelled detective story	twain 3180: a double barrelled detective story
3181	twain	the stolen white elephant	twain 3181: the stolen white elephant
3182	twain	some rambling notes of an idle excursion	twain 3182: some rambling notes of an idle exc...
3183	twain	the facts concerning the recent carnival of cr...	twain 3183: the facts concerning the recent ca...
3184	twain	alonzo fitz and other stories	twain 3184: alonzo fitz and other stories

book_id	author	title	label
3185	twain	those extraordinary twins	twain 3185: those extraordinary twins
3186	twain	the mysterious stranger and other stories	twain 3186: the mysterious stranger and other ...
3188	twain	mark twain speeches	twain 3188: mark twain speeches
3189	twain	sketches new and old	twain 3189: sketches new and old
3190	twain	1601 conversation as it was by the social fire...	twain 3190: 1601 conversation as it was by the...
3191	twain	goldsmiths friend abroad again	twain 3191: goldsmiths friend abroad again
3192	twain	the curious republic of gondour and other whim...	twain 3192: the curious republic of gondour an...
3199	twain	the letters of mark twain	twain 3199: the letters of mark twain
3250	twain	how to tell a story and other essays	twain 3250: how to tell a story and other essays
3251	twain	the man that corrupted hadleyburg and other st...	twain 3251: the man that corrupted hadleyburg ...
19484	twain	editorial wild oats	twain 19484: editorial wild oats
19987	twain	chapters from my autobiography	twain 19987: chapters from my autobiography
33077	twain	the treaty with china its provisions explained	twain 33077: the treaty with china its provisi...
60900	twain	merry tales	twain 60900: merry tales
61522	twain	the 1000000 bank note	twain 61522: the 1000000 bank note
62636	twain	to the person sitting in darkness	twain 62636: to the person sitting in darkness
62739	twain	king leopolds soliloquy	twain 62739: king leopolds soliloquy

Normalized Tables for Clustering

In [58]:
```
# binary table
L0 = mean_TFIDF_sigs.astype('bool').astype('int')

# Manhattan distance (L1 norm): divide each value by sum down cols
L1 = mean_TFIDF_sigs.apply(lambda x: x / x.sum(), 1)

# Euclidean distance (L2 norm)
L2 = mean_TFIDF_sigs.apply(lambda x: x / norm(x), 1) # Euclidean
```

In [59]:
```
assert round(L1.sum(1).sum()) == len(mean_TFIDF_sigs)
```

In [60]:
```
assert round(((L2.T)**2).sum().sum()) == len(mean_TFIDF_sigs)
```

Create table of book pairs (doc pair table PAIRS)

In [61]:
```
mean_TFIDF_sigs.T.corr().stack()
```

Out[61]:
```
book_id  book_id
70       70          1.000000
         74          0.065653
         76         -0.006768
         86          0.079010
         91         -0.002018
                        ...
62739    33077       0.158020
         60900      -0.028375
         61522       0.001523
         62636       0.089411
         62739       1.000000
Length: 2025, dtype: float64
```

In [62]:
```
# correlation between books --> stack and convert to df with col for  raw correl
PAIRS = 1 - mean_TFIDF_sigs.T.corr().stack().to_frame('corr_raw')

# rename indices
PAIRS.index.names = ['doc_a', 'doc_b']

# remove identities (e.g., corr(105, 105) and reverse dupliciates (e.g., corr(10
PAIRS = PAIRS.query("doc_a > doc_b")
```

In [63]:
```
PAIRS
```

Out[63]:

		corr_raw
doc_a	doc_b	
74	70	0.934347
76	70	1.006768
	74	0.500537
86	70	0.920990
	74	0.857285
...	...	...
62739	19987	0.915541
	33077	0.841980
	60900	1.028375
	61522	0.998477
	62636	0.910589

990 rows × 1 columns

Compute distance measures between all pairs of books using `pdist()`

In [64]:
```
combos  = [
    (mean_TFIDF_sigs, 'cityblock', 'cityblock–raw'),
    (mean_TFIDF_sigs, 'euclidean', 'euclidean–raw'),
    (L2, 'euclidean', 'euclidean–l2'),
    (mean_TFIDF_sigs, 'cosine', 'cosine–raw'),
    (L1, 'cityblock', 'cityblock–l1'),
    (L0, 'jaccard', 'jaccard–l0'),
    (L0, 'jensenshannon', 'js–l0'),
    (L1, 'jensenshannon', 'js–l1'),
    (L2, 'jensenshannon', 'js–l2'),
]
```

In [65]:
```
for X, metric, label in combos:
    PAIRS[label] = pdist(X, metric)
```

Compare Distributions

In [66]:
```
PAIRS.head(20)
```

Out[66]:

		corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0	
doc_a	doc_b								
74	70	0.934347	4.546088	0.234767	0.939812	0.441623	0.810756	0.037000	0.1
76	70	1.006768	4.975003	0.297118	1.093061	0.597391	0.996911	0.128000	0.2
	74	0.500537	3.477613	0.178243	0.844661	0.356726	0.648449	0.008000	0.0
86	70	0.920990	4.889381	0.292809	1.114627	0.621196	1.037877	0.233233	0.2
	74	0.857285	5.511088	0.338610	1.151917	0.663457	1.127231	0.311000	0.3
	76	0.873024	4.661049	0.244347	0.926136	0.428864	0.754032	0.043000	0.1
91	70	1.002018	3.006633	0.140112	0.758177	0.287416	0.623418	0.006000	0.0
	74	0.504966	3.346067	0.161667	0.766304	0.293611	0.635085	0.009000	0.0
	76	0.176894	3.242933	0.186845	0.881473	0.388497	0.644574	0.007000	0.0
	86	0.851959	6.592539	0.435993	1.107597	0.613386	1.009056	0.301603	0.3
93	70	1.028131	4.887719	0.249661	1.002306	0.502308	0.993955	0.259519	0.3
	74	0.562528	4.077018	0.188483	0.923076	0.426035	0.823772	0.052000	0.1
	76	0.168563	3.664222	0.168963	0.841515	0.354074	0.723915	0.030030	0.1
	86	0.885217	3.639949	0.174221	0.860308	0.370065	0.747925	0.053000	0.1
	91	0.261414	2.630389	0.129681	0.712435	0.253782	0.567135	0.005000	0.0
102	70	0.958330	5.277857	0.308049	0.978975	0.479196	0.912322	0.310241	0.3
	74	0.531515	5.342585	0.279884	1.101894	0.607085	1.197783	0.582915	0.5
	76	0.559315	6.213281	0.375334	0.995281	0.495292	0.985214	0.393180	0.4
	86	0.892180	3.245179	0.158915	0.843261	0.355545	0.689906	0.006000	0.0

		corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0	
doc_a	doc_b								
	91	0.586890	3.370931	0.159585	0.833132	0.347054	0.687526	0.006000	0.0

Hierarchical agglomerative cluster diagrams for the distance measures

In [67]:
```
LIB['label'] = book_DOC['label']
```

In [140…
```
def hca(sims, title="My Dendrogram", linkage_method='weighted', color_thresh=Non

    # calculate linkage using given method
    tree = sch.linkage(sims, method=linkage_method)

    # extract labels (title, year)
    labels = LIB.label.values

    # set color threshold
    if not color_thresh:
        color_thresh = pd.DataFrame(tree)[2].median()

    # plot dendrograms for each distance metric and linkage method
    plt.figure()
    fig, axes = plt.subplots(figsize=figsize)
    dendrogram = sch.dendrogram(tree,
                                labels=labels,
                                orientation="left",
                                count_sort=True,
                                distance_sort=True,
                                above_threshold_color='.75',
                                color_threshold=color_thresh
                               )
    plt.tick_params(axis='both', which='major', labelsize=14)
    fig.suptitle(title, fontsize=20)
```

In [141…
```
for combo in combos:
    # column in df (i.e., distance metric)
    m = combo[-1]

    # two lnikage methods
    for l in ['ward','weighted']:
        # title: distance metric - linkage method
        title = f"{m}—{l}"
        hca(PAIRS[m], title, linkage_method=l)
```

```
/var/folders/3n/4b11y5qn5cn20kztppfbsxq40000gn/T/ipykernel_24732/163819248.py:1
4: RuntimeWarning:

More than 20 figures have been opened. Figures created through the pyplot interf
ace (`matplotlib.pyplot.figure`) are retained until explicitly closed and may co
nsume too much memory. (To control this warning, see the rcParam `figure.max_ope
```

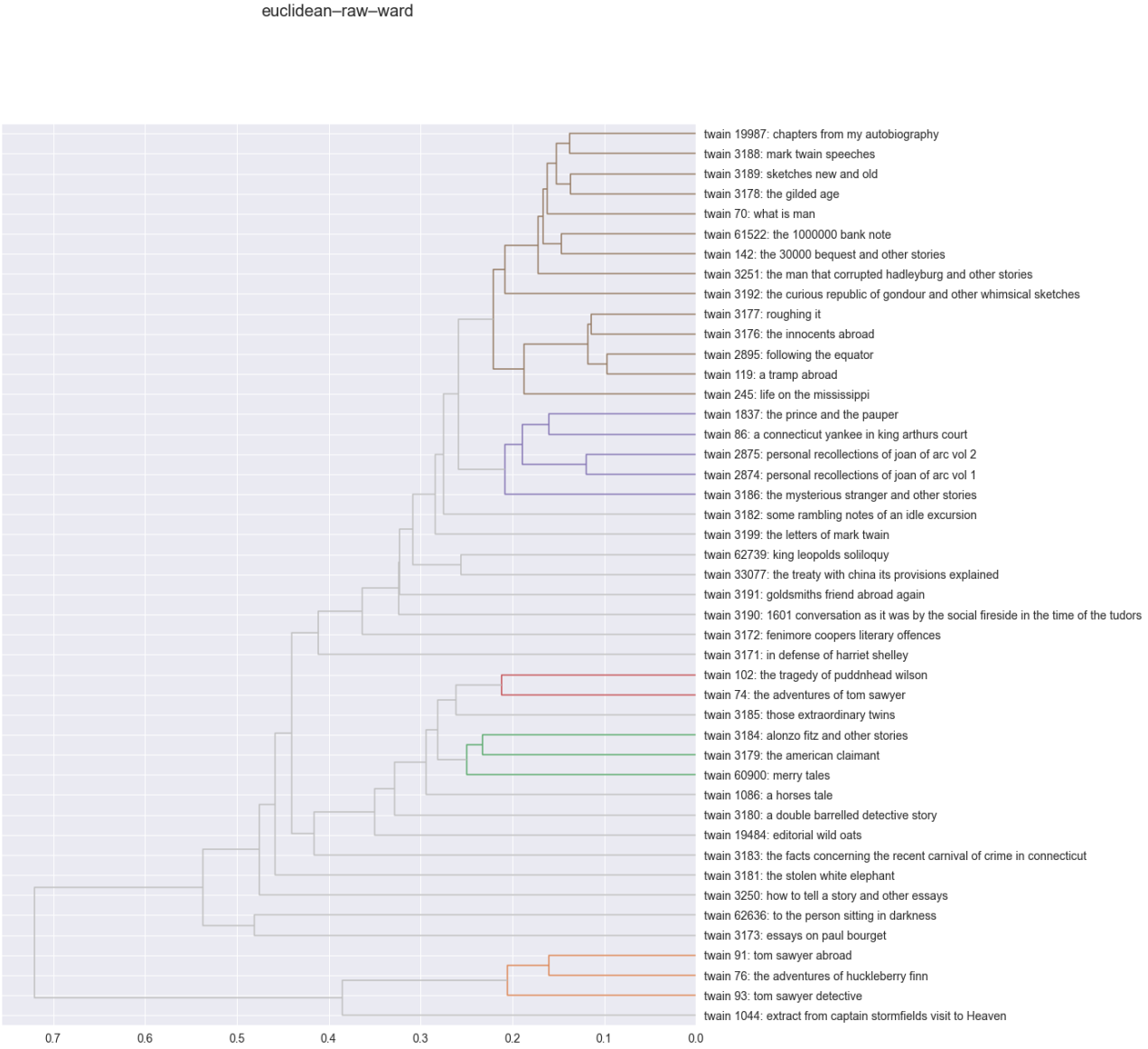
```
n_warning`).
```

```
<Figure size 432x288 with 0 Axes>
```


```
<Figure size 432x288 with 0 Axes>
```

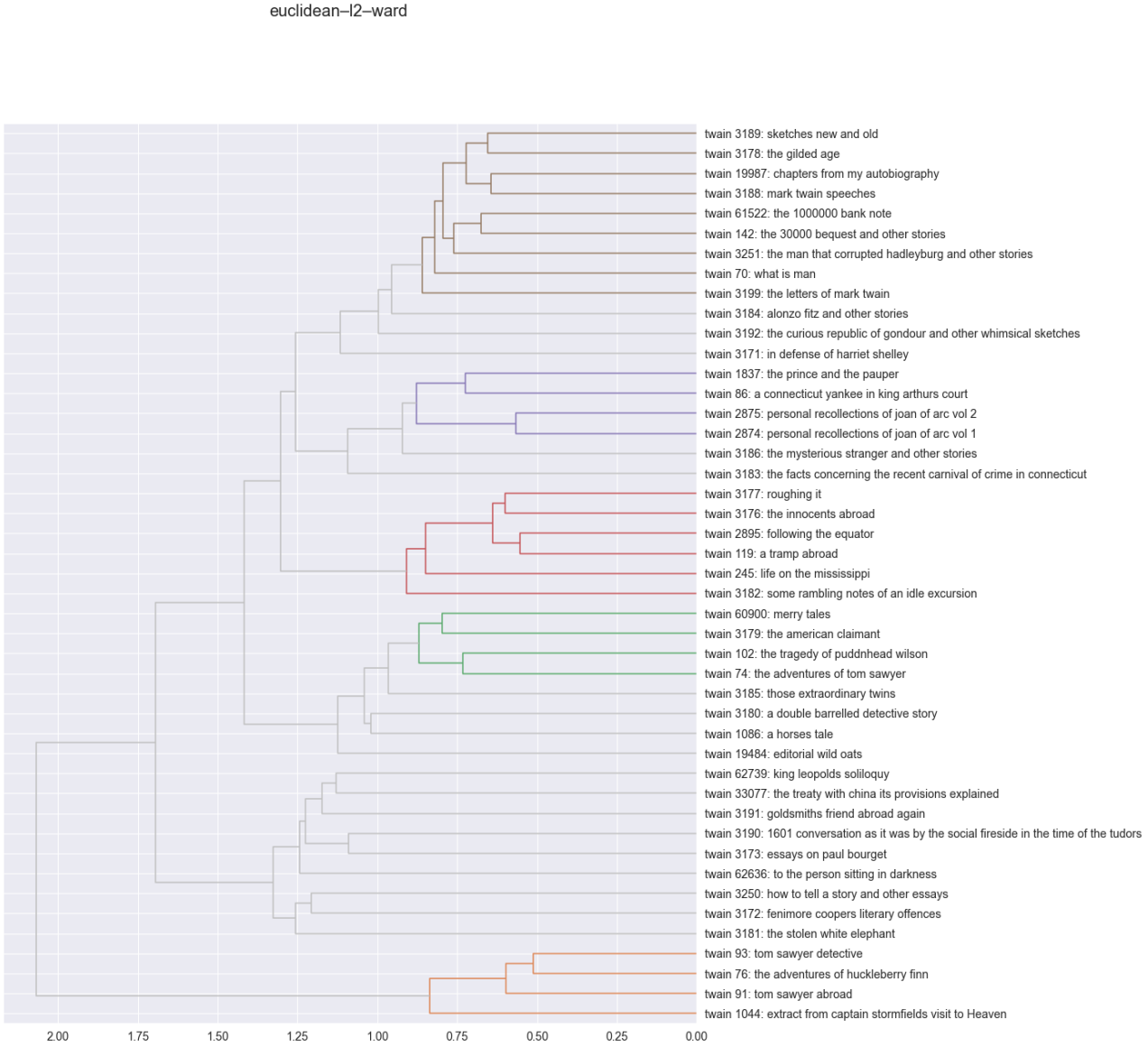
cityblock–raw–weighted

<Figure size 432x288 with 0 Axes>




```
<Figure size 432x288 with 0 Axes>
```

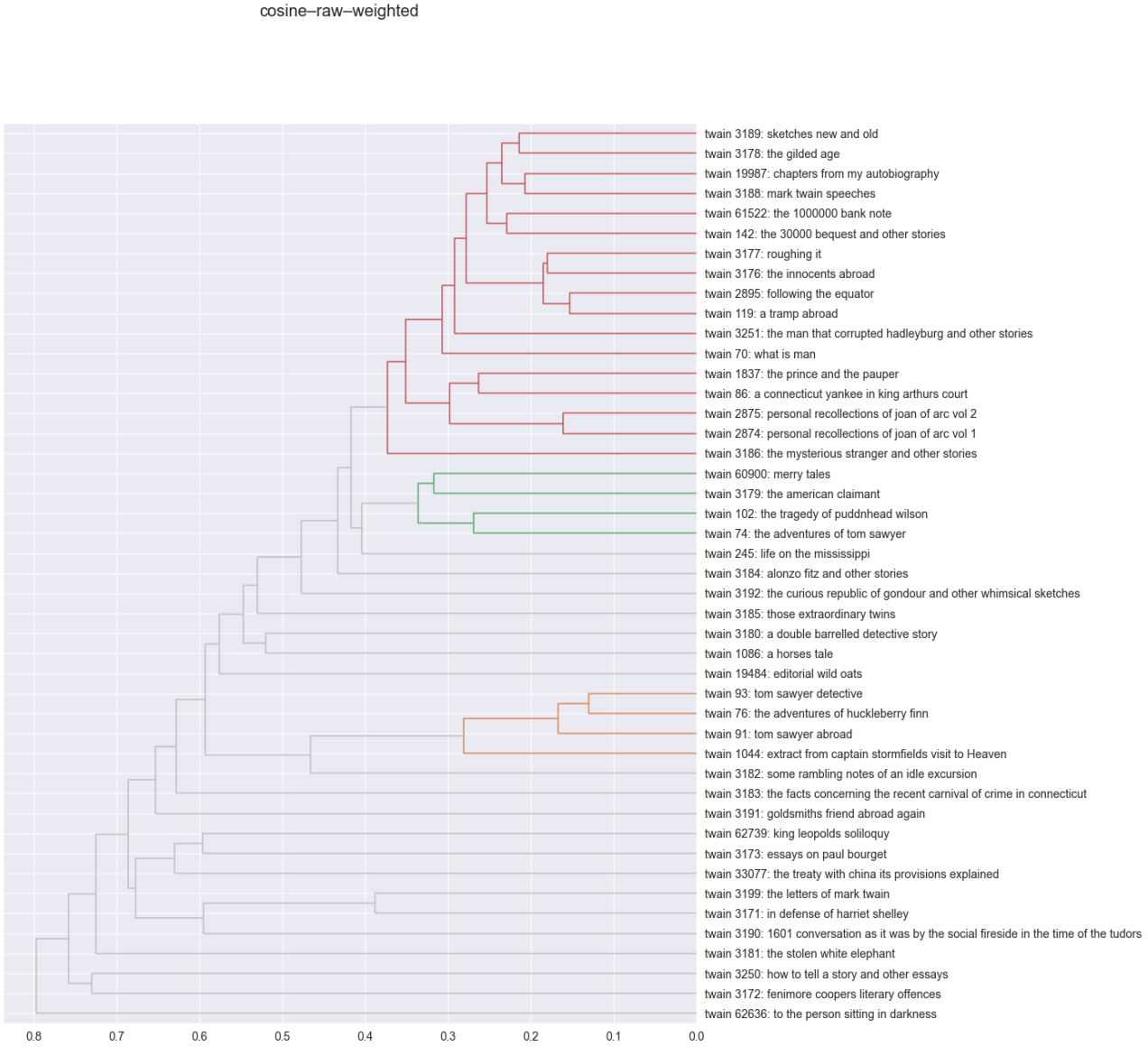

```
<Figure size 432x288 with 0 Axes>
```



<Figure size 432x288 with 0 Axes>


```
<Figure size 432x288 with 0 Axes>
```


```
<Figure size 432x288 with 0 Axes>
```



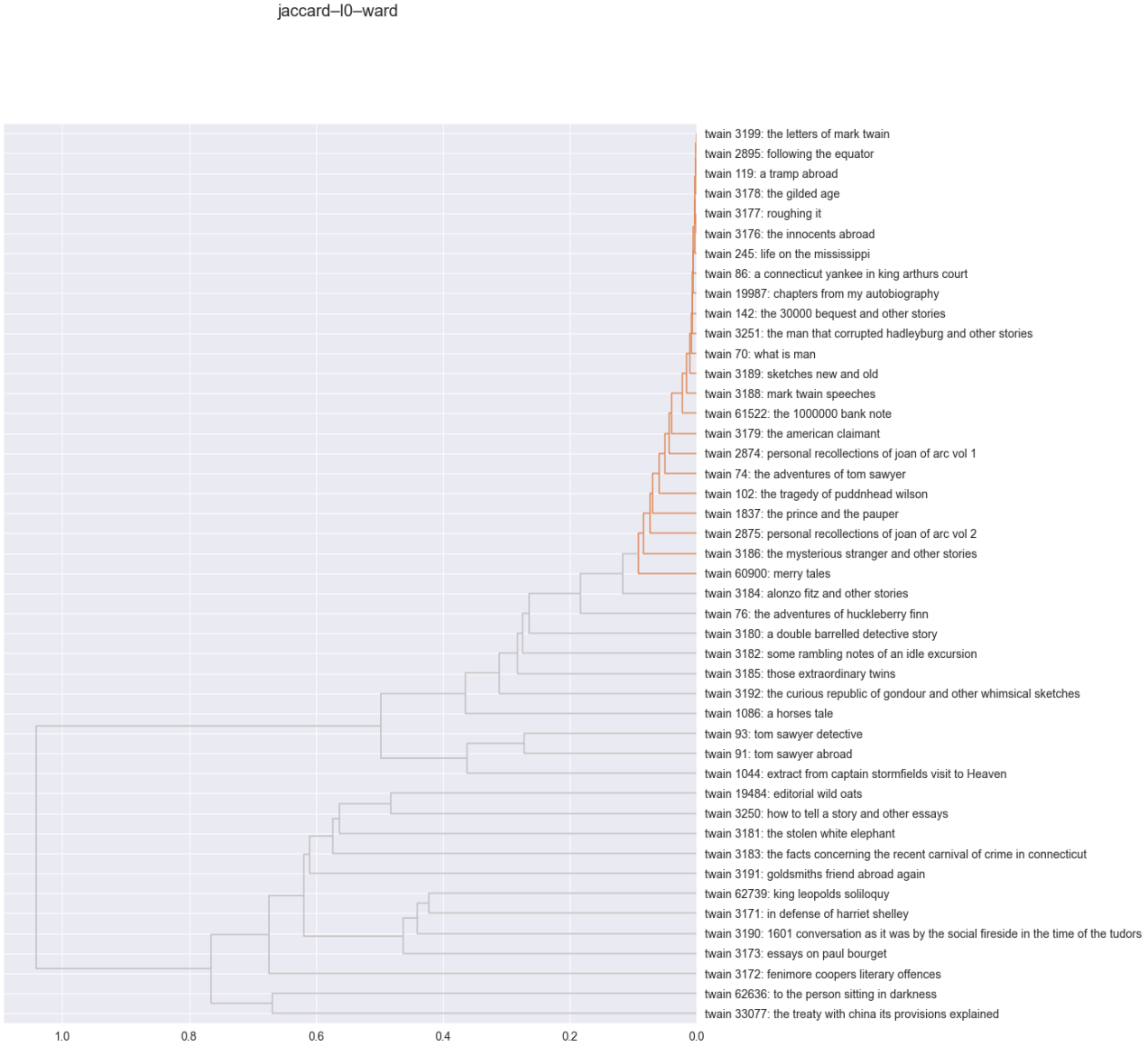

```
<Figure size 432x288 with 0 Axes>
```

cityblock–l1–ward

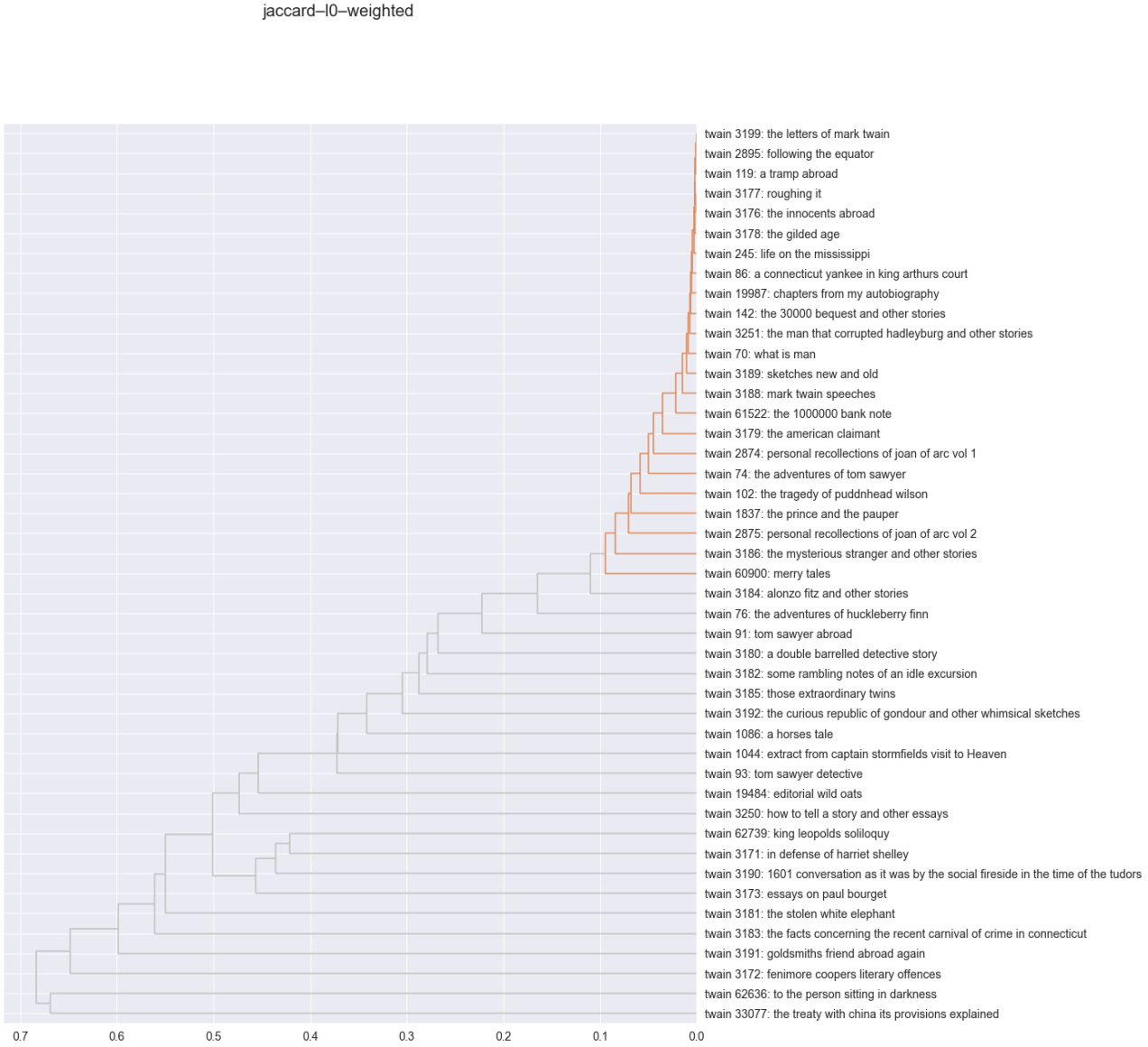
<Figure size 432x288 with 0 Axes>

cityblock–l1–weighted

<Figure size 432x288 with 0 Axes>

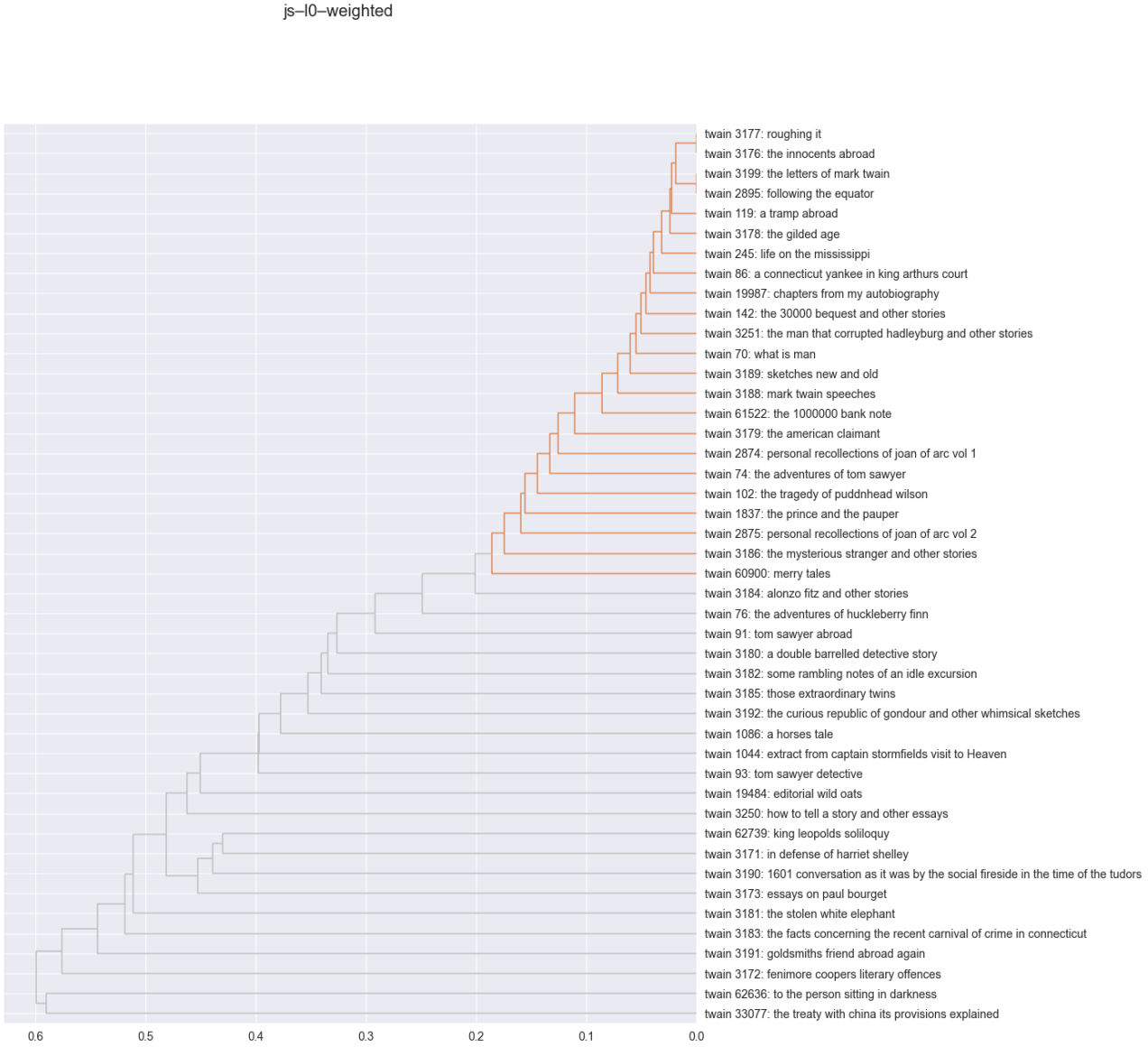



```
<Figure size 432x288 with 0 Axes>
```



<Figure size 432x288 with 0 Axes>

<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>


```
<Figure size 432x288 with 0 Axes>
```


```
<Figure size 432x288 with 0 Axes>
```


```
<Figure size 432x288 with 0 Axes>
```

Top 20 nouns by DFIDF, sorted in descending order (including plural nouns but not proper nouns)

In [70]:
```
# noun taglist (excluding proper nouns)
noun_tags = ['NN', 'NNS']

SIGS.loc[SIGS.max_pos.isin(noun_tags)].sort_values('dfidf', ascending = False).h
```

Out[70]:

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
indeed	373	758	6	0.000255	11.936159	NN	12	{'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'POS', 'IN'...	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
door	319	885	4	0.000298	11.712680	NN	8	{'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'FW', 'JJ',...	0	
deal	414	682	4	0.000230	12.088585	NN	4	{'VBP', 'VB', 'NN', 'NNP'}	0	
children	317	890	8	0.000300	11.704552	NNS	9	{'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'VBN', 'VBP...	0	
money	244	1226	5	0.000413	11.242470	NN	12	{'PDT', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'VBN...	0	
ones	402	704	4	0.000237	12.042782	NNS	16	{'NNPS', 'RP', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP...	0	
miles	263	1117	5	0.000376	11.376800	NNS	9	{'PDT', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'VBP...	0	
everybody	339	842	9	0.000283	11.784537	NN	12	{'RBR', 'RP', 'VBG', 'NNS', 'VB', 'VBD', 'NNP'...	0	
family	297	988	6	0.000333	11.553846	NN	10	{'VBZ', 'NNS', 'VBD', 'NNP', 'VBN', 'RB', 'MD'...	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
ground	381	742	6	0.000250	11.966938	NN	9	{'NNS', 'VB', 'VBD', 'NNP', 'VBN', 'VBG', 'VBP...	0	
friend	344	817	6	0.000275	11.828021	NN	11	{'PDT', 'VBZ', 'NNS', 'VB', 'VBD', 'NNP', 'IN'...	0	
none	362	781	4	0.000263	11.893034	NN	11	{'NNS', 'VB', 'CC', 'NNP', 'IN', 'VBN', 'RB', ...	0	
chance	386	727	6	0.000245	11.996402	NN	4	{'VBP', 'NNP', 'JJ', 'NN'}	0	
state	348	812	5	0.000273	11.836877	NN	5	{'VBZ', 'VB', 'NNP', 'JJ', 'NN'}	0	
friends	336	846	7	0.000285	11.777699	NNS	12	{'NNPS', 'PDT', 'VBZ', 'NNS', 'VB', 'VBD', 'NN...	0	
air	431	661	3	0.000223	12.133707	NN	9	{'VBZ', 'VB', 'NNP', 'IN', 'FW', 'RB', 'VBP', ...	0	
city	267	1108	4	0.000373	11.388471	NN	7	{'NNS', 'VB', 'NNP', 'VBP', 'JJ', 'NN', 'CD'}	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
words	268	1107	5	0.000373	11.389774	NNS	15	{'PDT', 'VBZ', 'NNS', 'WP', 'VB', 'VBD', 'NNP'...	0	
hours	316	901	5	0.000303	11.686830	NNS	9	{'PDT', 'NNS', 'VB', 'VBD', 'NNP', 'VBP', 'JJ'...	0	
nobody	395	711	6	0.000239	12.028507	NN	8	{'VBZ', 'NNS', 'VB', 'NNP', 'RB', 'VBP', 'JJ',...	0	

In [71]:
```
top_20_nouns = list(VOCAB.loc[VOCAB.max_pos.isin(noun_tags)].sort_values('dfidf'
```

In [72]:
```
print(top_20_nouns)
```

```
['indeed', 'door', 'deal', 'children', 'money', 'ones', 'miles', 'family', 'ever
ybody', 'ground', 'friend', 'state', 'none', 'friends', 'chance', 'air', 'city',
'words', 'hours', 'nobody']
```

Most "Significant" Book based on mean TFIDF

In [73]:
```
BOW.groupby(BOOKS).mean().sort_values('tfidf', ascending = False).join(LIB, on =
```

Out[73]:

book_id	n	tf	tfidf	source_file_path	title
3188	2.540928	0.054027	0.078733	Twain/3188-mark_twain_speeches.txt	mark twain speeches
3191	2.330049	0.043699	0.062809	Twain/3191-goldsmiths_friend_abroad_again.txt	goldsmiths friend abroad again
1086	2.659144	0.040938	0.059174	Twain/1086-a_horses_tale.txt	a horses tale
3192	2.431230	0.032768	0.057134	Twain/3192-the_curious_republic_of_gondour_and...	the curious republic of gondour and other whim...

book_id	n	tf	tfidf	source_file_path	title	
3250	2.963259	0.032567	0.051521	Twain/3250-how_to_tell_a_story_and_other_essay...	how to tell a story and other essays	HC
3189	2.875911	0.027712	0.045378	Twain/3189-sketches_new_and_old.txt	sketches new and old	MY
102	3.021061	0.026733	0.043308	Twain/102-the_tragedy_of_puddnhead_wilson.txt	the tragedy of puddnhead wilson	Wh
19484	2.700829	0.026177	0.042429	Twain/19484-editorial_wild_oats.txt	editorial wild oats	^[
74	2.786740	0.026721	0.041384	Twain/74-the_adventures_of_tom_sawyer.txt	the adventures of tom sawyer	
3180	2.894254	0.028208	0.040873	Twain/3180-a_double_barrelled_detective_story.txt	a double barrelled detective story	
3185	2.962158	0.026052	0.039382	Twain/3185-those_extraordinary_twins.txt	those extraordinary twins	
3190	2.793029	0.015731	0.037841	Twain/3190-1601_conversation_as_it_was_by_the_...	1601 conversation as it was by the social fire...	
3179	3.006103	0.026183	0.037614	Twain/3179-the_american_claimant.txt	the american claimant	
62739	2.846308	0.019321	0.037133	Twain/62739-king_leopolds_soliloquy.txt	king leopolds soliloquy	
2875	2.898887	0.026507	0.037126	Twain/2875-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 2	
3181	2.742546	0.021657	0.036272	Twain/3181-the_stolen_white_elephant.txt	the stolen white elephant	
3177	2.671287	0.021494	0.036157	Twain/3177-roughing_it.txt	roughing it	

book_id	n	tf	tfidf	source_file_path	title
3184	3.026914	0.023980	0.035678	Twain/3184-alonzo_fitz_and_other_stories.txt	alonzo fitz and other stories
1837	2.736118	0.021699	0.035655	Twain/1837-the_prince_and_the_pauper.txt	the prince and the pauper
2874	3.034438	0.024597	0.033642	Twain/2874-personal_recollections_of_joan_of_a...	personal recollections of joan of arc vol 1
60900	3.801300	0.014654	0.033563	Twain/60900-merry_tales.txt	merry tales
245	2.905452	0.021770	0.033280	Twain/245-life_on_the_mississippi.txt	life on the mississippi
76	3.836670	0.026139	0.033157	Twain/76-the_adventures_of_huckleberry_finn.txt	the adventures of huckleberry finn
93	3.698687	0.027781	0.033141	Twain/93-tom_sawyer_detective.txt	tom sawyer detective
3178	3.009788	0.022064	0.032297	Twain/3178-the_gilded_age.txt	the gilded age
86	3.058316	0.022648	0.031542	Twain/86-a_connecticut_yankee_in_king_arthurs_...	a connecticut yankee in king arthurs court
91	3.753066	0.023020	0.029092	Twain/91-tom_sawyer_abroad.txt	tom sawyer abroad
142	3.774277	0.018769	0.028576	Twain/142-the_30000_bequest_and_other_stories.txt	the 30000 bequest and other stories
2895	2.888958	0.016800	0.027830	Twain/2895-following_the_equator.txt	following the equator
119	2.962312	0.016666	0.026501	Twain/119-a_tramp_abroad.txt	a tramp abroad
3173	3.853003	0.017103	0.026088	Twain/3173-essays_on_paul_bourget.txt	essays on paul bourget
3171	3.679526	0.015155	0.026074	Twain/3171-in_defense_of_harriet_shelley.txt	in defense of harriet shelley

book_id	n	tf	tfidf	source_file_path	title
19987	3.472678	0.018381	0.026042	Twain/19987-chapters_from_my_autobiography.txt	chapters from my autobiography
3176	2.915266	0.015238	0.025448	Twain/3176-the_innocents_abroad.txt	the innocents abroad
3186	3.631648	0.020284	0.025365	Twain/3186-the_mysterious_stranger_and_other_s...	the mysterious stranger and other stories
62636	3.318565	0.012764	0.022179	Twain/62636-to_the_person_sitting_in_darkness.txt	to the person sitting in darkness
3182	3.176685	0.013717	0.021264	Twain/3182-some_rambling_notes_of_an_idle_excu...	some rambling notes of an idle excursion
3183	3.909091	0.015390	0.020693	Twain/3183-the_facts_concerning_the_recent_car...	the facts concerning the recent carnival of cr...
70	4.404473	0.012261	0.019698	Twain/70-what_is_man.txt	what is man
1044	4.635578	0.013983	0.019182	Twain/1044-extract_from_captain_stormfields_vi...	extract from captain stormfields visit to Heaven
3172	3.585507	0.010577	0.018147	Twain/3172-fenimore_coopers_literary_offences.txt	fenimore coopers literary offences
3251	4.639982	0.011230	0.016951	Twain/3251-the_man_that_corrupted_hadleyburg_a...	the man that corrupted hadleyburg and other st...
3199	4.475668	0.010489	0.016611	Twain/3199-the_letters_of_mark_twain.txt	the letters of mark twain
33077	3.772847	0.007546	0.014359	Twain/33077-the_treaty_with_china_its_provisio...	the treaty with china its provisions explained

book_id	n	tf	tfidf	source_file_path	title
61522	4.577857	0.007410	0.011398	Twain/61522-the_1000000_bank_note.txt	the 1000000 bank note

Compare Distributions

In [74]:
```
# merge PAIRS, LIB to add label col twice (for doc_a, doc_b) to include author,

DISTS = pd.merge(PAIRS.reset_index(), LIB['label'], left_on = 'doc_a', right_on

DISTS = pd.merge(DISTS, LIB['label'], left_on = 'doc_b', right_on = 'book_id', h

DISTS = DISTS.set_index(['doc_a', 'doc_b']).rename({'label_x': 'label_a', 'label
```

In [75]:
```
# reorder df columns so that label_a and label_b first

DISTS.insert(loc = 0, column = 'label_a', value = DISTS.pop('label_a'))

DISTS.insert(loc = 1, column = 'label_b', value = DISTS.pop('label_b'))
```

In [76]:
```
DISTS.head(20).style.background_gradient(cmap='YlGnBu', high=.5, axis=0)
```

Out[76]:

doc_a	doc_b	label_a	label_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	ci
74.0	**70.0**	twain 74: the adventures of tom sawyer	twain 70: what is man	0.934347	4.546088	0.234767	0.939812	0.441623	
76.0	**70.0**	twain 76: the adventures of huckleberry finn	twain 70: what is man	1.006768	4.975003	0.297118	1.093061	0.597391	
86.0	**70.0**	twain 86: a connecticut yankee in king arthurs court	twain 70: what is man	0.920990	4.889381	0.292809	1.114627	0.621196	
91.0	**70.0**	twain 91: tom sawyer abroad	twain 70: what is man	1.002018	3.006633	0.140112	0.758177	0.287416	
93.0	**70.0**	twain 93: tom sawyer detective	twain 70: what is man	1.028131	4.887719	0.249661	1.002306	0.502308	

doc_a	doc_b	label_a	label_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	ci
102.0	**70.0**	twain 102: the tragedy of puddnhead wilson	twain 70: what is man	0.958330	5.277857	0.308049	0.978975	0.479196	
119.0	**70.0**	twain 119: a tramp abroad	twain 70: what is man	0.889692	4.536682	0.225397	0.870263	0.378679	
142.0	**70.0**	twain 142: the 30000 bequest and other stories	twain 70: what is man	0.749916	3.710108	0.191821	0.896461	0.401821	
245.0	**70.0**	twain 245: life on the mississippi	twain 70: what is man	0.914644	3.392531	0.163527	0.783583	0.307001	
1044.0	**70.0**	twain 1044: extract from captain stormfields visit to Heaven	twain 70: what is man	0.987545	3.968964	0.228053	0.881571	0.388584	
1086.0	**70.0**	twain 1086: a horses tale	twain 70: what is man	0.891234	4.154492	0.229774	0.903064	0.407763	
1837.0	**70.0**	twain 1837: the prince and the pauper	twain 70: what is man	1.016241	6.885359	0.392070	1.178571	0.694515	
2874.0	**70.0**	twain 2874: personal recollections of joan of arc vol 1	twain 70: what is man	0.921028	5.983874	0.382246	1.125937	0.633867	
2875.0	**70.0**	twain 2875: personal recollections of joan of arc vol 2	twain 70: what is man	0.874359	4.595717	0.274207	0.993646	0.493666	
2895.0	**70.0**	twain 2895: following the equator	twain 70: what is man	0.746805	4.854033	0.274275	0.973064	0.473426	
3171.0	**70.0**	twain 3171: in defense of harriet shelley	twain 70: what is man	0.810398	6.024164	0.419678	1.190627	0.708796	

doc_a	doc_b	label_a	label_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	ci
3172.0	**70.0**	twain 3172: fenimore coopers literary offences	twain 70: what is man	0.944777	5.018517	0.245490	0.947697	0.449065	(
3173.0	**70.0**	twain 3173: essays on paul bourget	twain 70: what is man	0.785908	4.344996	0.232379	0.910531	0.414533	(
3176.0	**70.0**	twain 3176: the innocents abroad	twain 70: what is man	0.936830	5.354566	0.267776	0.888264	0.394507	(
3177.0	**70.0**	twain 3177: roughing it	twain 70: what is man	0.981081	5.007724	0.273885	0.941828	0.443520	(

Compare Z normalized distributions

In [77]:
```
ZPAIRS = (PAIRS - PAIRS.mean()) / PAIRS.std()
```

In [78]:
```
ZPAIRS
```

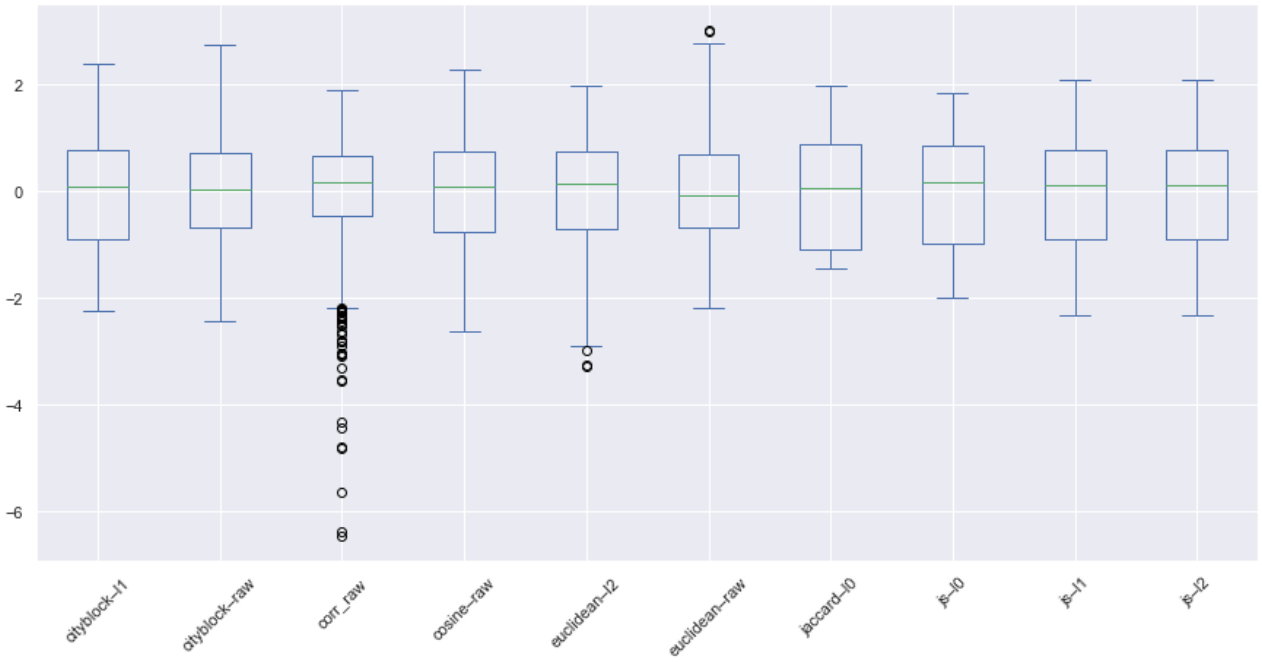
Out[78]:

doc_a	doc_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0
74	**70**	0.201052	-0.616593	-0.623747	-0.440606	-0.516566	-0.710584	-1.275181
76	**70**	0.831302	-0.291196	0.087454	0.572656	0.538375	0.064428	-0.831873
	74	-3.574216	-1.427193	-1.268485	-1.069729	-1.091531	-1.386310	-1.416455
86	**70**	0.084815	-0.356153	0.038305	0.715245	0.699595	0.234977	-0.319228
	74	-0.469586	0.115506	0.560733	0.961805	0.985803	0.606981	0.059614
...	**...**	...	...	...	...	...	...	...
62739	**19987**	0.037397	1.906864	1.518612	1.177751	1.244211	1.500613	1.585456
	33077	-0.602780	0.723475	0.241571	0.926680	0.944455	0.780878	0.284041
	60900	1.019339	1.072599	1.019166	0.915623	0.931478	1.284908	1.583729
	61522	0.759148	-0.323417	-0.658215	0.479398	0.434635	0.528006	0.242222
	62636	-0.005703	0.528097	1.206798	1.262290	1.347341	1.425303	1.582722

990 rows × 10 columns

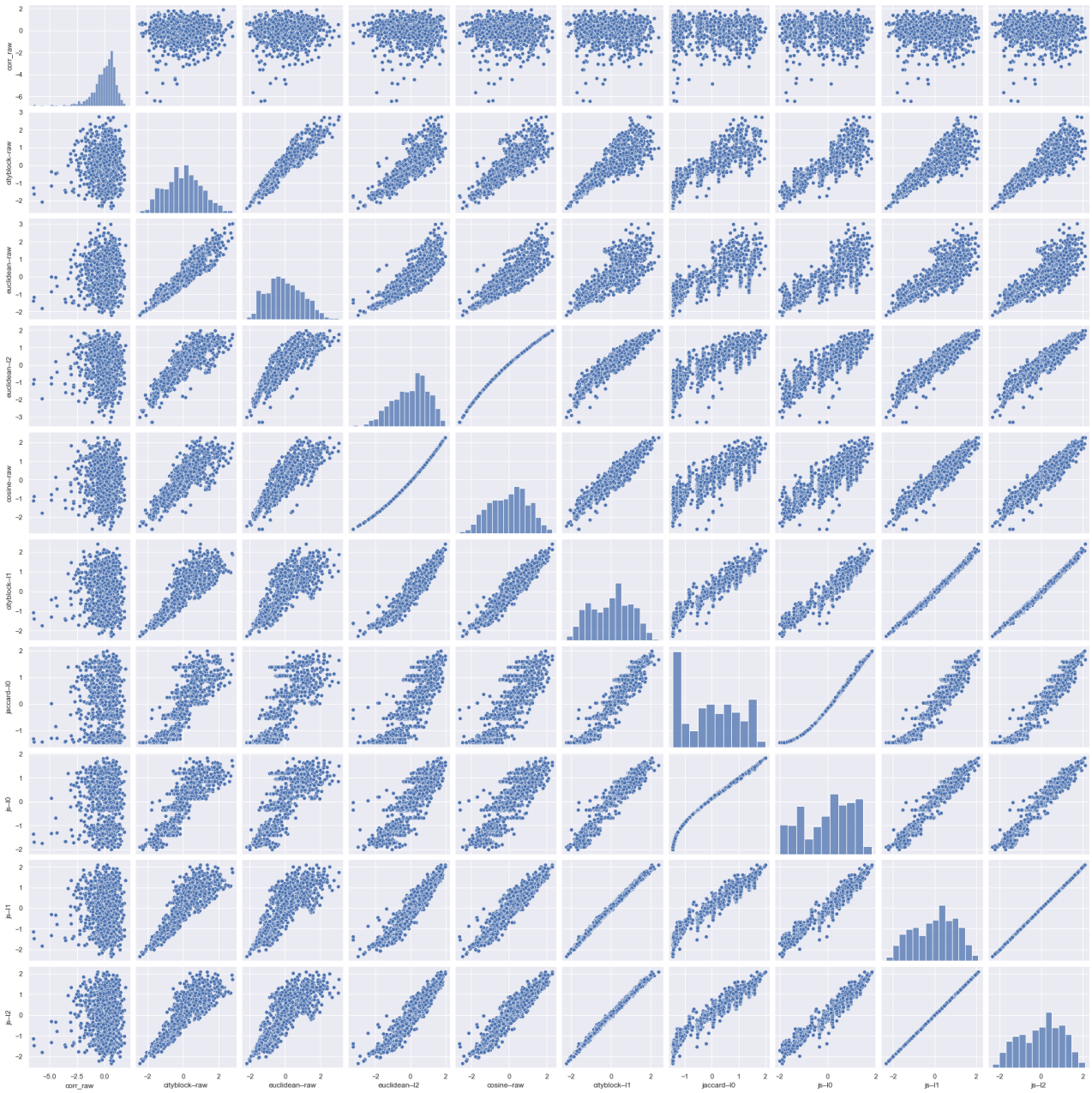
In [79]:
```
```

```
ZPAIRS.T.sort_index().T.plot.box(rot = 45, figsize = (15,7));
```



In [80]:

```
sns.pairplot(ZPAIRS);
```



K-Means

Alogirthm Overview

- Goal: partition n observations of d-dimenstional real vectors so as to minimize the within-cluster sum of squares (WCSS), or variance
- Steps
  - Given an initial set of k means
  - Assignment setep: assign each observation to cluster with the nearest mean (centroid: multivariate mean in Euclidean space) based on Euclidean distance
    - Euclidean distance: $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
    - Important to scale features to prevent those on larger scales from dominating
  - Update step: recalculate the means (centroids) for observations assigned to each cluster

- Converged when assignemnts no longer change

In [81]:
```
n_clusters = 4
```

In [82]:
```
# instantiate KMeans model
km = KMeans(n_clusters, random_state = 314)

# compute cluster centers and predict cluster index for each sample using raw an
book_DOC['y_raw'] = km.fit_predict(mean_TFIDF_sigs)
book_DOC['y_L0'] = km.fit_predict(L0)
book_DOC['y_L1'] = km.fit_predict(L1)
book_DOC['y_L2'] = km.fit_predict(L2)
```

In [83]:
```
book_DOC.iloc[:,1:].sort_values('label').style.background_gradient(cmap = 'RdBu'
```

Out[83]:

book_id	title	label	y_raw	y_L0	y_L1	y_L2
102	the tragedy of puddnhead wilson	twain 102: the tragedy of puddnhead wilson	0	0	0	1
1044	extract from captain stormfields visit to Heaven	twain 1044: extract from captain stormfields visit to Heaven	2	3	3	2
1086	a horses tale	twain 1086: a horses tale	0	0	0	1
119	a tramp abroad	twain 119: a tramp abroad	0	0	0	0
142	the 30000 bequest and other stories	twain 142: the 30000 bequest and other stories	0	0	0	0
1837	the prince and the pauper	twain 1837: the prince and the pauper	0	0	0	1
19484	editorial wild oats	twain 19484: editorial wild oats	0	1	0	0
19987	chapters from my autobiography	twain 19987: chapters from my autobiography	0	0	0	0
245	life on the mississippi	twain 245: life on the mississippi	0	0	0	0
2874	personal recollections of joan of arc vol 1	twain 2874: personal recollections of joan of arc vol 1	0	0	0	1
2875	personal recollections of joan of arc vol 2	twain 2875: personal recollections of joan of arc vol 2	0	0	0	1
2895	following the equator	twain 2895: following the equator	0	0	0	0
3171	in defense of harriet shelley	twain 3171: in defense of harriet shelley	0	1	0	0
3172	fenimore coopers literary offences	twain 3172: fenimore coopers literary offences	0	2	0	0
3173	essays on paul bourget	twain 3173: essays on paul bourget	0	1	0	0
3176	the innocents abroad	twain 3176: the innocents abroad	0	0	0	0
3177	roughing it	twain 3177: roughing it	0	0	0	0

book_id	title	label	y_raw	y_L0	y_L1	y_L2
3178	the gilded age	twain 3178: the gilded age	0	0	0	0
3179	the american claimant	twain 3179: the american claimant	0	0	0	1
3180	a double barrelled detective story	twain 3180: a double barrelled detective story	0	0	0	1
3181	the stolen white elephant	twain 3181: the stolen white elephant	0	1	0	0
3182	some rambling notes of an idle excursion	twain 3182: some rambling notes of an idle excursion	0	0	0	0
3183	the facts concerning the recent carnival of crime in connecticut	twain 3183: the facts concerning the recent carnival of crime in connecticut	3	1	0	1
3184	alonzo fitz and other stories	twain 3184: alonzo fitz and other stories	0	0	0	0
3185	those extraordinary twins	twain 3185: those extraordinary twins	0	0	0	1
3186	the mysterious stranger and other stories	twain 3186: the mysterious stranger and other stories	0	0	0	1
3188	mark twain speeches	twain 3188: mark twain speeches	0	0	0	0
3189	sketches new and old	twain 3189: sketches new and old	0	0	0	0
3190	1601 conversation as it was by the social fireside in the time of the tudors	twain 3190: 1601 conversation as it was by the social fireside in the time of the tudors	0	1	0	0
3191	goldsmiths friend abroad again	twain 3191: goldsmiths friend abroad again	0	1	0	0
3192	the curious republic of gondour and other whimsical sketches	twain 3192: the curious republic of gondour and other whimsical sketches	0	0	0	0
3199	the letters of mark twain	twain 3199: the letters of mark twain	0	0	0	0
3250	how to tell a story and other essays	twain 3250: how to tell a story and other essays	0	3	0	0
3251	the man that corrupted hadleyburg and other stories	twain 3251: the man that corrupted hadleyburg and other stories	0	0	0	0
33077	the treaty with china its provisions explained	twain 33077: the treaty with china its provisions explained	0	2	1	3
60900	merry tales	twain 60900: merry tales	0	0	0	1
61522	the 1000000 bank note	twain 61522: the 1000000 bank note	0	0	0	0
62636	to the person sitting in darkness	twain 62636: to the person sitting in darkness	1	2	2	0
62739	king leopolds soliloquy	twain 62739: king leopolds soliloquy	0	1	0	0

book_id	title	label	y_raw	y_L0	y_L1	y_L2
70	what is man	twain 70: what is man	0	0	0	0
74	the adventures of tom sawyer	twain 74: the adventures of tom sawyer	0	0	0	1
76	the adventures of huckleberry finn	twain 76: the adventures of huckleberry finn	2	0	3	2
86	a connecticut yankee in king arthurs court	twain 86: a connecticut yankee in king arthurs court	0	0	0	1
91	tom sawyer abroad	twain 91: tom sawyer abroad	2	3	3	2
93	tom sawyer detective	twain 93: tom sawyer detective	2	3	3	2

In [84]:

```
# k values to test
k_vals = list(range(2, 11))

# different feature vectors to use
feature_vectors = {'raw': mean_TFIDF_sigs,
                   'L0': L0,
                   'L1': L1,
                   'L2': L2}

# empty dataframe
km_results = pd.DataFrame(columns = ['k', 'raw_silhouette_score', 'L0_silhouette

# loop through k values (num of clusters) and compute silhouette score to find b
for k in k_vals:
    km = KMeans(k, random_state = 314)

    results = [k]

    for vec in feature_vectors.values():
        labels = km.fit_predict(vec)

        results.append(silhouette_score(vec, labels))

    km_results.loc[len(km_results)] = results
```

In [85]:

```
km_results.style.background_gradient(cmap = 'RdBu', axis = None, subset = km_res
```

Out[85]:

	k	raw_silhouette_score	L0_silhouette_score	L1_silhouette_score	L2_silhouette_score
0	2.000000	0.255373	0.377931	0.236509	0.091829
1	3.000000	0.259579	0.342163	0.242996	0.064479
2	4.000000	0.248763	0.337037	0.240644	0.054186
3	5.000000	0.138738	0.336095	0.247383	0.054036
4	6.000000	0.245425	0.336220	0.057003	0.042060
5	7.000000	0.142032	0.327749	0.219367	0.047595
6	8.000000	0.136787	0.335951	0.047044	0.057804

	k	raw_silhouette_score	L0_silhouette_score	L1_silhouette_score	L2_silhouette_score
7	9.000000	0.072876	0.330097	0.045014	0.042156
8	10.000000	0.049228	0.311804	0.087053	0.044631

In [86]:
```
km_results.iloc[:, 1:].idxmax()
```

Out[86]:
```
raw_silhouette_score    1
L0_silhouette_score     0
L1_silhouette_score     3
L2_silhouette_score     0
dtype: int64
```

In [87]:
```
# overall highest silhouette score
max_silhouette_score = km_results.iloc[:,1:].max().max()

# k value (num of clusters) corresponding to the highest silhouette score
max_score_cluster = km_results.loc[km_results[km_results == max_silhouette_score

# feature vector corresponding to the highest silhouette score
max_score_vec = km_results.loc[km_results[km_results == max_silhouette_score].an
```

In [88]:
```
# create a col with labels corresponding to k value, feature vector that yield h
km = KMeans(int(max_score_cluster), random_state = 314)

max_col_name = 'max_y_{}'.format(max_score_vec.split('_')[0])

book_DOC[max_col_name] = km.fit_predict(feature_vectors[max_score_vec.split('_')
```

In [89]:
```
# add to see cluster breakdown by type
book_DOC = book_DOC.join(LIB['type'])
```

In [90]:
```
book_DOC[['label', 'type', max_col_name]].sort_values(max_col_name).style.backgr
```

Out[90]:

book_id	label	type	max_y_L0
62739	twain 62739: king leopolds soliloquy	stories	0
62636	twain 62636: to the person sitting in darkness	non-fiction	0
3173	twain 3173: essays on paul bourget	non-fiction	0
3172	twain 3172: fenimore coopers literary offences	non-fiction	0
3171	twain 3171: in defense of harriet shelley	non-fiction	0
3190	twain 3190: 1601 conversation as it was by the social fireside in the time of the tudors	stories	0

book_id	label	type	max_y_L0
3183	twain 3183: the facts concerning the recent carnival of crime in connecticut	stories	0
1044	twain 1044: extract from captain stormfields visit to Heaven	stories	0
3191	twain 3191: goldsmiths friend abroad again	stories	0
19484	twain 19484: editorial wild oats	stories	0
93	twain 93: tom sawyer detective	novel	0
33077	twain 33077: the treaty with china its provisions explained	non-fiction	0
3250	twain 3250: how to tell a story and other essays	non-fiction	0
3181	twain 3181: the stolen white elephant	stories	0
3199	twain 3199: the letters of mark twain	non-fiction	1
3251	twain 3251: the man that corrupted hadleyburg and other stories	stories	1
3192	twain 3192: the curious republic of gondour and other whimsical sketches	stories	1
3189	twain 3189: sketches new and old	stories	1
3188	twain 3188: mark twain speeches	non-fiction	1
19987	twain 19987: chapters from my autobiography	non-fiction	1
60900	twain 60900: merry tales	stories	1
3186	twain 3186: the mysterious stranger and other stories	stories	1
3185	twain 3185: those extraordinary twins	stories	1
3184	twain 3184: alonzo fitz and other stories	stories	1
61522	twain 61522: the 1000000 bank note	stories	1
70	twain 70: what is man	non-fiction	1
3180	twain 3180: a double barrelled detective story	stories	1
74	twain 74: the adventures of tom sawyer	novel	1
76	twain 76: the adventures of huckleberry finn	novel	1
86	twain 86: a connecticut yankee in king arthurs court	novel	1
91	twain 91: tom sawyer abroad	novel	1
102	twain 102: the tragedy of puddnhead wilson	novel	1
119	twain 119: a tramp abroad	non-fiction	1
142	twain 142: the 30000 bequest and other stories	stories	1

book_id	label	type	max_y_L0
3182	twain 3182: some rambling notes of an idle excursion	non-fiction	1
245	twain 245: life on the mississippi	non-fiction	1
1837	twain 1837: the prince and the pauper	novel	1
2874	twain 2874: personal recollections of joan of arc vol 1	non-fiction	1
2875	twain 2875: personal recollections of joan of arc vol 2	non-fiction	1
2895	twain 2895: following the equator	non-fiction	1
3176	twain 3176: the innocents abroad	non-fiction	1
3177	twain 3177: roughing it	novel	1
3178	twain 3178: the gilded age	novel	1
1086	twain 1086: a horses tale	novel	1
3179	twain 3179: the american claimant	novel	1

In [91]:
```
book_DOC.groupby(max_col_name).size()
```

Out[91]:
```
max_y_L0
0    14
1    31
dtype: int64
```

In [92]:
```
# cluster breakdown by type
book_DOC.groupby(['type', max_col_name]).size()
```

Out[92]:
```
type         max_y_L0
non-fiction  0            6
             1           11
novel        0            1
             1           10
stories      0            7
             1           10
dtype: int64
```

M07: Features and Components

In [93]:
```
TFIDF_sigs
```

Out[93]:

book_id	chap_id	term_str	saying	seem	indeed	couldnt	door	taken	deal
70	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000

	term_str	saying	seem	indeed	couldnt	door	taken	deal	
book_id	chap_id								
	2	0.005272	0.005285	0.005246	0.010622	0.001062	0.004187	0.002093	0.00
	3	0.006831	0.000000	0.000000	0.006881	0.055051	0.000000	0.000000	0.00
	4	0.005859	0.000000	0.005830	0.005902	0.000000	0.000000	0.011632	0.01
	5	0.008429	0.012674	0.004194	0.012737	0.004246	0.004183	0.004183	0.01
...	...	...	...	...	...	...	...	...	
62739	2	0.016015	0.008027	0.003984	0.004033	0.000000	0.003974	0.003974	0.00
	3	0.045041	0.045152	0.000000	0.000000	0.000000	0.000000	0.000000	0.04
	4	0.005093	0.005106	0.005068	0.010261	0.005131	0.000000	0.000000	0.00
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.013626	0.013626	0.00
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.043356	0.000000	0.00

1108 rows × 1000 columns

In [94]:
```
chap_DOC = pd.DataFrame(index = TFIDF.index)

chap_DOC = chap_DOC.join(LIB[['author', 'title', 'type', 'decade']], on = 'book_

chap_DOC['label'] = chap_DOC.apply(lambda x: "{}-{}-{}".format(x.name[0], x.auth

chap_DOC['mean_tfidf'] = TFIDF.mean(1)

chap_DOC['n_tokens'] = BOW.groupby(OHCO[:2]).n.sum()
```

In [95]:
```
chap_DOC
```

Out[95]:

		author	title	type	decade	label	mean_tfidf	n_tokens
book_id	chap_id							
70	1	twain	what is man	non-fiction	1900	70-twain-1	0.000506	71
	2	twain	what is man	non-fiction	1900	70-twain-2	0.000605	26698
	3	twain	what is man	non-fiction	1900	70-twain-3	0.000507	4367
	4	twain	what is man	non-fiction	1900	70-twain-4	0.000380	3669
	5	twain	what is man	non-fiction	1900	70-twain-5	0.000419	5465
...	...	...	...	...	...	...	...	...
62739	2	twain	king leopolds soliloquy	stories	1900	62739-twain-2	0.000536	6390

book_id	chap_id	author	title	type	decade	label	mean_tfidf	n_tokens
	3	twain	king leopolds soliloquy	stories	1900	62739-twain-3	0.000782	686
	4	twain	king leopolds soliloquy	stories	1900	62739-twain-4	0.000420	3033
	5	twain	king leopolds soliloquy	stories	1900	62739-twain-5	0.000354	1129
	6	twain	king leopolds soliloquy	stories	1900	62739-twain-6	0.000428	356

1108 rows × 7 columns

In [96]:
```
TFIDF_sigs
```

Out[96]:

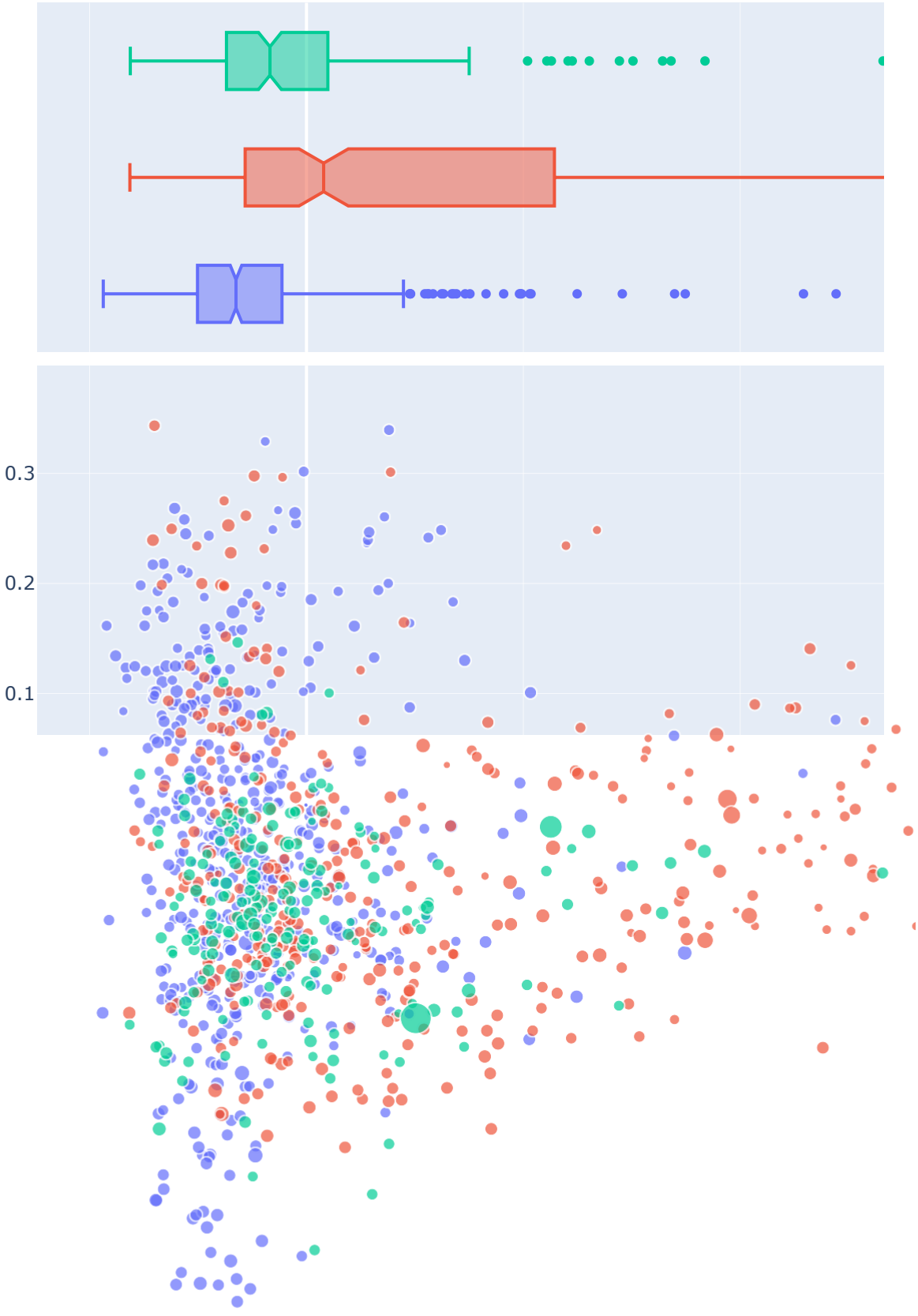
book_id	chap_id	term_str	saying	seem	indeed	couldnt	door	taken	deal	
70	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	2	0.005272	0.005285	0.005246	0.010622	0.001062	0.004187	0.002093	0.00
	3	0.006831	0.000000	0.000000	0.006881	0.055051	0.000000	0.000000	0.00
	4	0.005859	0.000000	0.005830	0.005902	0.000000	0.000000	0.011632	0.01
	5	0.008429	0.012674	0.004194	0.012737	0.004246	0.004183	0.004183	0.01
...	...	...	...	...	...	...	...	...	...	
62739	2	0.016015	0.008027	0.003984	0.004033	0.000000	0.003974	0.003974	0.000
	3	0.045041	0.045152	0.000000	0.000000	0.000000	0.000000	0.000000	0.04
	4	0.005093	0.005106	0.005068	0.010261	0.005131	0.000000	0.000000	0.005
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.013626	0.013626	0.000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.043356	0.000000	0.000

1108 rows × 1000 columns

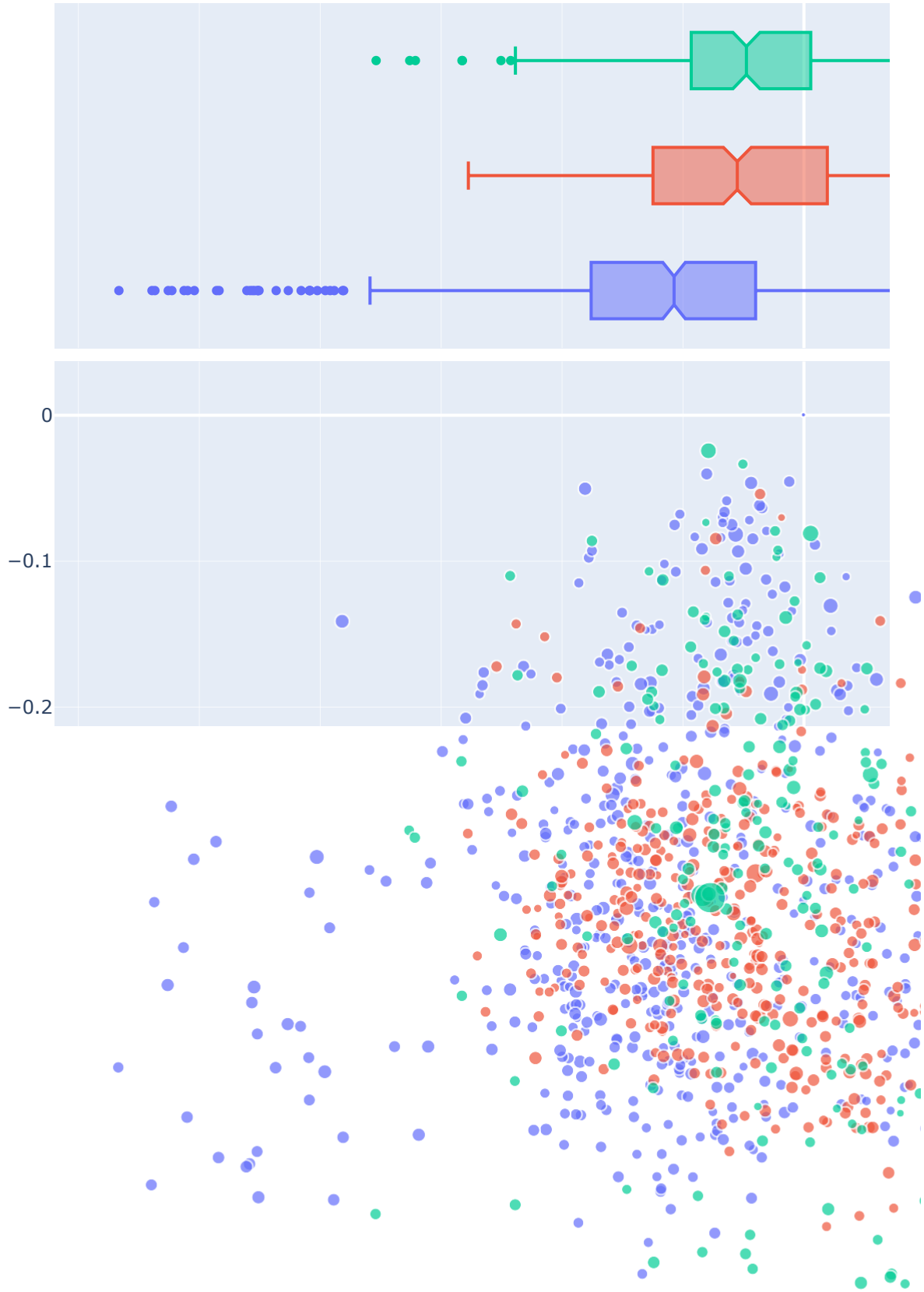
Manual PCA Methods with Only 1000 Most Significant Terms (excluding proper nouns)

In [97]:
```
LOADINGS, DCM, COMPINF = get_pca(TFIDF_sigs, norm_docs = True, center_by_mean =
```

In [98]:
```
px.scatter(DCM, 0, 1, color=chap_DOC.type,
           size=np.abs(chap_DOC.mean_tfidf), hover_name=chap_DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```

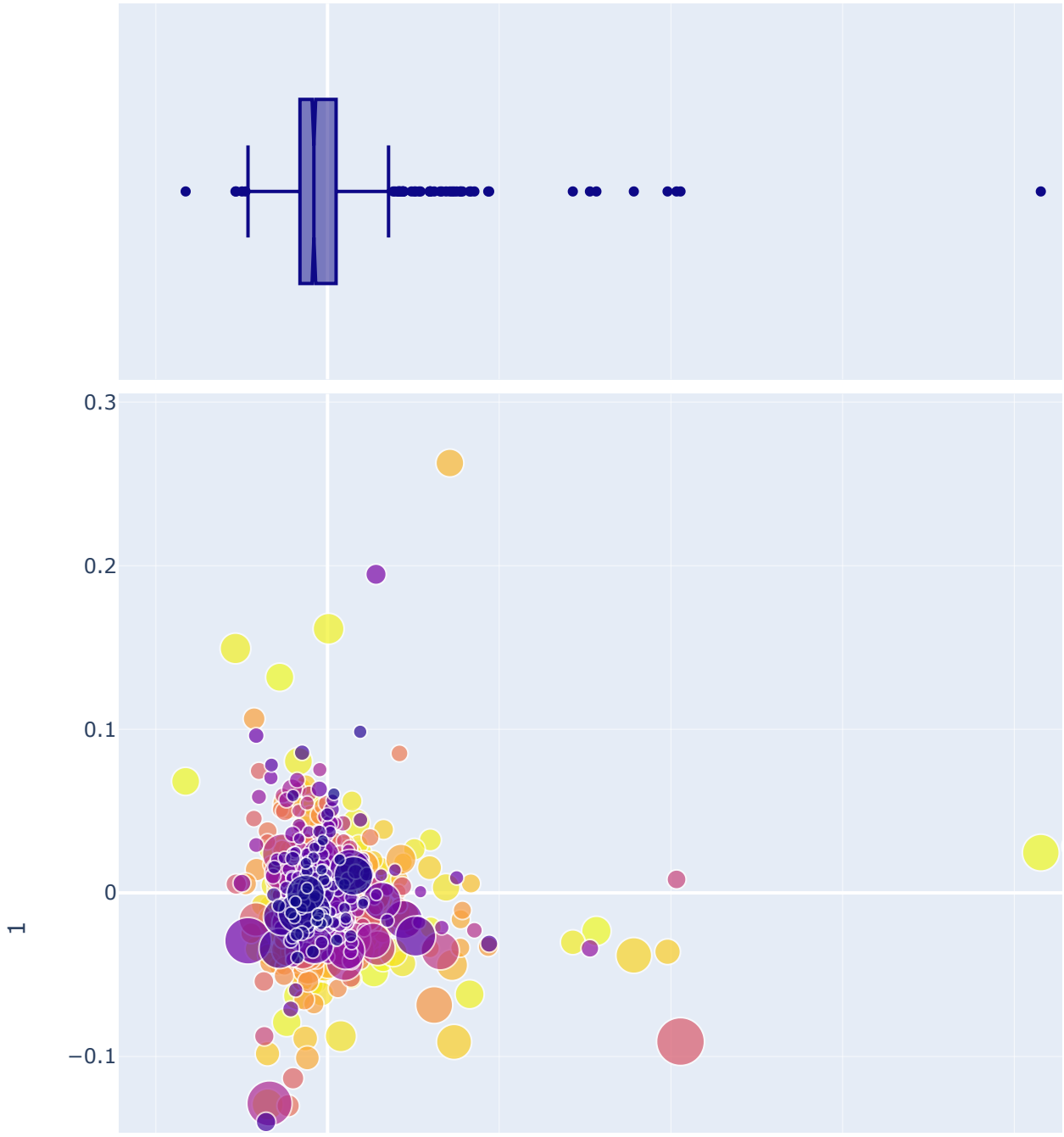

0.3
0.2
0.1

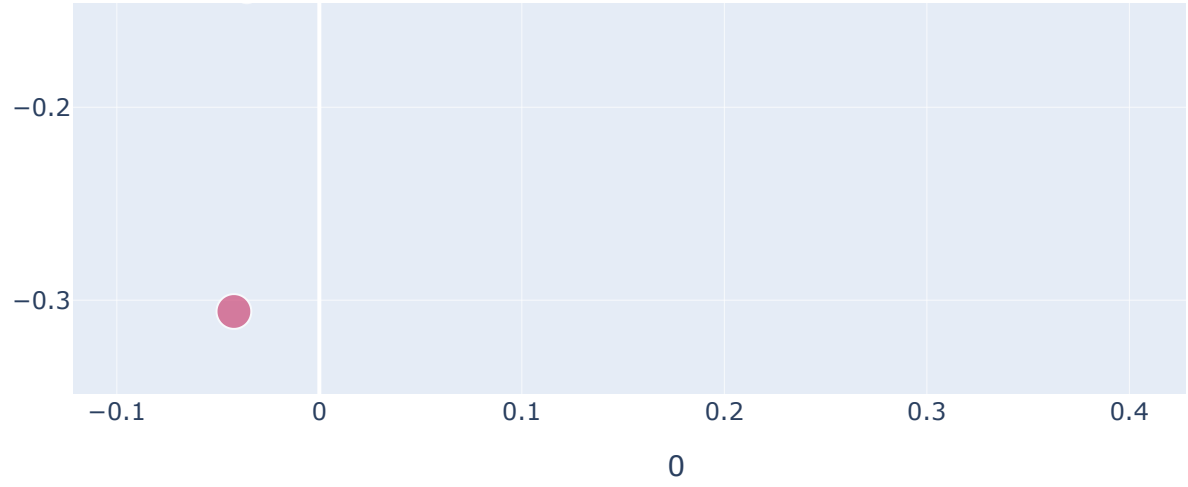
In [99]:
```
px.scatter(DCM, 2, 3, color=chap_DOC.type,
           size=np.abs(chap_DOC.mean_tfidf), hover_name=chap_DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```



In [100…
```
X = LOADINGS.join(SIGS, how='inner').reset_index()
```

In [101…
```
px.scatter(X, 0, 1, size=X.n, color=X.dfidf,
           hover_name='term_str', hover_data=['max_pos'],
           marginal_x='box', marginal_y='box',
           height=1000, width=1000)
```





In [102…
```
COMPINF
```

Out[102…

pc_id	pos	neg	eig_val	exp_var
0	aint says dont reckon thats	city feet government french war	0.025783	0.224980
1	river boat water feet miles	letter letters wrote book am	0.016925	0.147683
2	sir court face voice father	letter letters book river wrote	0.013273	0.115819
3	bill american sir honor government	letter feet oh mother boys	0.012755	0.111299
4	says city aint church sea	river boat sir letter shore	0.009164	0.079961
5	sir government dollars bill money	letter letters book boys mother	0.008131	0.070952
6	sir letter ship sea letters	river french government boat war	0.007849	0.068485
7	dollars money boys silver boy	says sir french ship sea	0.007382	0.064418
8	ship boat sea dont oh	says dollars letter silver gold	0.006870	0.059950
9	ship horse french yes army	says father sir woman river	0.006470	0.056452

Prince PCA Method with entire TFIDF

In [103…
```
pca = PCA(
    n_components=6,
    n_iter=3,
    rescale_with_mean=False, # Already set and applied to TFIDF
    rescale_with_std=False, # Already set and applied to TFIDF
    copy=True,
    check_input=True,
    engine='auto',
    random_state=42
)
```

In [104…
```
pca = pca.fit(TFIDF)
```

In [105…
```
dcm = pca.transform(TFIDF)
```

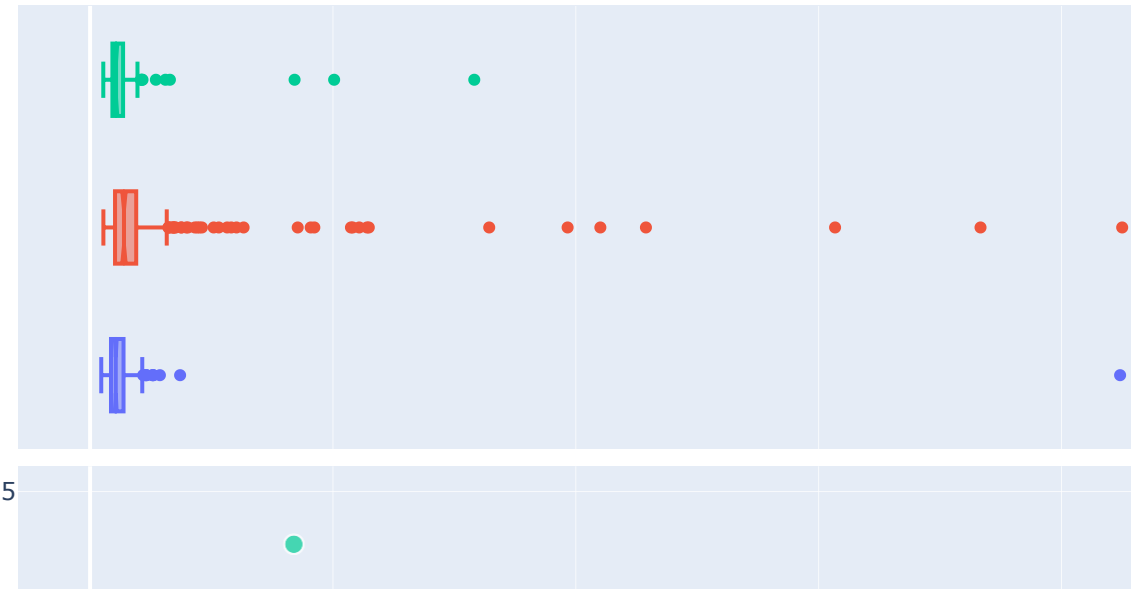
In [106…
```
dcm
```

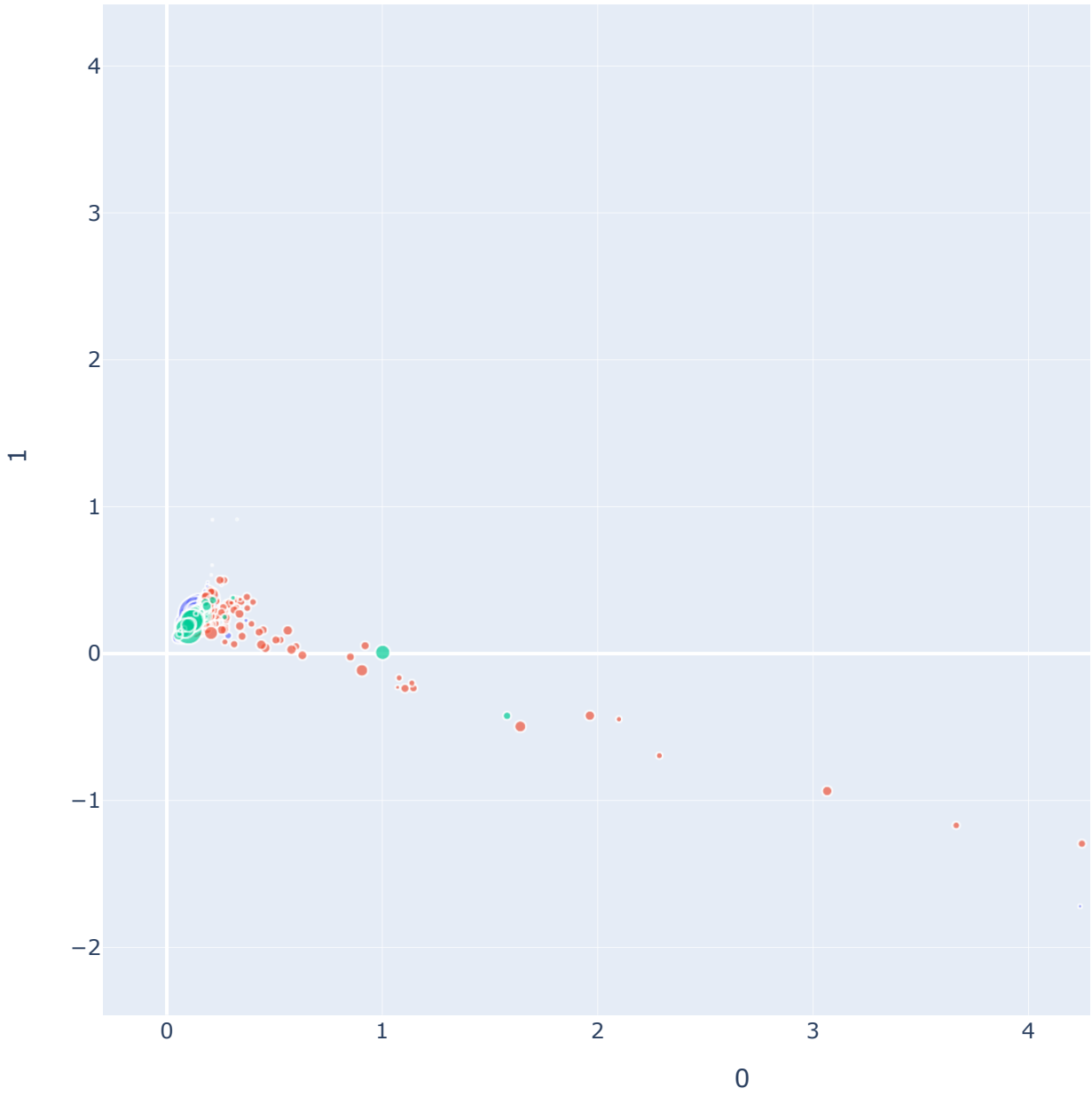
Out[106…

		0	1	2	3	4	5
book_id	**chap_id**						
70	**1**	0.113931	0.302671	-0.170870	0.040518	-0.037079	-0.075492
	2	0.120829	0.241899	-0.146759	-0.027912	-0.016165	0.163990
	3	0.162728	0.294099	-0.166797	-0.021350	-0.015659	-0.026615
	4	0.075285	0.150127	-0.085885	-0.019615	-0.021718	0.001746
	5	0.091257	0.178199	-0.095310	-0.023824	-0.011637	-0.013031
…	**…**	…	…	…	…	…	…
62739	**2**	0.102761	0.195586	-0.099366	-0.025425	-0.013681	-0.010889
	3	0.138763	0.272401	-0.149855	-0.030387	-0.053265	-0.013249
	4	0.055947	0.124940	-0.060920	-0.019205	-0.005833	-0.005138
	5	0.061812	0.135055	-0.077031	-0.020244	-0.007853	-0.007598
	6	0.065727	0.158000	-0.077146	-0.014555	-0.012263	0.021367

1108 rows × 6 columns

In [107…
```
px.scatter(dcm, 0, 1,
           color=chap_DOC.type,
           size=chap_DOC.n_tokens, hover_name=chap_DOC.label,
           height=1000, width=1200,
           marginal_x='box', marginal_y='box')
```





THERE ARE OUTLIERS IN THE ABOVE PLOT THAT ARE SKEWING THE RESULTS

It seems as though the first principal component (PC 0) is for vernacular and the second one (PC 1) is for German

The Adventures of Huckleberry Finn (76)

- Chapter 4: lots of vernacular and misspellings
- Chapter 8: lots of vernacular and misspellings
- Chapter 15: lots of vernacular and misspellings
- Chapter 23: lots of vernacular and misspellings
- Chapter 38: lots of vernacular and misspellings
- Chapter 43: lots of vernacular and misspellings

Mark Twain Speeches (3188)

- Chapter 10: "A New German Word" → contains a "German" word

The Tragedy of Pudd'nhead Wilson (102)

Note that the chapter numbers listed below are one chapter greater than those in the book (e.g., Chapter 4 below is actually Chapter 14) but there is an intro

- Chapter 4 (Roxy Plays a Shrewd Trick): lots of vernacular and misspellings
- Chapter 9 (Marse Tom Tramples His Chance): lots of vernacular and misspellings
- Chapter 10 (Tom Practises Syncophancy): lots of vernacular and misspellings
- Chapter 15 (Roxana Insists Upeon Reform): lots of vernacular and misspellings
- Chapter 19 (Roxana Commands): lots of vernacular and misspellings

How to Tell A Story and Other Essays (3250)

- Chapter 3: "The Golden Arm" → lots of vernacular and misspellings

Merry Tales (60900)

- Ch. 6: "Meisterschaft" play → speaking German

Sketches New and Old (3189)

- Chapter 3: "The Jumping Frog" → in both English and French
- Chapter 7: A Couple of Poems by (Thomas) Moore and Twain ("Those Evening Bills" by Moore and "Those Annual Bills" by Twain) → short, different style (poems vs. stories and authors)
- Chapter 31: "A True Story" → lots of Southern vernacular (and improper spellings)

Prince PCA Method with Outliers from above Removed

In [108…

```
# function to calculate the upper fence / bound in the box plots above for the d

def upper_fence(df, books, pc):
    pc_IQR = df.loc[books, pc].quantile(0.75) - df.loc[books, pc].quantile(0.25)
    return 1.5 * pc_IQR + df.loc[books, pc].quantile(0.75)
```

In [109…

```
# upper fences for pc 0
twain_0_upper_fence = upper_fence(dcm, LIB.index.values, 0)

# upper fences for pc 1
twain_1_upper_fence = upper_fence(dcm, LIB.index.values, 1)
```

In [110…

```
# outliers the chapters in books with PC 0 or PC 1 greater than the max of the u
outliers = dcm.loc[(dcm[0] > twain_0_upper_fence) | (dcm[1] > twain_1_upper_fenc
```

In [111…

```
# known Twain outliers from experimentation (see twain_analysis_M7.ipynb)
twain_outliers = [(70, 2), (76, 4), (76, 8), (76, 15), (76, 23), (76, 38), (76,
```

```
(102, 15), (102, 19), (142, 16), (3180, 10), (3250, 3), (60900
```

In [112…
```
# remove outliers from corpus
small_CORPUS = CORPUS.loc[~CORPUS.index.droplevel(['para_num', 'sent_num', 'toke
                                  .isin(outliers)]

# remove known Twain outliers from corpus
small_CORPUS = small_CORPUS.loc[~small_CORPUS.index.droplevel(['para_num', 'sent
                                           .isin(twain_outliers)]
```

In [113…
```
# remove outliers from vocab
small_VOCAB = VOCAB.loc[VOCAB.index.isin(small_CORPUS.term_str)]

# remove proper nounrs

proper_nouns = ['NNP', 'NNPS']

small_VOCAB = VOCAB.loc[~VOCAB.max_pos.isin(proper_nouns)]
```

In [114…
```
# remove ~18% of VOCAB data

(VOCAB.shape[0] - small_VOCAB.shape[0]) / VOCAB.shape[0]
```

Out[114…
```
0.17647714190217997
```

In [115…
```
# remove proper nouns from corpus
small_CORPUS = small_CORPUS.loc[small_CORPUS.term_str.isin(small_VOCAB.index.val
```

In [116…
```
# remove ~11% of data

(CORPUS.shape[0] - small_CORPUS.shape[0]) / CORPUS.shape[0]
```

Out[116…
```
0.10910914381979803
```

In [117…
```
small_BOW = create_bow(small_CORPUS, CHAPS)
```

In [118…
```
# suppress chained assignment warning
pd.options.mode.chained_assignment = None
```

In [119…
```
small_DTCM, small_TFIDF, small_BOW, small_DFIDF, small_VOCAB = get_tfidf(small_B
```

In [120…
```
small_chap_DOC = pd.DataFrame(index = small_TFIDF.index)

small_chap_DOC = small_chap_DOC.join(LIB[['author', 'title', 'type', 'decade']],

small_chap_DOC['label'] = small_chap_DOC.apply(lambda x: "{}-{}-{}".format(x.nam

small_chap_DOC['mean_tfidf'] = TFIDF.mean(1)
```

```
small_chap_DOC['n_tokens'] = small_BOW.groupby(OHCO[:2]).n.sum()
```

In [121…

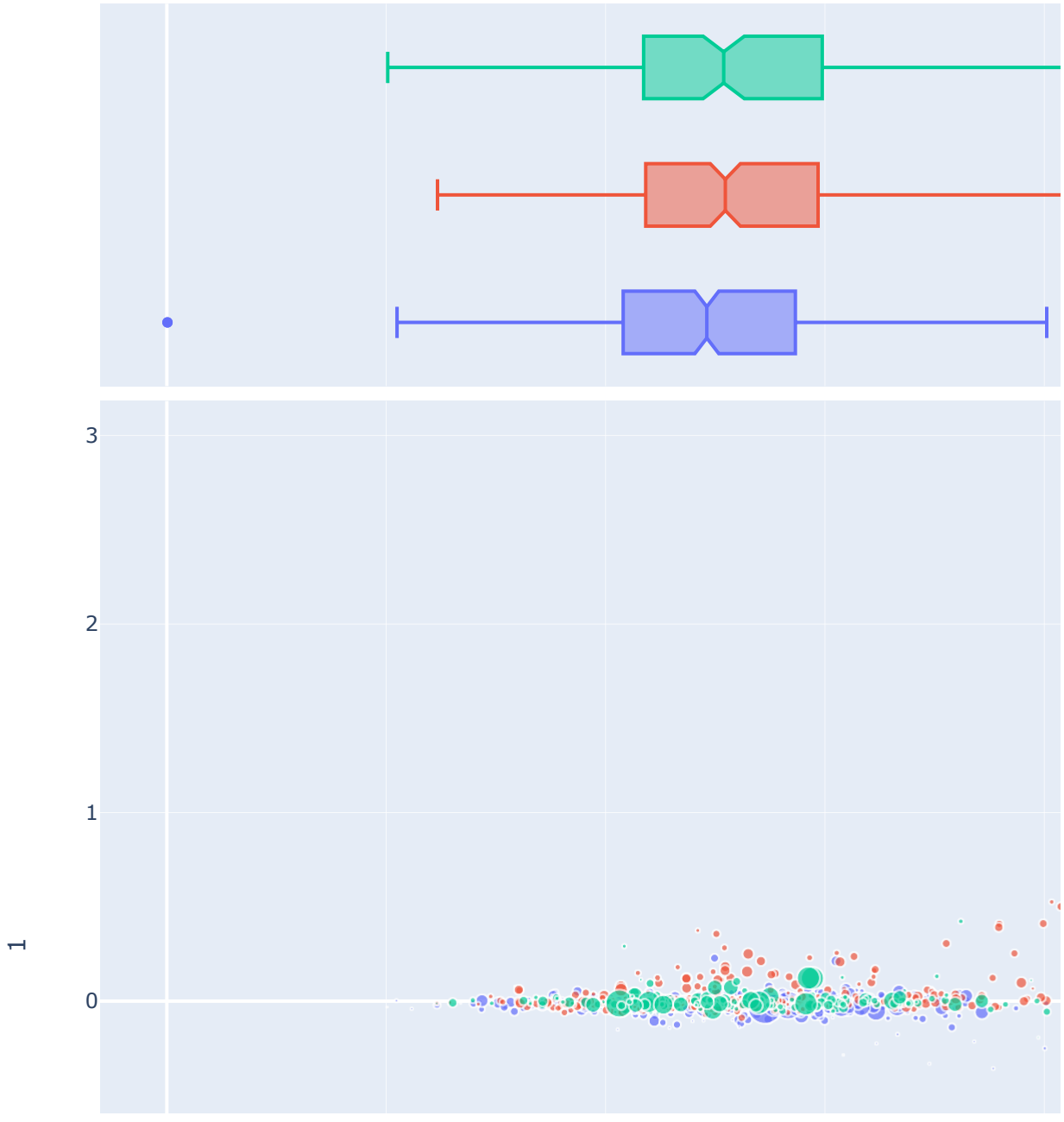
```
small_pca = pca.fit(small_TFIDF)
```

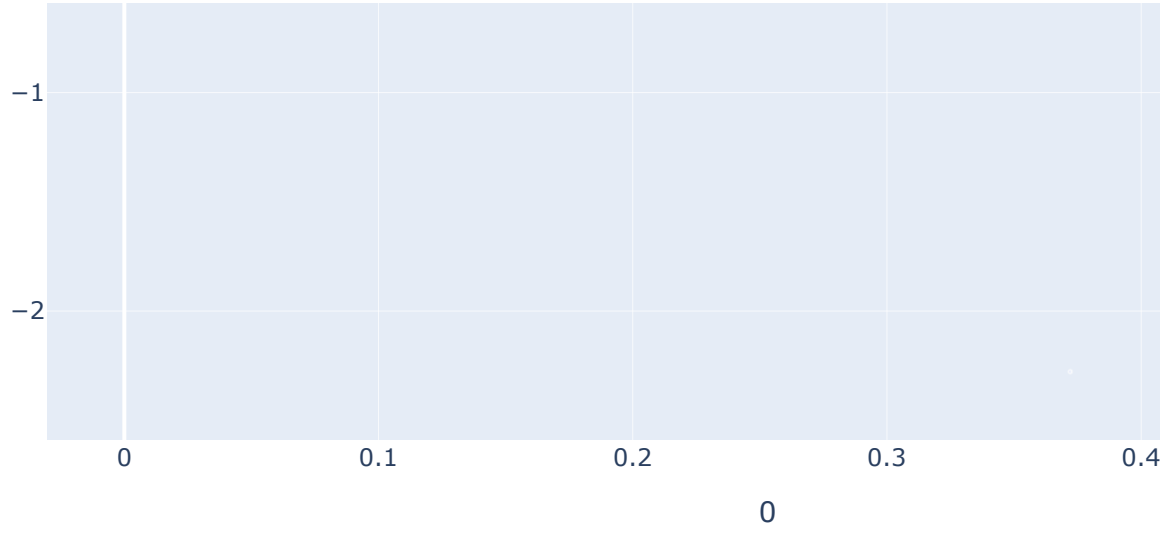
In [122…

```
small_dcm = pca.transform(small_TFIDF)
```

In [123…

```
px.scatter(small_dcm, 0, 1,
           color=small_chap_DOC.type,
           size=small_chap_DOC.n_tokens, hover_name=small_chap_DOC.label,
           height=1000, width=1200,
           marginal_x='box', marginal_y='box')
```





Prince PCA Method with Only 1000 Most Significant Words (`TFIDF_sigs`)

In [124…
```
pca_sigs = pca.fit(TFIDF_sigs)
```

In [125…
```
dcm_sigs = pca_sigs.transform(TFIDF_sigs)
```

In [126…
```
dcm_sigs
```

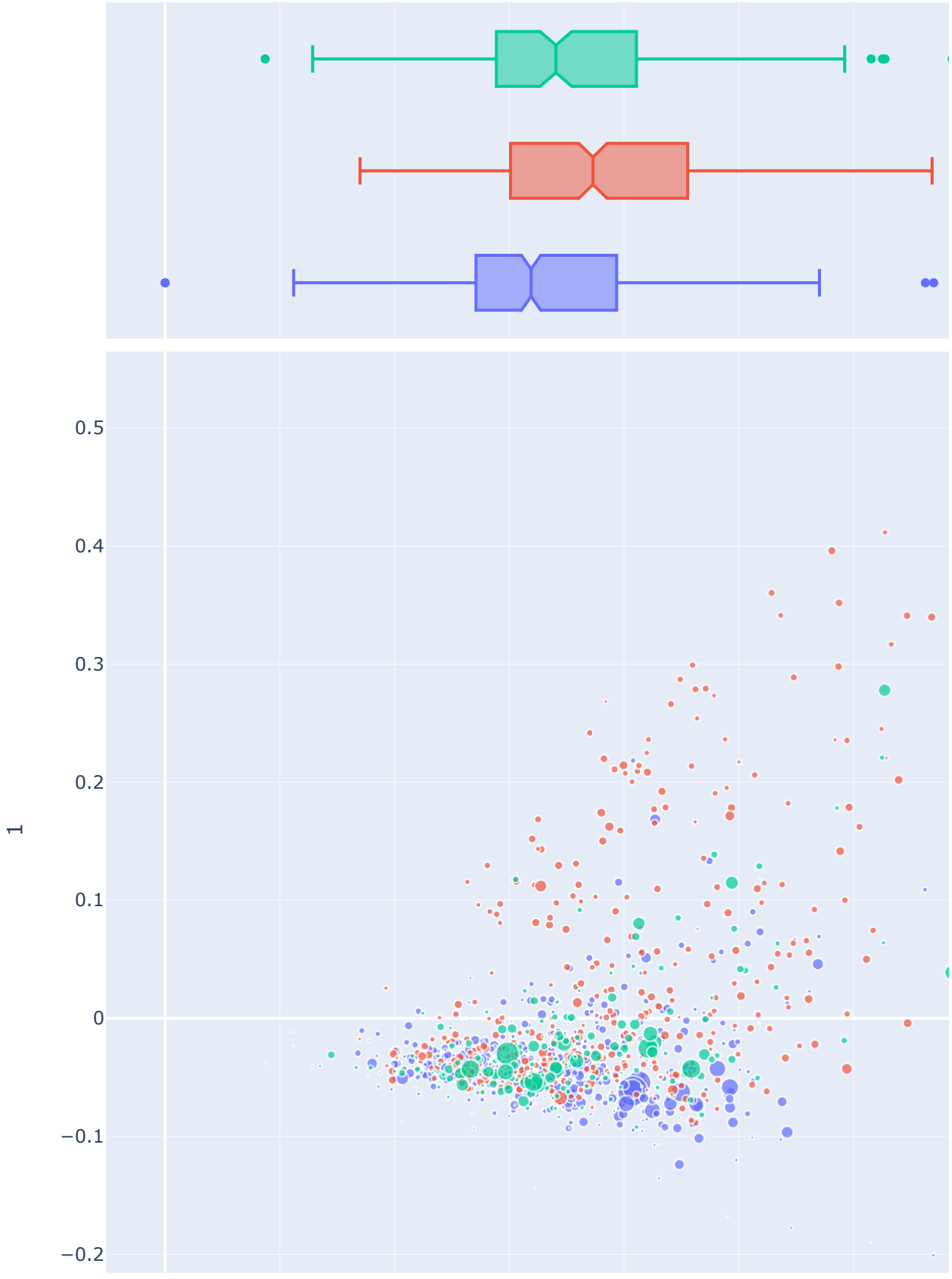
Out[126…

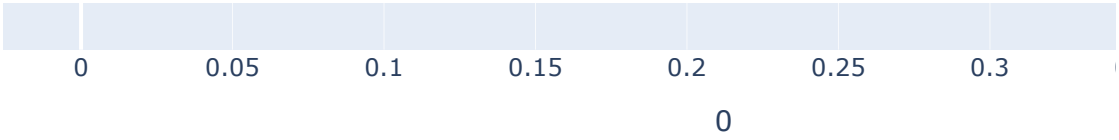
		0	1	2	3	4	5
book_id	**chap_id**						
70	**1**	0.056153	-0.023210	-0.013009	0.004759	-0.005821	0.004595
	2	0.180922	-0.038227	0.022268	-0.039044	-0.009613	0.004492
	3	0.220261	-0.079028	0.039250	0.012342	0.013575	-0.047303
	4	0.133059	-0.036709	-0.007816	0.012793	0.007986	0.016466
	5	0.154748	-0.040718	-0.017498	0.031480	0.008514	-0.012437
...	**...**	...	...	...	...	...	...
62739	**2**	0.168050	-0.050184	-0.021159	0.007675	-0.006063	0.012953
	3	0.159232	-0.048936	0.003923	-0.028582	0.020467	0.037295
	4	0.072618	-0.030855	0.001835	0.009470	0.000830	0.018404
	5	0.089742	-0.041856	0.026121	-0.001859	-0.024549	0.000448
	6	0.064286	-0.040945	0.010348	0.004402	-0.011561	0.000698

1108 rows × 6 columns

In [127…
```
px.scatter(dcm_sigs, 0, 1,
```

```
        color=chap_DOC.type,
        size=chap_DOC.n_tokens, hover_name=chap_DOC.label,
        height=1000, width=1200,
        marginal_x='box', marginal_y='box')
```





In [128…

```
# save BOW for topic modeling

BOW.to_csv(f'twain_BOW.csv')
```

Sources

- `pandas.DataFrame.droplevel` to drop one or more levels of a `MultiIndex` : https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.droplevel.html
- How to retrieve specific combinations of `MultiIndex` levels: https://stackoverflow.com/questions/52798386/pandas-dataframe-how-to-retrieve-specific-combinations-of-multiindex-levels
- Accessing data in a `MultiIndex` : https://towardsdatascience.com/accessing-data-in-a-multiindex-dataframe-in-pandas-569e8767201d
- `sklearn.metrics.silhouette_score` : https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
- Append a row as a list to a dataframe: https://sparkbyexamples.com/pandas/pandas-append-list-as-a-row-to-dataframe/
- pandas `background_gradient` : https://pandas.pydata.org/docs/reference/api/pandas.io.formats.style.Styler.background_gradie
- Deal with SettingWithCopyWarning in pandas: https://stackoverflow.com/questions/20625582/how-to-deal-with-settingwithcopywarning-in-pandas

In [ ]: