Analysis for Dickens 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("dickens_pre_LIB.csv").set_index(BOOKS).sort_index()
```

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

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

In [7]:
```
VOCAB = pd.read_csv("dickens_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_porter	st
0	60	1	1.207251e-05	16.337915	CD	4	{'RB', 'CD', 'NN', 'JJ'}	0	0	
1	38	1	7.645923e-06	16.996878	CD	5	{'NNP', 'CD', 'VB', 'NN', 'JJ'}	0	1	
10	8	2	1.609668e-06	19.244805	CD	4	{'NNP', 'IN', 'CD', 'NN'}	0	10	
100	4	3	8.048340e-07	20.244805	CD	4	{'JJ', 'IN', 'CD', 'NN'}	0	100	
1000	1	4	2.012085e-07	22.244805	JJ	1	{'JJ'}	0	1000	
...	...	...	...	...	...	...	...	...	...	
æolian	2	6	4.024170e-07	21.244805	JJ	1	{'JJ'}	0	æolian	
æsop	1	4	2.012085e-07	22.244805	NN	1	{'NN'}	0	æsop	
éclat	1	5	2.012085e-07	22.244805	NN	1	{'NN'}	0	éclat	
élite	1	5	2.012085e-07	22.244805	NN	1	{'NN'}	0	élite	

term_str	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	stem_porter	st
ěngine	1	6	2.012085e-07	22.244805	NNP	1	{'NNP'}	0	ěngine	

55272 rows × 11 columns

In [9]:
```
LIB
```

Out[9]:

book_id	source_file_path	title	chap_r
98	Dickens/98-a_tale_of_two_cities.txt	a tale of two cities	^\s*CHAPTER\s*[IVXLCM
564	Dickens/564-the_mystery_of_edwin_drood.txt	the mystery of edwin drood	^CHAPTER\s[IVXLCM
580	Dickens/580-the_pickwick_papers.txt	the pickwick papers	^CHAPTER\s[IVXLCM]+\.\s[A
588	Dickens/588-master_humphreys_clock.txt	master humphreys clock	^(?:[IVXLCM]+$|TO THE READER
644	Dickens/644-the_haunted_man_and_the_ghosts_bar...	the haunted man and the ghosts bargain	^CHAPTER\s[IVXLC
650	Dickens/650-pictures_from_italy.txt	pictures from italy	THE READER'S PASSPORT|GOING THRO FRANCE|I
653	Dickens/653-the_chimes.txt	the chimes	^CHAPTER\s[IVXL
675	Dickens/675-american_notes.txt	american notes	^CHAPTER\s[IVXLC
676	Dickens/676-the_battle_of_life.txt	the battle of life	^Part the [A-Z][a-
699	Dickens/699-a_childs_history_of_england.txt	a childs history of england	^CHAPTER\s[IVXLC
700	Dickens/700-the_old_curiosity_shop.txt	the old curiosity shop	^CHAP
730	Dickens/730-oliver_twist.txt	oliver twist	^\s*CHAPTER\s*[IVXLCM
766	Dickens/766-david_copperfield.txt	david copperfield	\s*(PREFACE\sTO|CHAPTER\s*[C
786	Dickens/786-hard_times.txt	hard times	CHAPTER\s[IVXL
807	Dickens/807-hunted_down.txt	hunted down	^[IVXLCM
809	Dickens/809-holiday_romance.txt	holiday romance	^PART\s[IVXLCM

book_id	source_file_path	title	chap_r
810	Dickens/810-george_silvermans_explanation.txt	george silvermans explanation	[A-Z]+\sCHAP
821	Dickens/821-dombey_and_sons.txt	dombey and sons	^\s*CHAPTER\s*[IVXLCM
824	Dickens/824-speeches_of_charles_dickens.txt	speeches of charles dickens	[IVXLCM
872	Dickens/872-reprinted_pieces.txt	reprinted pieces	THE LONG VO $\|THEBEGGING-LETTERWRI$
882	Dickens/882-sketches_by_boz.txt	sketches by boz	^(PREFACE|CHAPTER\s[IVXLC
883	Dickens/883-our_mutual_friend.txt	our mutual friend	^\s*Chap
888	Dickens/888-the_lazy_tour_of_two_idle_apprenti...	the lazy tour of two idle apprentices	CHAPTER\s[IVXLC
912	Dickens/912-the_mudfog_and_other_sketches.txt	the mudfog and other sketches	PUBLIC LIFE OF MR. TULRUMBLE$| REPORT C
914	Dickens/914-the_uncommerical_traveller.txt	the uncommerical traveller	^[IVXLC
916	Dickens/916-sketches_of_young_couples.txt	sketches of young couples	AN URGENT REMONSTRANCE $\|THEYOUNGCOUP$
917	Dickens/917-barnaby_rudge.txt	barnaby rudge	^Chapter\s([0-9]+|the
918	Dickens/918-sketches_of_young_gentlemen.txt	sketches of young gentlemen	THE BASHFUL YOUNG GENTLEMAN$ OUT-AND-OU
922	Dickens/922-sunday_under_three_heads.txt	sunday under three heads	^[IVXLC
927	Dickens/927-the_lamplighter.txt	the lamplighter	
967	Dickens/967-nicholas_nickleby.txt	nicholas nickleby	^(AUTHOR'S PREFACE|CHAPTER 9]+|Conclus
968	Dickens/968-martin_chuzzlewit.txt	martin chuzzlewit	^(PREFACE|CHAPTER\s[A-Z]+[-]?[A-Z
1023	Dickens/1023-bleak_house.txt	bleak house	^\s*(PREFACE|CHAPTER\s*[IVXLCM
1289	Dickens/1289-three_ghost_stories.txt	three ghost stories	THE HAUNTED HOUSE\.|THE TRIAL MURDER\.|T
1394	Dickens/1394-the_holly_tree.txt	the holly tree	^[A-Z]+\sBRA

book_id	source_file_path	title	chap_r
1400	Dickens/1400-great_expectations.txt	great expectations	^\s*Chapter\s*[IVXL
1406	Dickens/1406-the_perils_of_certain_english_pri...	the perils of certain english prisoners	^CHAPTER\s[IVXL
1407	Dickens/1407-a_message_from_the_sea.txt	a message from the sea	^CHAPTER\s[IVXL
1413	Dickens/1413-tom_tiddlers_ground.txt	tom tiddlers ground	^CHAPTER\s[IVXL
1414	Dickens/1414-somebodys_luggage.txt	somebodys luggage	^CHAPTER\s[IVXL
1415	Dickens/1415-doctor_marigold.txt	doctor marigold	* * \
1416	Dickens/1416-mrs_lirripers_lodgings.txt	mrs lirripers lodgings	^CHAPTER\s[IVXL
1421	Dickens/1421-mrs_lirripers_legacy.txt	mrs lirripers legacy	^CHAPTER\s[IVXL
1435	Dickens/1435-miscellaneous_papers.txt	miscellaneous papers	THE AGRICULT INTEREST$|THREATENING LETT
1467	Dickens/1467-some_christmas_stories.txt	some christmas stories	A CHRISTMAS TREE[\.]?|WHAT CHRIST IS AS \
2324	Dickens/2324-a_house_to_let.txt	a house to let	OVER THE |*THEMANCHESTERMARRI* |GOIN
19337	Dickens/19337-a_christmas_carol.txt	a christmas carol	^\s*STAVE\s[A-
20795	Dickens/20795-the_cricket_on_the_hearth.txt	the cricket on the hearth	^CHIRP\
27924	Dickens/27924-mugby_junction.txt	mugby junction	BARBOX BROT |*BARBOXBROTHERSANDC* |I
35536	Dickens/35536-the_poems_and_verses_of_charles_...	the poems and verses of charles dickens	THE VILLAGE COQUE |*THELAMPLIGHTER*|SON

In [10]:
```
CORPUS
```

Out[10]:

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
98	**1**	**0**	**0**	**0**	('The', 'DT')	DT	The	the

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

4969964 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]:
```
55271
```

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]:
```
238366
```

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

Out[16]:
```
['the period',
 'it was the best of times it was the worst of times it was the age of wisdom it was the age of foolishness it was the epoch of belief it was the epoch of incredulity it was the season of light it was the season of darkness it was the spring of hope it was the winter of despair we had everything before us we had nothing before us we were all going direct to heaven we were all going direct the other way in short the period was so far like the present period that some of its noisiest authorities insisted on its being received for good or for evil in the superlative degree of comparison only',
 'there were a king with a large jaw and a queen with a plain face on the throne of england there were a king with a large jaw and a queen with a fair face on the throne of france',
 'in both countries it was clearer than crystal to the lords of the state preserves of loaves and fishes that things in general were settled for ever',
 'it was the year of our lord one thousand seven hundred and seventy five']
```

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>	the
	1	<s>	the	period
	2	the	period	</s>
	3	period	</s>	<s>
	4	</s>	<s>	<s>
...	**...**	...	...	...
238365	**15**	dickens	by	charles
	16	by	charles	dickens
	17	charles	dickens	</s>
	18	dickens	</s>	NaN
	19	</s>	NaN	NaN

5685062 rows × 3 columns

Unigram table

In [144…
```
stop_words = VOCAB.loc[VOCAB.stop == 1].index.values
```

In [145…
```
unigram_df = train.LM[0].sort_values('n', ascending = False)
unigram_df
```

Out[145…

w0	n	mle	p	log_p
<s>	476732	8.385696e-02	8.385696e-02	-3.575926
the	257061	4.521692e-02	4.521692e-02	-4.466993
</s>	238366	4.192848e-02	4.192848e-02	-4.575926
and	181975	3.200933e-02	3.200933e-02	-4.965364
of	136169	2.395207e-02	2.395207e-02	-5.383706
…	…	…	…	…
modernizing	1	1.758996e-07	1.758996e-07	-22.438745
modernised	1	1.758996e-07	1.758996e-07	-22.438745
brimmy	1	1.758996e-07	1.758996e-07	-22.438745
moderator	1	1.758996e-07	1.758996e-07	-22.438745
ĕngine	1	1.758996e-07	1.758996e-07	-22.438745

55272 rows × 4 columns

In [146…
```
unigram_df.filter(regex = '^[^<]', axis = 0)
```

Out[146…

w0	n	mle	p	log_p
the	257061	4.521692e-02	4.521692e-02	-4.466993
and	181975	3.200933e-02	3.200933e-02	-4.965364
of	136169	2.395207e-02	2.395207e-02	-5.383706
to	131034	2.304883e-02	2.304883e-02	-5.439163
a	112660	1.981685e-02	1.981685e-02	-5.657129
…	…	…	…	…
modernizing	1	1.758996e-07	1.758996e-07	-22.438745
modernised	1	1.758996e-07	1.758996e-07	-22.438745
brimmy	1	1.758996e-07	1.758996e-07	-22.438745
moderator	1	1.758996e-07	1.758996e-07	-22.438745
ĕngine	1	1.758996e-07	1.758996e-07	-22.438745

55270 rows × 4 columns

Bigram table

In [147…
```
bigram_df = train.LM[1].sort_values('n', ascending = False)
bigram_df
```

Out[147…

w0	w1	n	mle	mle2	p	log_p
<s>	<s>	238366	4.192849e-02	0.500000	0.448055	-1.158253
</s>	<s>	238365	4.192831e-02	0.999996	0.811768	-0.300860
of	the	27530	4.842516e-03	0.202175	0.143809	-2.797771
<s>	i	23498	4.133289e-03	0.049290	0.044171	-4.500766
in	the	22859	4.020889e-03	0.258294	0.159002	-2.652885
…	…	…	…	…	…	…
his	doubled	1	1.758996e-07	0.000016	0.000017	-15.828756
	doublet	1	1.758996e-07	0.000016	0.000017	-15.828756
	doubt	1	1.758996e-07	0.000016	0.000017	-15.828756
	doughty	1	1.758996e-07	0.000016	0.000017	-15.828756
ĕngine	driver	1	1.758996e-07	1.000000	0.000036	-14.754287

1027728 rows × 5 columns

In [148…
```
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[148…

w0	w1	n	mle	mle2	p	log_p
said	mr	4790	8.425591e-04	0.160523	0.056291	-4.150964
mr	pickwick	2062	3.627050e-04	0.068685	0.024187	-5.369612
said	mrs	1379	2.425656e-04	0.046213	0.016214	-5.946622
dont	know	1304	2.293731e-04	0.198962	0.021108	-5.566092
old	man	1215	2.137180e-04	0.121841	0.018638	-5.745630
…	…	…	…	…	…	…
hired	phaeton	1	1.758996e-07	0.010526	0.000036	-14.756739
	post	1	1.758996e-07	0.010526	0.000036	-14.756739
	service	1	1.758996e-07	0.010526	0.000036	-14.756739

		n	mle	mle2	p	log_p
w0	w1					
	serving	1	1.758996e-07	0.010526	0.000036	-14.756739
ĕngine	driver	1	1.758996e-07	1.000000	0.000036	-14.754287

476314 rows × 5 columns

Trigram table

In [149…

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

Out[149…

			n	mle	mle2	p	log_p
w0	w1	w2					
</s>	<s>	<s>	238365	4.192832e-02	1.000000	7.801894e-05	-13.645816
<s>	<s>	i	23498	4.133290e-03	0.098579	7.691395e-06	-16.988323
		the	15448	2.717298e-03	0.064808	5.056571e-06	-17.593409
		he	9972	1.754071e-03	0.041835	3.264236e-06	-18.224823
		it	8071	1.419686e-03	0.033860	2.642025e-06	-18.529924
…	…	…	…	…	…	…	…
he	closed	one	1	1.758996e-07	0.025641	6.546658e-10	-30.508522
		with	1	1.758996e-07	0.025641	6.546658e-10	-30.508522
	closes	early	1	1.758996e-07	0.500000	6.546658e-10	-30.508522
		his	1	1.758996e-07	0.500000	6.546658e-10	-30.508522
ĕngine	driver	whom	1	1.758996e-07	1.000000	6.546658e-10	-30.508522

2917374 rows × 5 columns

In [150…

```
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[150…

			n	mle	mle2	p	log_p
w0	w1	w2					
said	mr	pickwick	598	1.051880e-04	0.124843	1.960721e-07	-22.282112
sir	said	mr	322	5.663968e-05	0.268110	1.057285e-07	-23.173133

			n	mle	mle2	p	log_p
w0	w1	w2					
said	mr	pecksniff	247	4.344721e-05	0.051566	8.117843e-08	-23.554328
		boffin	203	3.570763e-05	0.042380	6.677580e-08	-23.836099
		dombey	201	3.535583e-05	0.041962	6.612114e-08	-23.850313
...	...	...	...	...	...	...	...
havisham	dwelt	upon	1	1.758996e-07	1.000000	6.546658e-10	-30.508522
	ever	made	1	1.758996e-07	1.000000	6.546658e-10	-30.508522
	going	along	1	1.758996e-07	1.000000	6.546658e-10	-30.508522
	horribly	cruel	1	1.758996e-07	1.000000	6.546658e-10	-30.508522
	however	sir	1	1.758996e-07	1.000000	6.546658e-10	-30.508522

271763 rows × 5 columns

In [152…
```
LIB.type.value_counts()
```

Out[152…
```
stories        26
novel          17
non-fiction     7
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. I NEVER THOUGHT ABOUT GOING TO BE DYING IN THAT NEIGHBOURHOOD BY ACCIDENT.
```

```
02. YES SIR.

03. OF OUR TWO INDIVIDUAL NATURES AND THEY LIVE IN A MOMENT.

04. GIVE HIM TO GO OUT WITH THE OBJECT OF THE CONTENTS OF HIS.

05. BURIED HOW LONG THAT HE WAS CARRIED OFF UPON HER AT THE ALBION IN LITTLE RUS
SELL STREET.

06. FOR A MOMENT.

07. MUCH NEEDED COULD A FEW LEAVES AND THERE HE SAID AGAIN.

08. YES CHILD SAID THE MILITARY GENTLEMAN ADDRESSING THE OLD GENTLEMAN NOT EIGHT
BLUE SKIES IN AS INDIFFERENT AND SO HAVE I MAY SAID MR CHESTER.

09. IT HAD RECEIVED IN TURF AND THE SEDATE FACE IN THE COMPANY OF A BUCKET AND M
RS QUILP HE SAID THAT WHEREVER GAY IS PROBABLY ACQUAINTEDCANNOT SAY A COINCIDENC
E A REMARKABLE KIND OF NEAT LITTLE ROOM PUT A BRIGHTER COUNTENANCE AND AS THERE
CAN BE TAKEN OFF THE GROUND THE STOPPING OF THE UNFORTUNATE CHILDREN WHOSE WERRY
SUBSISTENCE DEPENDS ON THEIR WAY.

10. O NO.

11. I HAD TO TELL ME ALL THEM ANCESTORS OF YOURS I BEG YOUR PARDON AND TELL US.

12. MY DEAR BOY ME AND THANK GOD SAID THE MAGISTRATE.

13. NOT YOU SAYS TWEMLOW AT THE LODGE WHERE SUCH DREAMS COME.

14. THAT DOOR AND SAID THAT BEFORE THE HOUSE.

15. AND WHY IN A VERY DECENT AIM AT ME SIRRAH DEMANDED RALPH.

16. ALL REPLIED NICHOLAS.

17. EVENSON.

18. MR PECKSNIFF SEVERAL TIMES LEAVING MY FRIEND DOMBEY WILL BE PLEASED THAT HE
WAS READING.

19. OTHER SOUND INAUDIBLE.

20. I WONDER HOW IT CHIRPED.
```

Examining redundancy for unigrams, bigrams, trigrams $\rightarrow$ 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, 59]
```

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]:

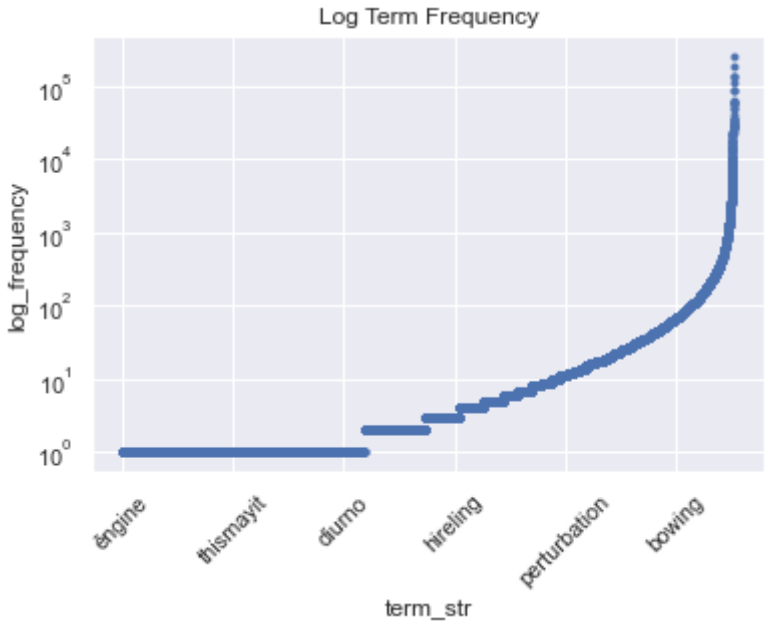
w1	said	heard
w0		
he	1855	197
she	692	59

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

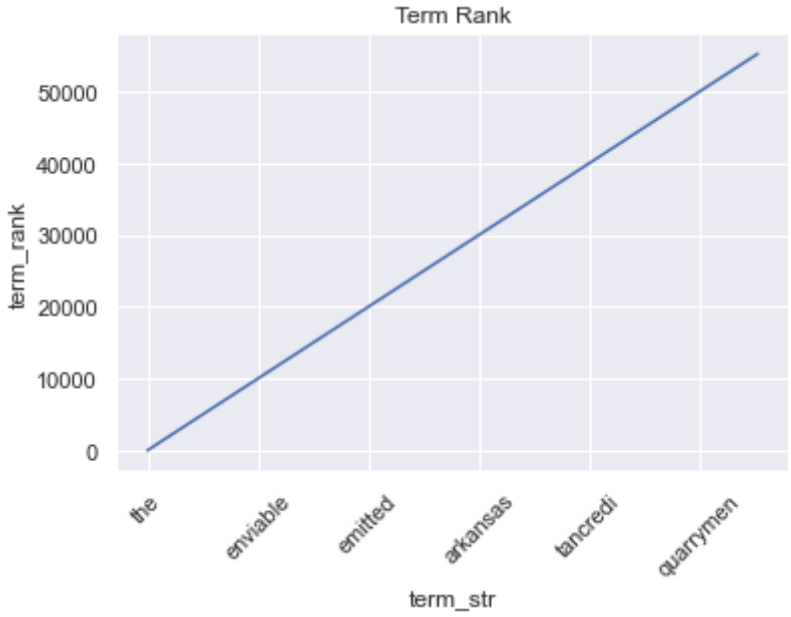
In [30]:
```
VOCAB.n.sort_values().plot(ylabel = "log_frequency", logy=True, style = '.', rot
```



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]:

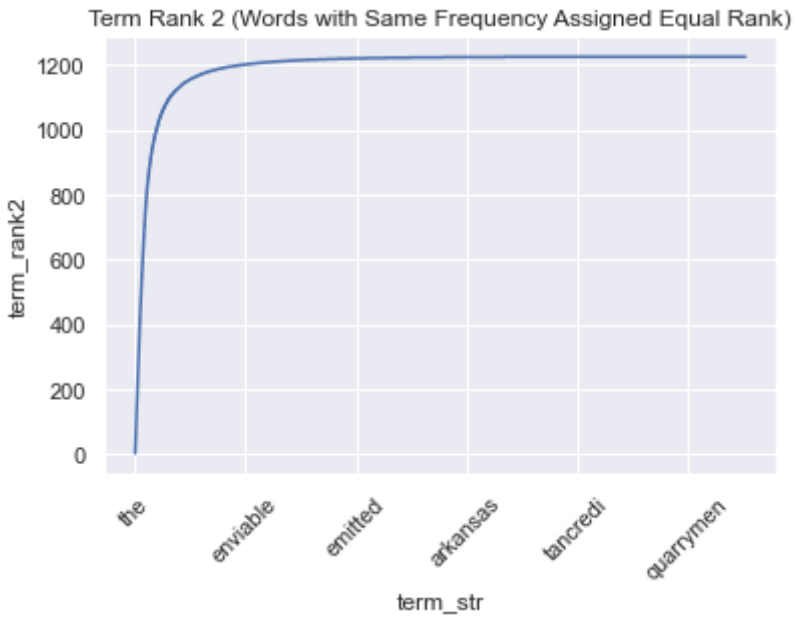
	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
term_str										
the	1	257061	3	0.051723	4.273054	DT	20	{'NNP', 'CD', 'NNS', 'FW', 'POS', 'PRP', 'CC',...	1	
and	2	181975	3	0.036615	4.771425	CC	18	{'NNP', 'CC', 'VBD', 'CD', 'DT', 'PDT', 'VBN',...	1	
of	3	136169	2	0.027398	5.189767	IN	18	{'NNP', 'PRP', 'VBD', 'CD', 'PDT', 'VBN', 'VBP...	1	
to	4	131034	2	0.026365	5.245224	TO	21	{'NNP', 'CD', 'EX', 'NNS', 'FW', 'POS', 'CC', ...	1	
a	5	112660	1	0.022668	5.463190	DT	19	{'NNP', 'CD', 'NNS', 'FW', 'POS', 'PRP', 'VBN'...	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
```

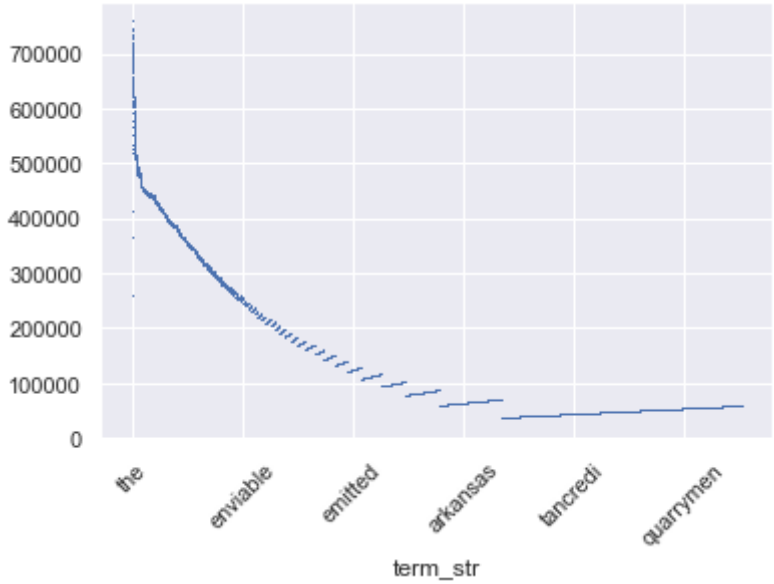


Compute Zipf's k using `term_rank` and `term_rank2`

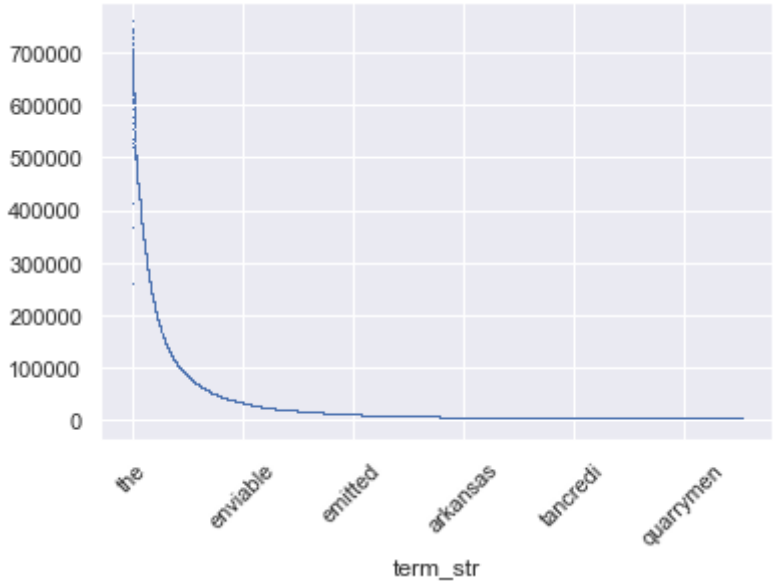
- **Equation**: $k = fr$

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);
```



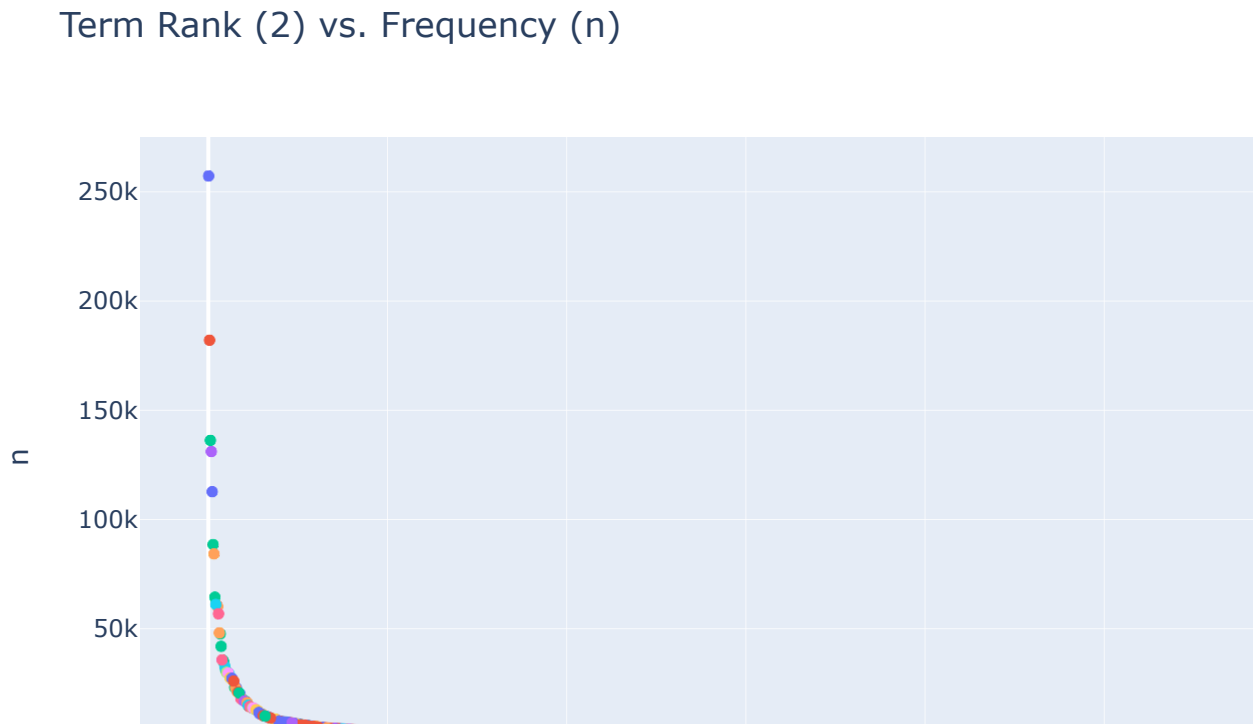
In [38]:
```
VOCAB.zipf_k2.plot(style = ',', rot = 45);
```

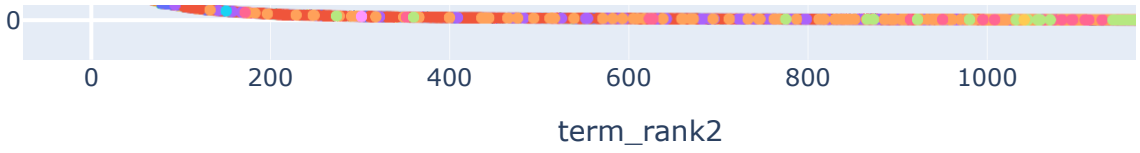


Rank vs. N (frequency n)

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_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop
the	1	257061	3	5.172286e-02	4.273054	DT	20	{'NNP', 'CD', 'NNS', 'FW', 'POS', 'PRP', 'CC',...	1
and	2	181975	3	3.661492e-02	4.771425	CC	18	{'NNP', 'CC', 'VBD', 'CD', 'DT', 'PDT', 'VBN',...	1
of	3	136169	2	2.739836e-02	5.189767	IN	18	{'NNP', 'PRP', 'VBD', 'CD', 'PDT', 'VBN', 'VBP...	1
to	4	131034	2	2.636515e-02	5.245224	TO	21	{'NNP', 'CD', 'EX', 'NNS', 'FW', 'POS', 'CC', ...	1

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop
a	5	112660	1	2.266815e-02	5.463190	DT	19	{'NNP', 'CD', 'NNS', 'FW', 'POS', 'PRP', 'VBN'...	1
...	...	...	...	...	...	...	...	...	...
lushed	55268	1	6	2.012085e-07	22.244805	VBP	1	{'VBP'}	0
lurker	55269	1	6	2.012085e-07	22.244805	NN	1	{'NN'}	0
lunns	55270	1	5	2.012085e-07	22.244805	NNP	1	{'NNP'}	0
lungsa	55271	1	6	2.012085e-07	22.244805	NN	1	{'NN'}	0
ěngine	55272	1	6	2.012085e-07	22.244805	NNP	1	{'NNP'}	0

55272 rows × 20 columns

Document-Term Count Matrix `DTCM`

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

Out[43]:

book_id	chap_id	term_str	0	1	10	100	1000	10000	10000l	1000l	100l	1030	...	zooks	zool
98	**1**		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	
	3		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	
	5		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
...	**...**		...	...	...	...	...	...	...	...	...	...	...	...	
35536	**9**		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
	10		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
	11		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
	12		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
	13		0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

1182 rows × 55270 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 [154…
```
SIGS.head(10).index.values
```

Out[154…
```
array(['seem', 'entered', 'sleep', 'cut', 'top', 'windows', 'dress',
       'thank', 'worse', 'sent'], 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]:

	term_str	0	1	10	100	1000	10000	10000l	1000l	100l	1030	…	zooks
book_id	chap_id												
98	1	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	2	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	3	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	4	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	5	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
35536	9	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	10	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	11	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	12	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0
	13	0.0	0.255548	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0

1182 rows × 55270 columns

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

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

Out[48]:

	term_str	seem	entered	sleep	cut	top	windows	dress	thanl
book_id	chap_id								
98	1	0.0	0.000000	0.000000	0.018297	0.000000	0.000000	0.000000	0.00000(
	2	0.0	0.000000	0.000000	0.007300	0.014634	0.000000	0.000000	0.00000(
	3	0.0	0.000000	0.000000	0.000000	0.011779	0.011779	0.000000	0.00000(
	4	0.0	0.000000	0.007121	0.000000	0.000000	0.000000	0.000000	0.01414:
	5	0.0	0.005042	0.000000	0.005054	0.010131	0.010131	0.005019	0.00000(
...	...	...	...	...	...	...	...	...	...
35536	9	0.0	0.000000	0.024499	0.000000	0.000000	0.000000	0.000000	0.00000(
	10	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000(
	11	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000(
	12	0.0	0.000000	0.027797	0.000000	0.000000	0.000000	0.000000	0.00000(
	13	0.0	0.000000	0.065703	0.000000	0.000000	0.000000	0.000000	0.00000(

1182 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())
```

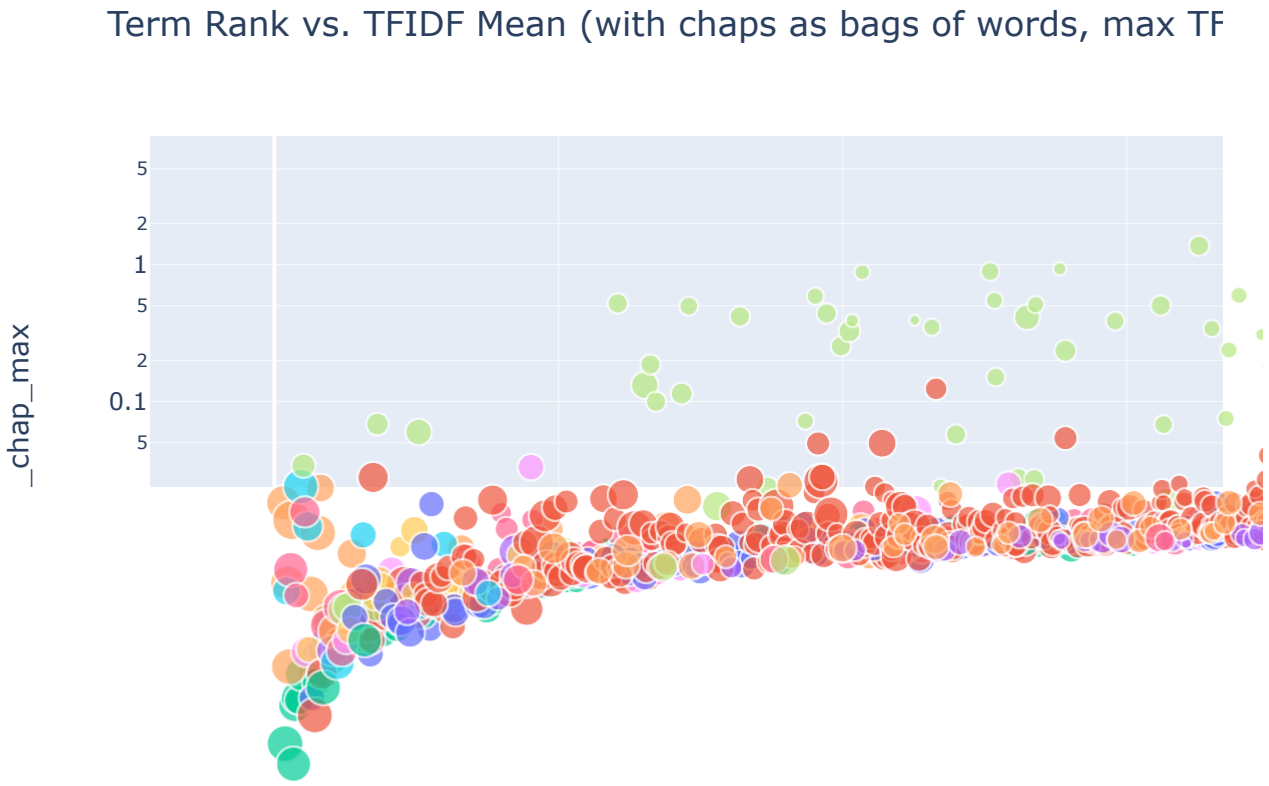
```
max term TFIDF value: 16.677083003000984
max TFIDF term: says
```

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: 6.716943644781024
(Book_id, chap_id) with max total TFIDF:  (786, 2)
Title of book with max total TFIDF: Hard Times
```

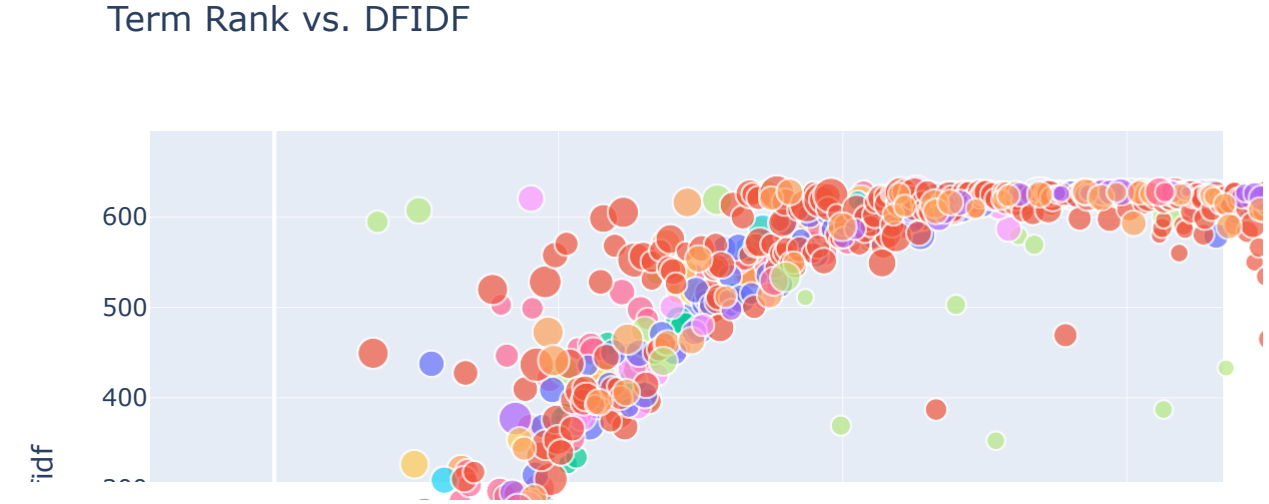
Rank and TFIDF Mean

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	1	10	100	1000	10000	10000l	1000l	
book_id									
98	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
564	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
580	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
588	0.043555	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
644	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
650	0.000000	0.000558	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
653	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
675	0.000000	0.003390	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
676	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
699	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
700	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
730	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
766	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
786	0.006187	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
807	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

term_str	0	1	10	100	1000	10000	10000l	1000l	
book_id									
809	0.000000	0.005130	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
810	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
821	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
824	0.000000	0.020283	0.012691	0.005939	0.001913	0.001257	0.000000	0.000000	0.0
872	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
882	0.000166	0.000444	0.000000	0.000000	0.000000	0.000000	0.001721	0.000223	0.0
883	0.000000	0.000312	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
888	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
912	0.006222	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
914	0.000000	0.001287	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
916	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
917	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
918	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
922	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
927	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
967	0.000000	0.000451	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
968	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1023	0.000000	0.000285	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1289	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1394	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1400	0.000000	0.000696	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1406	0.000000	0.006614	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1407	0.000000	0.006741	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1413	0.000000	0.030722	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1414	0.381109	0.084841	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1415	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1416	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1421	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1435	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1467	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
2324	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
19337	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
20795	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
27924	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

term_str	0	1	10	100	1000	10000	10000l	1000l	
book_id									
35536	0.000000	0.019658	0.019656	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

50 rows × 55270 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	seem	entered	sleep	cut	top	windows	dress	thank	
book_id									
98	0.002393	0.003802	0.003404	0.002195	0.003124	0.004978	0.004351	0.005878	0.0
564	0.010476	0.002771	0.004620	0.003763	0.004910	0.009007	0.004279	0.009971	0.0
580	0.001633	0.006892	0.004072	0.003512	0.004892	0.002329	0.002140	0.003107	0.0
588	0.005328	0.002433	0.003914	0.004394	0.005007	0.001695	0.001646	0.000554	0.0
644	0.001645	0.002681	0.008029	0.002687	0.003306	0.003628	0.002669	0.003647	0.0
650	0.002964	0.004217	0.002264	0.002753	0.005130	0.010857	0.004136	0.000236	0.0
653	0.006889	0.002919	0.006016	0.006939	0.002016	0.003991	0.004952	0.006012	0.0
675	0.005027	0.001698	0.003040	0.003983	0.005156	0.007240	0.003041	0.000871	0.0
676	0.008455	0.001075	0.000930	0.006221	0.004716	0.002210	0.004477	0.004379	0.0
699	0.000934	0.000836	0.000395	0.004907	0.001019	0.001196	0.003669	0.000059	0.0
700	0.003311	0.005115	0.009056	0.002884	0.002256	0.004031	0.004122	0.005187	0.0
730	0.001835	0.006976	0.005113	0.003666	0.004380	0.002570	0.002549	0.002576	0.0
766	0.006026	0.003700	0.006636	0.003710	0.004879	0.004065	0.004315	0.007947	0.0
786	0.004518	0.002056	0.004104	0.003240	0.000131	0.002032	0.004440	0.008957	0.0
807	0.017699	0.000000	0.000000	0.002110	0.002115	0.002115	0.000000	0.016911	0.0
809	0.000000	0.002962	0.000000	0.009059	0.007700	0.000000	0.001310	0.004514	0.0
810	0.007320	0.006486	0.000000	0.000000	0.000000	0.010435	0.000000	0.005908	0.0
821	0.003427	0.003858	0.004322	0.003516	0.002988	0.004392	0.003857	0.008407	0.0
824	0.002665	0.000538	0.000719	0.000705	0.000981	0.000272	0.000527	0.008800	0.0
872	0.004458	0.001253	0.006411	0.004847	0.005617	0.004636	0.001825	0.000664	0.0
882	0.002308	0.003893	0.002298	0.003083	0.005221	0.004121	0.006375	0.001652	0.0
883	0.006677	0.002344	0.003785	0.006628	0.003369	0.003248	0.006358	0.008766	0.0
888	0.003790	0.003175	0.003524	0.002611	0.006719	0.006547	0.001476	0.001917	0.0
912	0.000000	0.005130	0.001595	0.001858	0.008419	0.001955	0.005252	0.003798	0.0

term_str	seem	entered	sleep	cut	top	windows	dress	thank	
book_id									
914	0.005221	0.001883	0.003344	0.003503	0.003771	0.005822	0.003032	0.000712	0.0
916	0.005401	0.000000	0.000000	0.002974	0.002333	0.001161	0.008105	0.001173	0.0
917	0.003310	0.005957	0.007799	0.003352	0.003465	0.004823	0.006813	0.005851	0.0
918	0.001018	0.001018	0.000000	0.001635	0.001527	0.002882	0.001434	0.001600	0.0
922	0.005485	0.000000	0.000000	0.000000	0.001632	0.003634	0.006466	0.000000	0.0
927	0.000000	0.003765	0.000000	0.007548	0.003783	0.000000	0.000000	0.000000	0.0
967	0.005470	0.002989	0.004341	0.003985	0.005311	0.002392	0.003876	0.004675	0.0
968	0.004166	0.004235	0.005099	0.004158	0.005300	0.002521	0.003120	0.006518	0.0
1023	0.004549	0.001671	0.004547	0.002342	0.002594	0.005166	0.005302	0.010107	0.0
1289	0.005463	0.005458	0.005007	0.008711	0.008324	0.008017	0.002515	0.000000	0.0
1394	0.003727	0.000000	0.006181	0.007812	0.011454	0.007013	0.002900	0.008700	0.0
1400	0.002926	0.004321	0.003929	0.007371	0.004933	0.006148	0.006626	0.004032	0.0
1406	0.001054	0.002751	0.005927	0.008684	0.002118	0.001704	0.006936	0.003378	0.0
1407	0.000000	0.003458	0.000000	0.003466	0.002137	0.002137	0.008025	0.002117	0.0
1413	0.002205	0.000000	0.009781	0.002210	0.000000	0.004102	0.002195	0.000000	0.0
1414	0.002077	0.001345	0.002859	0.002859	0.000000	0.008698	0.000000	0.002111	0.0
1415	0.000000	0.000000	0.000000	0.000000	0.006526	0.000000	0.000000	0.000000	0.0
1416	0.000000	0.001152	0.003464	0.003464	0.000000	0.001157	0.001147	0.000000	0.0
1421	0.000000	0.000000	0.000000	0.000000	0.012617	0.002415	0.002393	0.012502	0.0
1435	0.001261	0.000848	0.002496	0.002113	0.001916	0.000000	0.004557	0.000844	0.0
1467	0.005943	0.001306	0.006087	0.004426	0.001110	0.008658	0.000000	0.000550	0.0
2324	0.001191	0.004067	0.003903	0.002297	0.004299	0.006567	0.002526	0.006794	0.0
19337	0.000790	0.007504	0.002705	0.006386	0.006561	0.007194	0.006789	0.002413	0.0
20795	0.001823	0.005518	0.007296	0.002834	0.005553	0.001894	0.002724	0.000000	0.0
27924	0.006111	0.002694	0.004254	0.004628	0.005332	0.002912	0.001697	0.002605	0.0
35536	0.000000	0.001656	0.013753	0.000000	0.000000	0.000000	0.004330	0.002165	0.0

50 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
98	dickens	a tale of two cities	dickens 98: a tale of two cities
564	dickens	the mystery of edwin drood	dickens 564: the mystery of edwin drood
580	dickens	the pickwick papers	dickens 580: the pickwick papers
588	dickens	master humphreys clock	dickens 588: master humphreys clock
644	dickens	the haunted man and the ghosts bargain	dickens 644: the haunted man and the ghosts ba...
650	dickens	pictures from italy	dickens 650: pictures from italy
653	dickens	the chimes	dickens 653: the chimes
675	dickens	american notes	dickens 675: american notes
676	dickens	the battle of life	dickens 676: the battle of life
699	dickens	a childs history of england	dickens 699: a childs history of england
700	dickens	the old curiosity shop	dickens 700: the old curiosity shop
730	dickens	oliver twist	dickens 730: oliver twist
766	dickens	david copperfield	dickens 766: david copperfield
786	dickens	hard times	dickens 786: hard times
807	dickens	hunted down	dickens 807: hunted down
809	dickens	holiday romance	dickens 809: holiday romance
810	dickens	george silvermans explanation	dickens 810: george silvermans explanation
821	dickens	dombey and sons	dickens 821: dombey and sons
824	dickens	speeches of charles dickens	dickens 824: speeches of charles dickens
872	dickens	reprinted pieces	dickens 872: reprinted pieces
882	dickens	sketches by boz	dickens 882: sketches by boz
883	dickens	our mutual friend	dickens 883: our mutual friend
888	dickens	the lazy tour of two idle apprentices	dickens 888: the lazy tour of two idle apprent...
912	dickens	the mudfog and other sketches	dickens 912: the mudfog and other sketches
914	dickens	the uncommerical traveller	dickens 914: the uncommerical traveller
916	dickens	sketches of young couples	dickens 916: sketches of young couples
917	dickens	barnaby rudge	dickens 917: barnaby rudge
918	dickens	sketches of young gentlemen	dickens 918: sketches of young gentlemen

book_id	author	title	label
922	dickens	sunday under three heads	dickens 922: sunday under three heads
927	dickens	the lamplighter	dickens 927: the lamplighter
967	dickens	nicholas nickleby	dickens 967: nicholas nickleby
968	dickens	martin chuzzlewit	dickens 968: martin chuzzlewit
1023	dickens	bleak house	dickens 1023: bleak house
1289	dickens	three ghost stories	dickens 1289: three ghost stories
1394	dickens	the holly tree	dickens 1394: the holly tree
1400	dickens	great expectations	dickens 1400: great expectations
1406	dickens	the perils of certain english prisoners	dickens 1406: the perils of certain english pr...
1407	dickens	a message from the sea	dickens 1407: a message from the sea
1413	dickens	tom tiddlers ground	dickens 1413: tom tiddlers ground
1414	dickens	somebodys luggage	dickens 1414: somebodys luggage
1415	dickens	doctor marigold	dickens 1415: doctor marigold
1416	dickens	mrs lirripers lodgings	dickens 1416: mrs lirripers lodgings
1421	dickens	mrs lirripers legacy	dickens 1421: mrs lirripers legacy
1435	dickens	miscellaneous papers	dickens 1435: miscellaneous papers
1467	dickens	some christmas stories	dickens 1467: some christmas stories
2324	dickens	a house to let	dickens 2324: a house to let
19337	dickens	a christmas carol	dickens 19337: a christmas carol
20795	dickens	the cricket on the hearth	dickens 20795: the cricket on the hearth
27924	dickens	mugby junction	dickens 27924: mugby junction
35536	dickens	the poems and verses of charles dickens	dickens 35536: the poems and verses of charles...

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
98       98          1.000000
         564         0.182350
         580        -0.002878
         588         0.053118
         644         0.321641
                        ...
35536    2324        0.075841
         19337       0.057664
         20795       0.008652
         27924       0.033643
         35536       1.000000
Length: 2500, 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**	
564	**98**	0.817650
580	**98**	1.002878
	564	0.877140
588	**98**	0.946882
	564	0.787186
...	**...**	...
35536	**1467**	0.833318
	2324	0.924159
	19337	0.942336
	20795	0.991348
	27924	0.966357

1225 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								
564	98	0.817650	2.181992	0.102131	0.705128	0.248603	0.560497	0.017000	0.0
580	98	1.002878	2.289080	0.110589	0.812632	0.330185	0.651664	0.006000	0.0
	564	0.877140	3.089993	0.164050	0.906774	0.411119	0.766355	0.047047	0.1
588	98	0.946882	2.587732	0.125556	0.770693	0.296984	0.689925	0.123246	0.1
	564	0.787186	2.589356	0.115496	0.878369	0.385766	0.828921	0.054162	0.1
	580	0.721002	2.894603	0.139821	0.776074	0.301146	0.694771	0.134134	0.2
644	98	0.678359	2.233104	0.105889	0.811974	0.329651	0.668478	0.010010	0.0
	564	0.891082	3.092410	0.154481	0.832096	0.346192	0.731800	0.117117	0.2
	580	0.985001	2.331377	0.104579	0.846610	0.358374	0.767106	0.036145	0.1
	588	0.984815	2.082619	0.103715	0.739933	0.273750	0.561057	0.006000	0.0
650	98	0.871224	2.152545	0.108974	0.795297	0.316249	0.607677	0.007000	0.0
	564	0.975258	2.305293	0.103135	0.679688	0.230988	0.542875	0.006000	0.0
	580	1.049085	2.750142	0.142709	0.770513	0.296845	0.626226	0.021021	0.0
	588	0.995085	4.381819	0.222845	0.999091	0.499091	1.055990	0.451256	0.4
	644	1.040313	3.624371	0.256947	1.177945	0.693777	1.093038	0.394975	0.4
653	98	0.723809	3.706198	0.211257	1.039313	0.540086	1.052495	0.409045	0.4
	564	0.910575	1.936816	0.089189	0.634497	0.201293	0.502645	0.006000	0.0
	580	0.828846	3.064535	0.151449	1.008603	0.508640	0.922165	0.060241	0.1

		corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0	
doc_a	doc_b								
	588	0.875357	2.195757	0.106937	0.807463	0.325998	0.654300	0.014014	0.0
	644	0.658470	2.283973	0.101496	0.808304	0.326678	0.703473	0.008000	0.0

Hierarchical agglomerative cluster diagrams for the distance measures

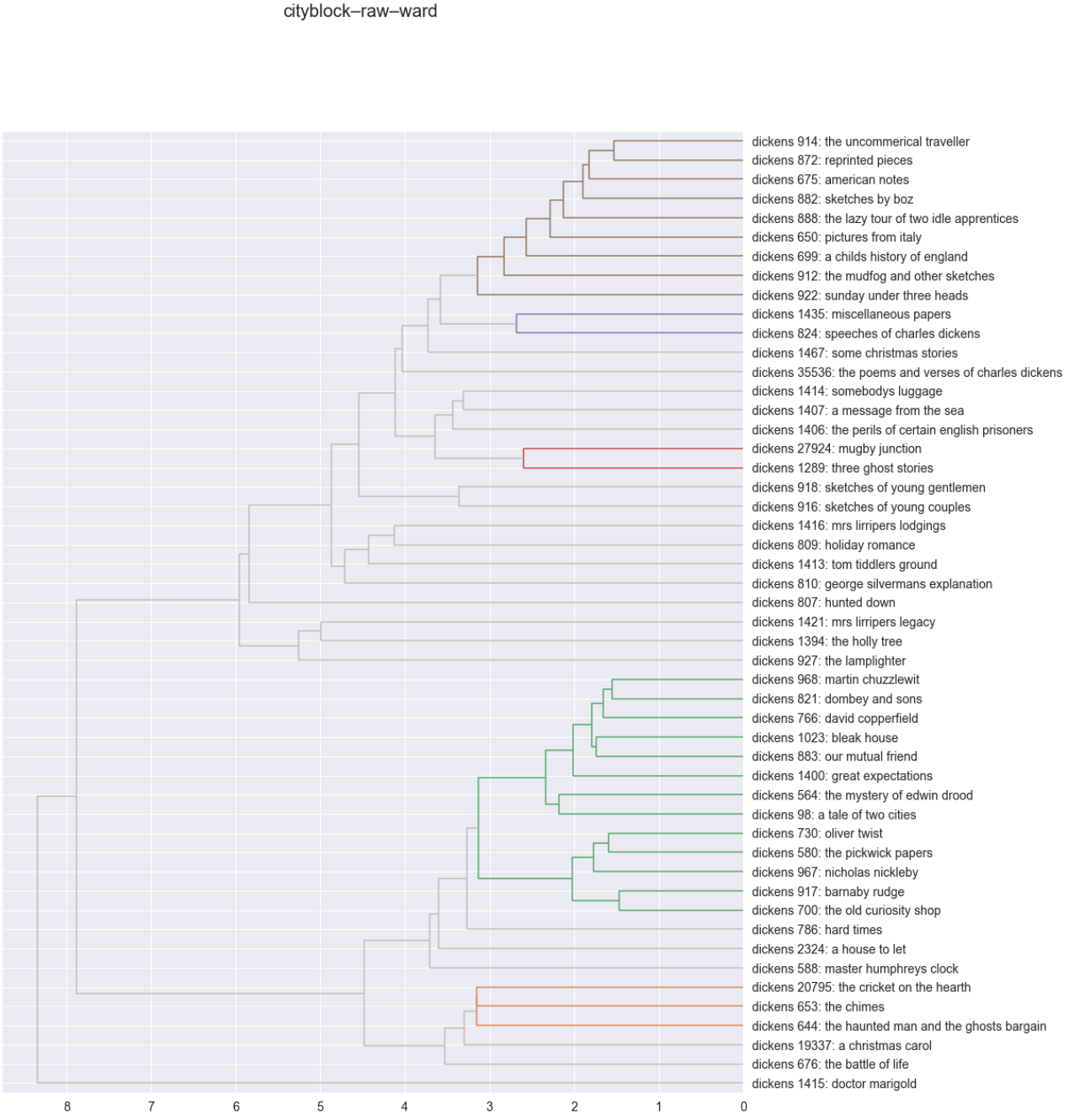
In [67]:
```
LIB['label'] = book_DOC['label']
```

In [155…
```
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 [156…
```
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_24727/163819248.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>
```

euclidean–raw–weighted

<Figure size 432x288 with 0 Axes>

euclidean–l2–ward

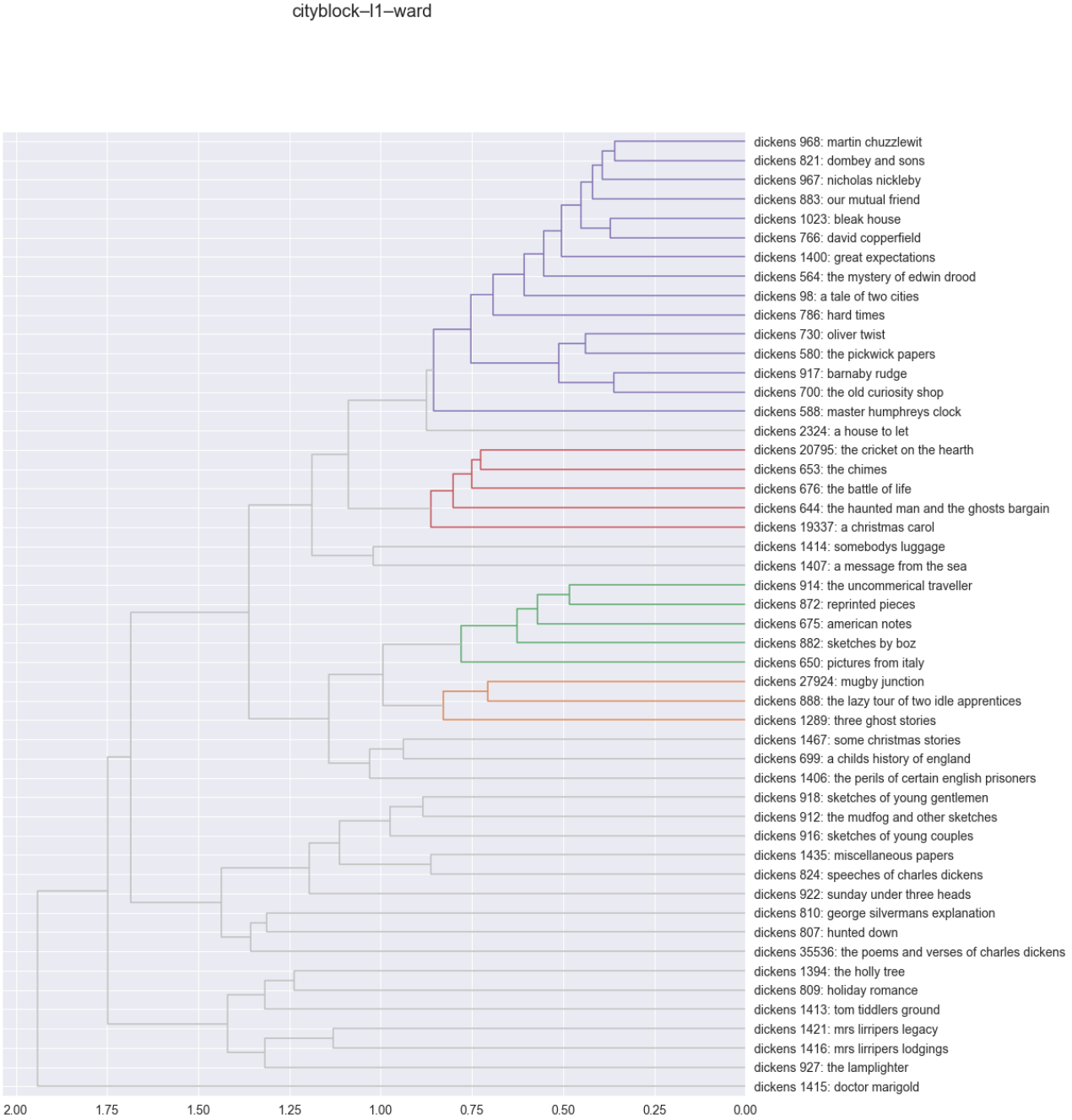

```
<Figure size 432x288 with 0 Axes>
```

<Figure size 432x288 with 0 Axes>


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

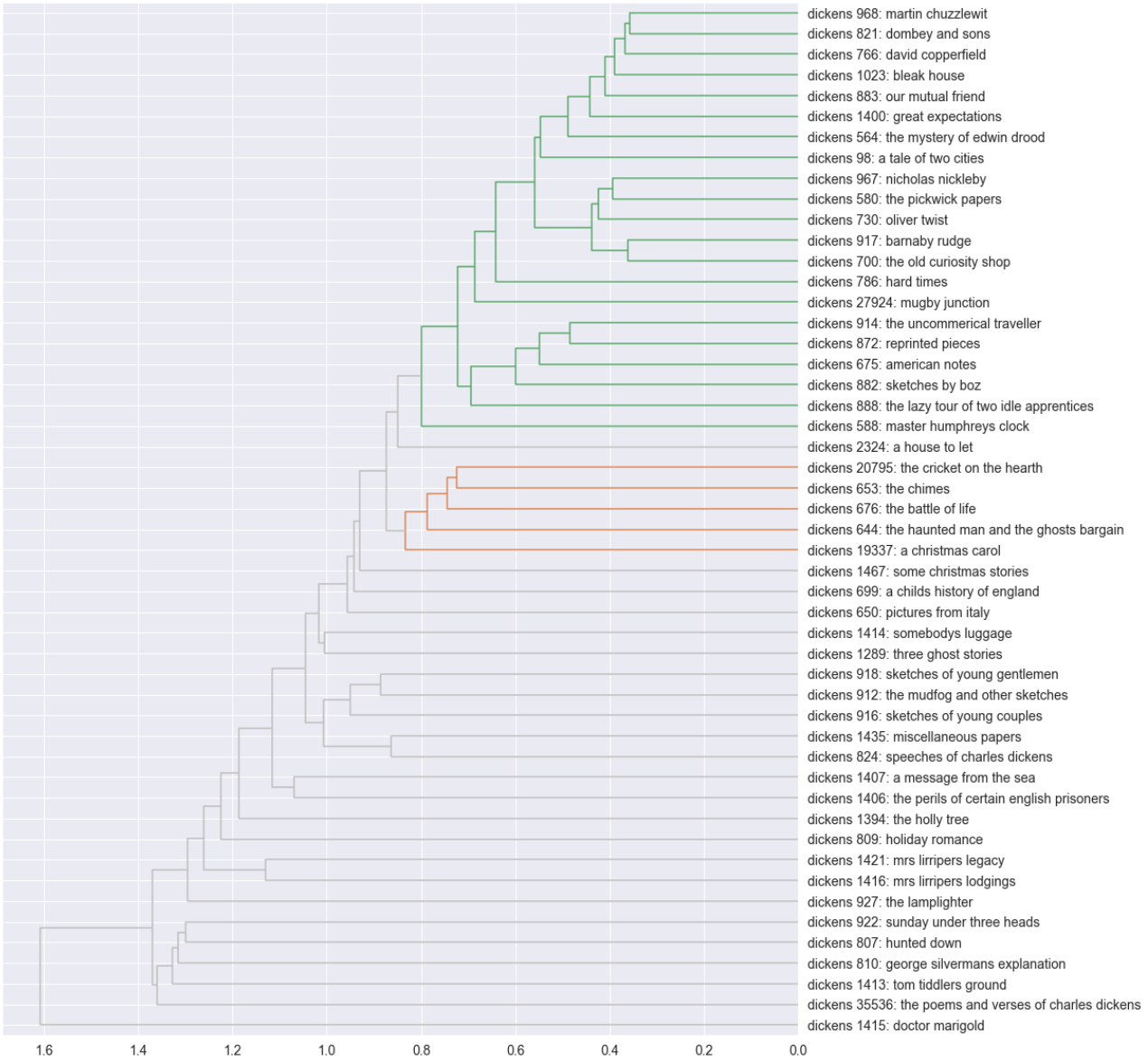
cosine–raw–weighted

<Figure size 432x288 with 0 Axes>

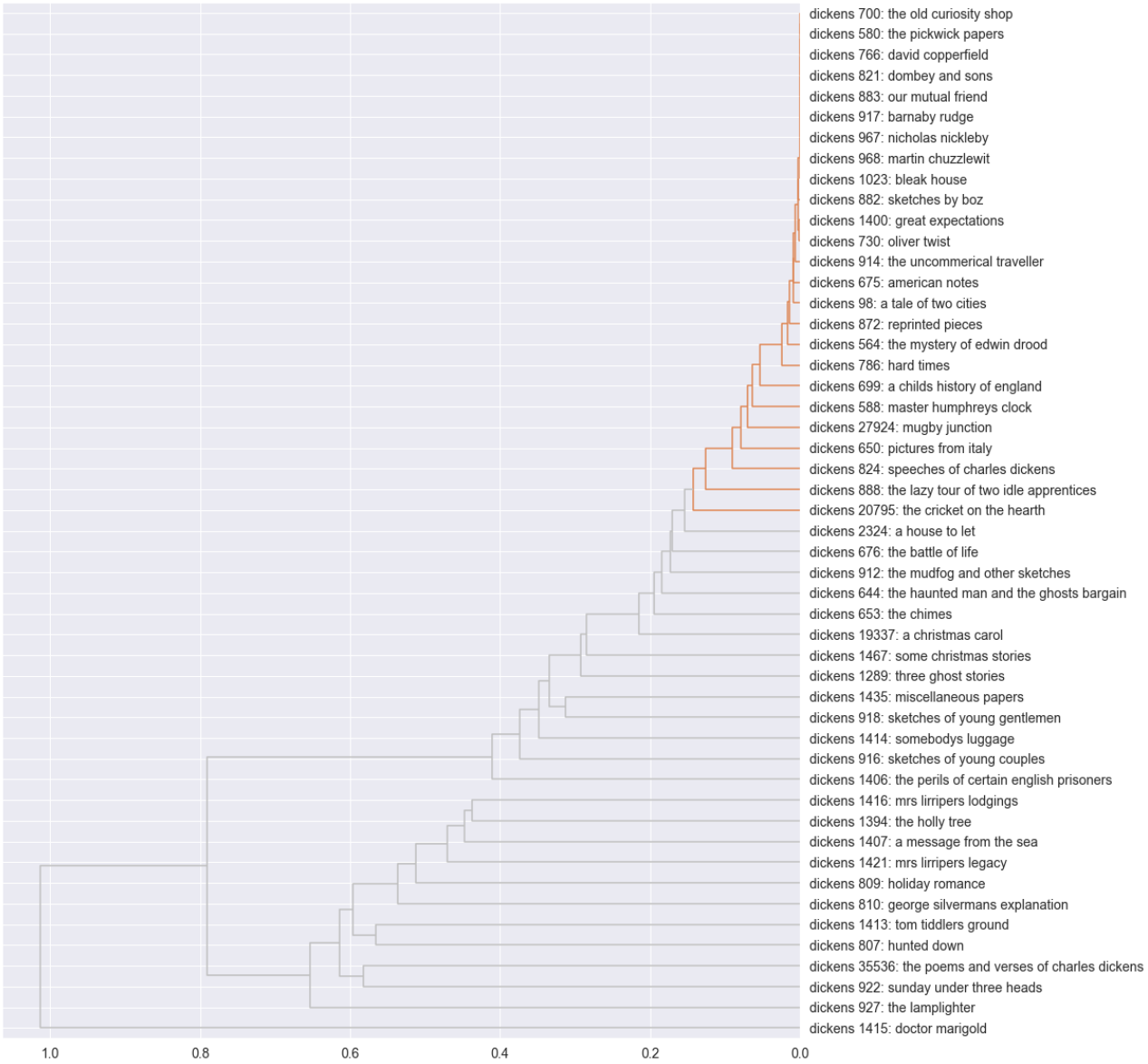


<Figure size 432x288 with 0 Axes>

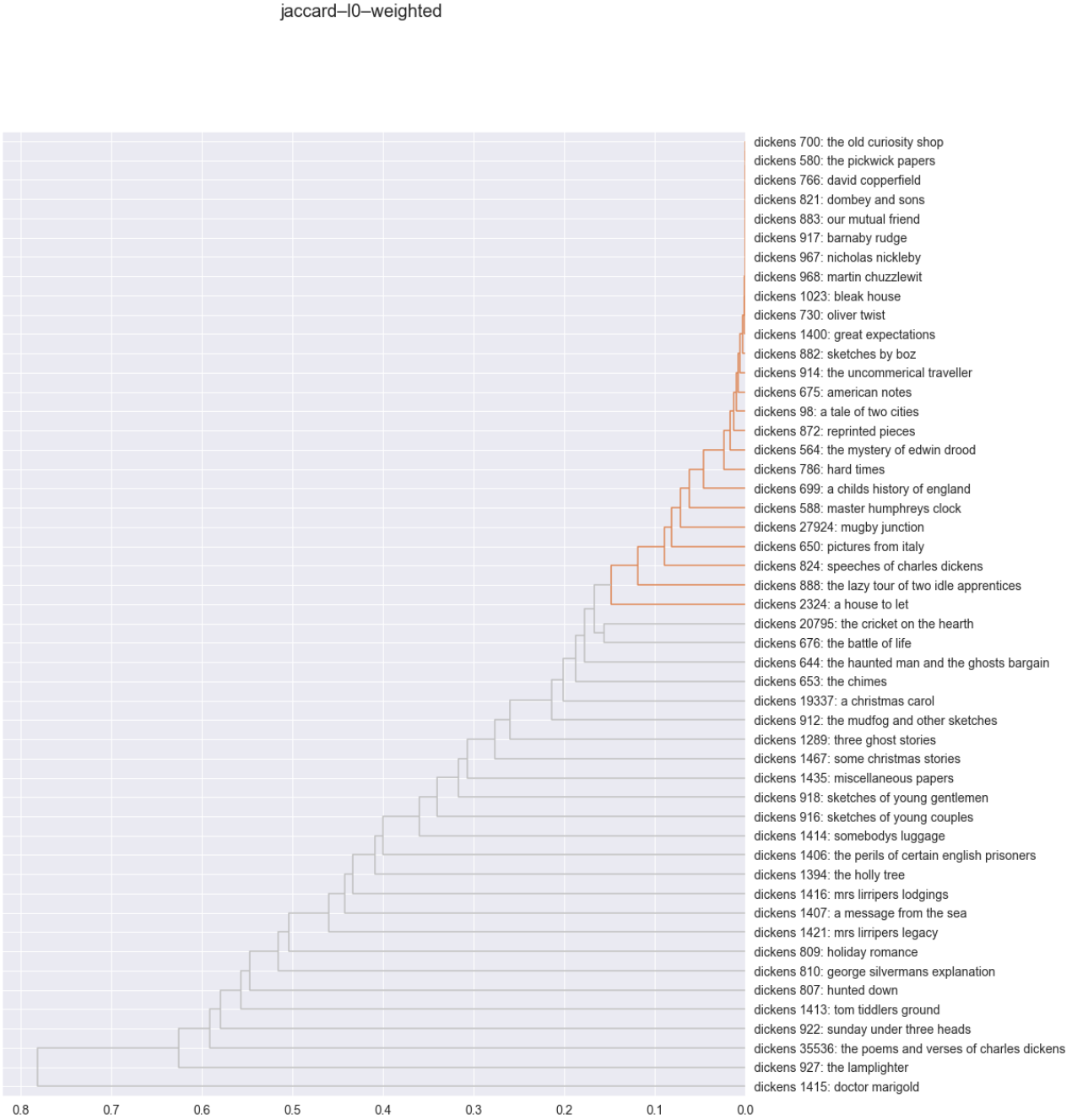
cityblock–l1–weighted



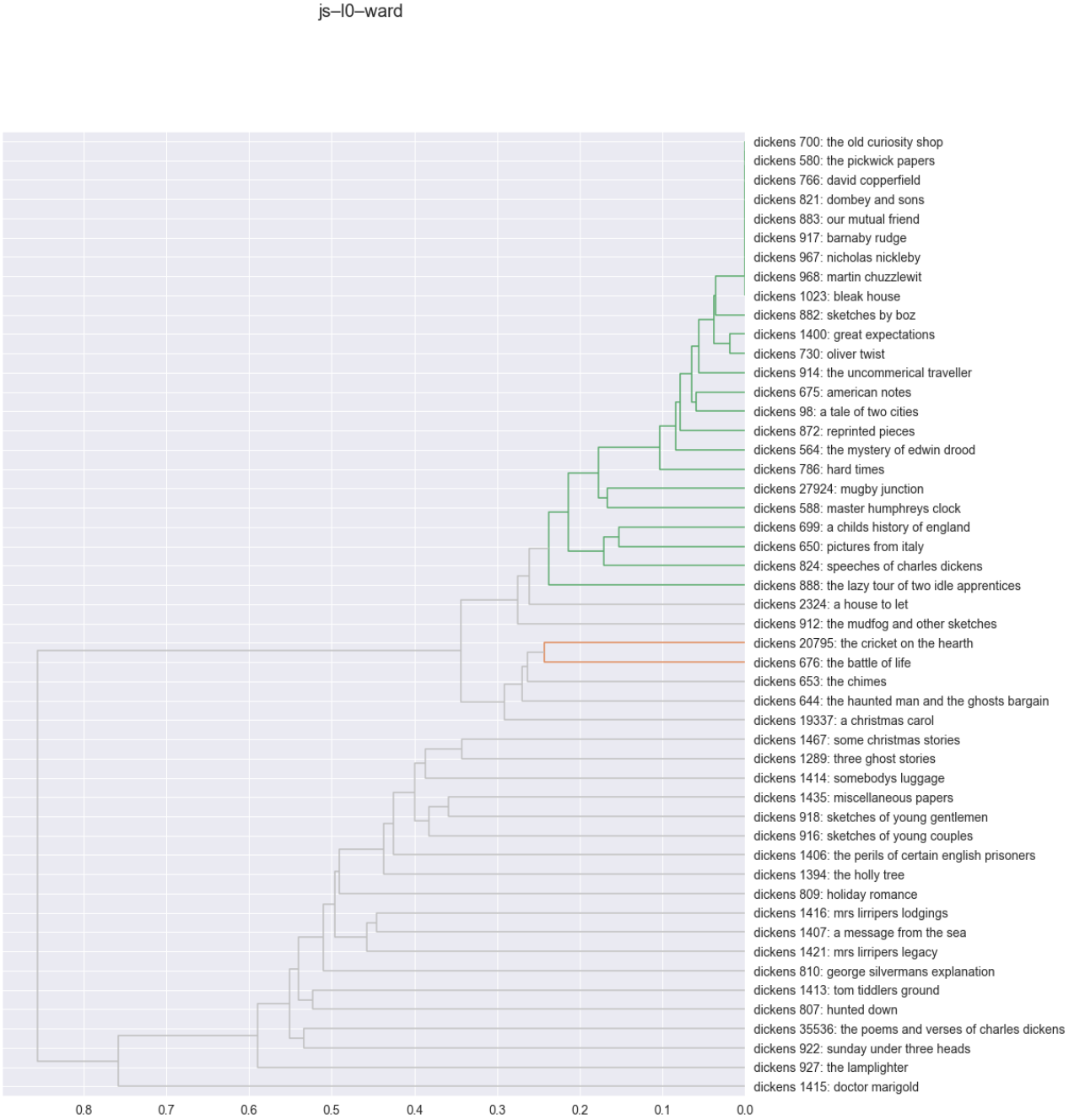
<Figure size 432x288 with 0 Axes>



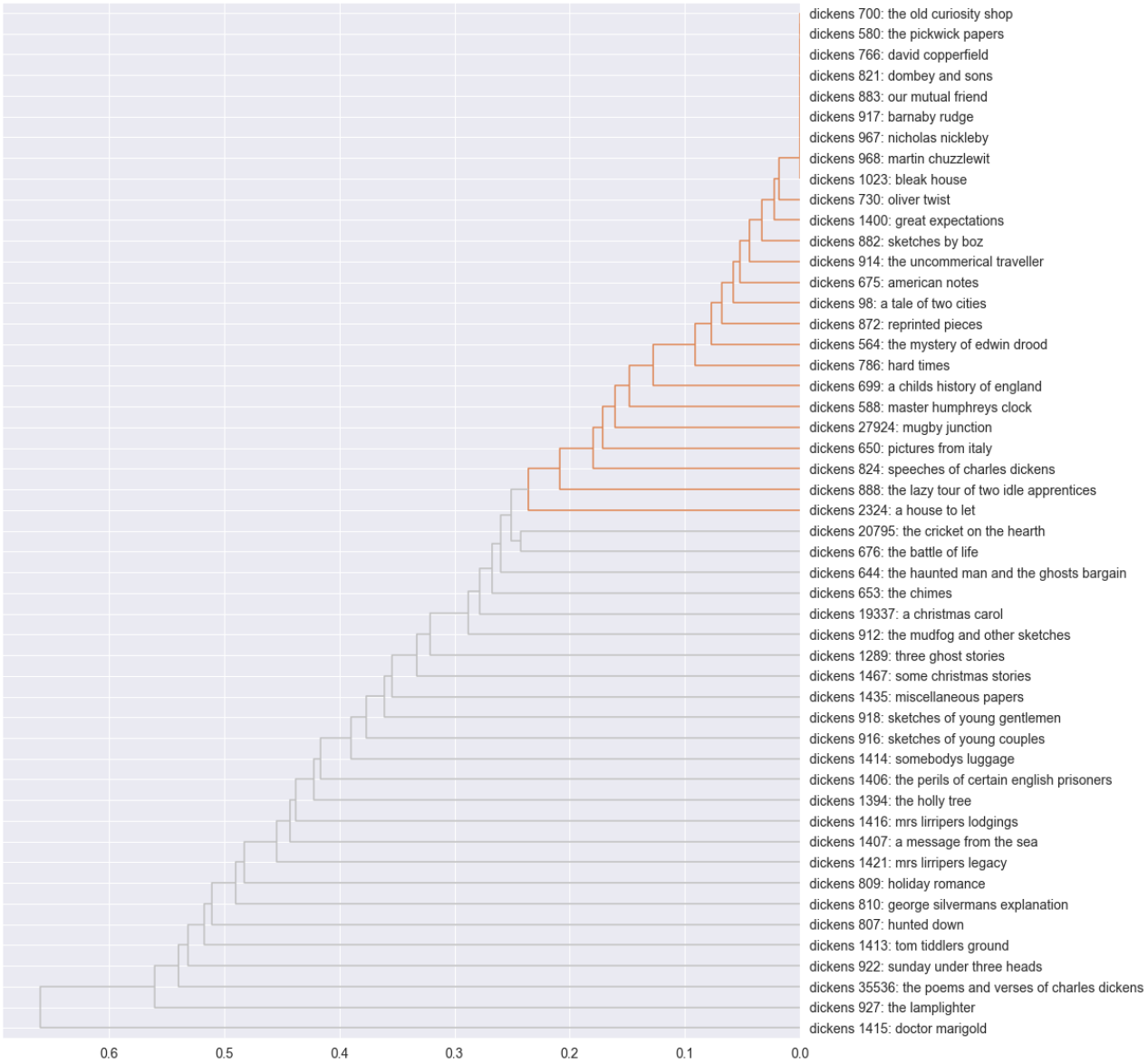

```
<Figure size 432x288 with 0 Axes>
```




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

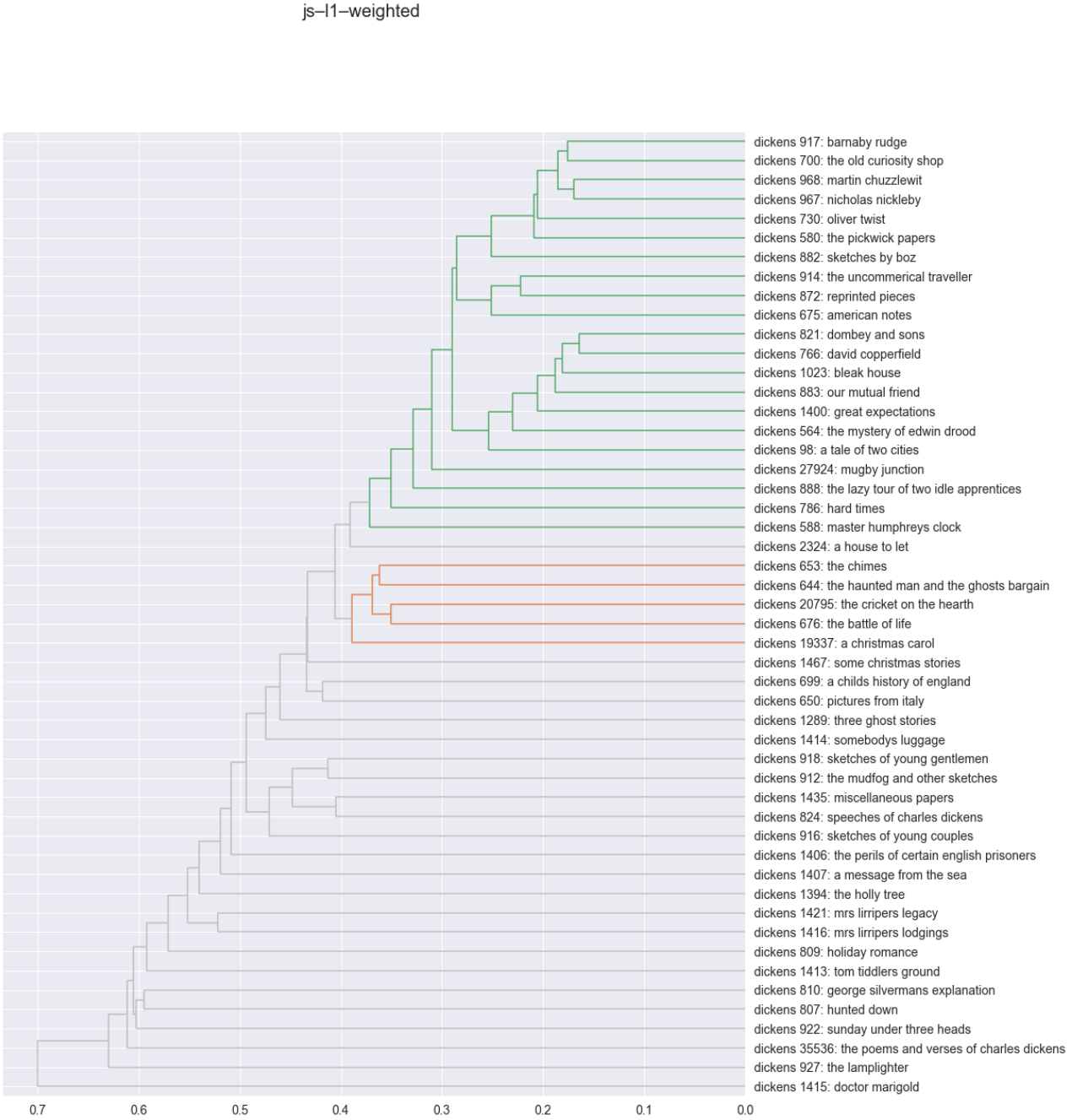


<Figure size 432x288 with 0 Axes>



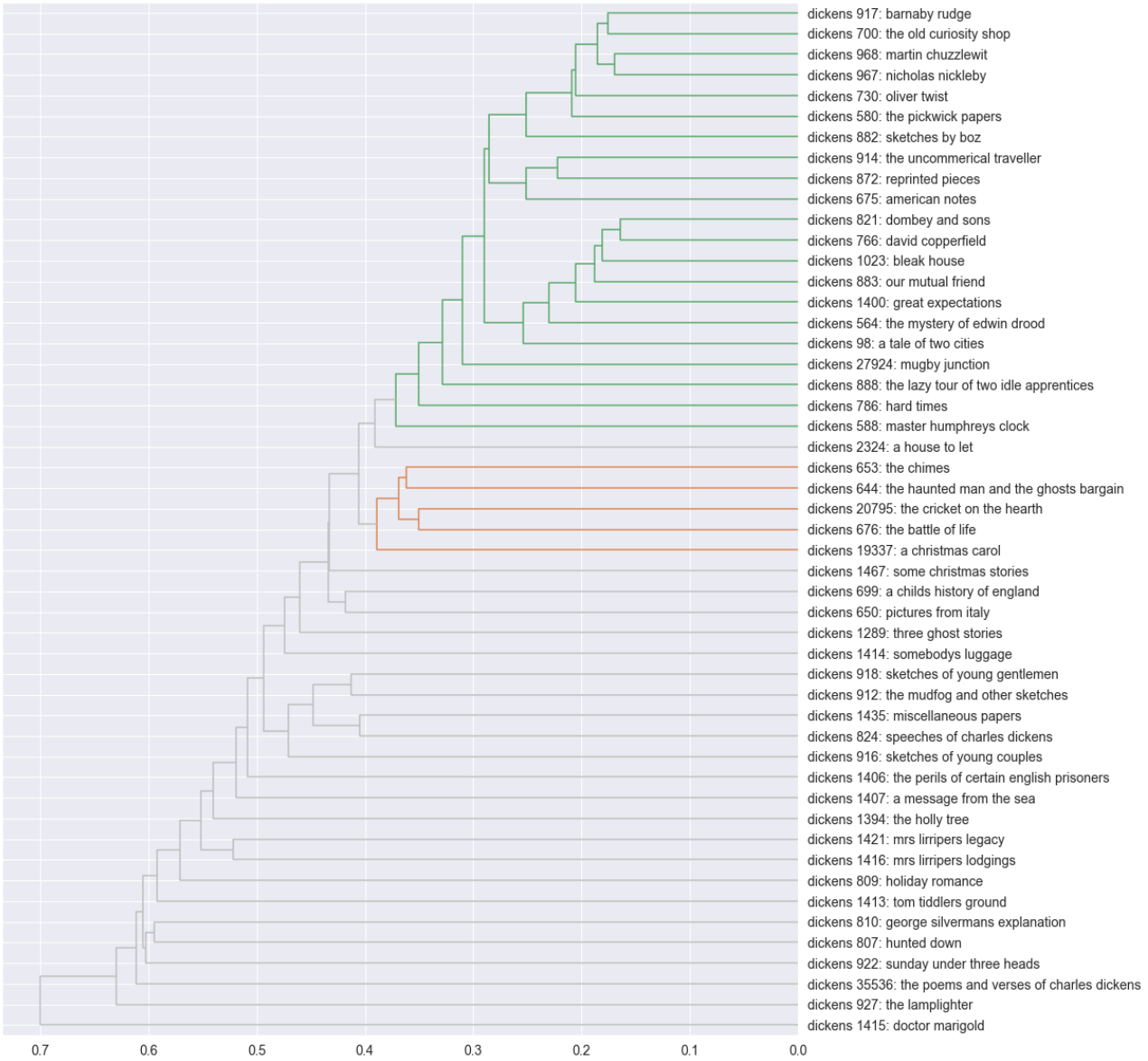

```
<Figure size 432x288 with 0 Axes>
```


```
<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_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
sleep	594	797	5	0.000160	12.606370	NN	9	{'NNP', 'VBD', 'VBG', 'VBP', 'IN', 'VB', 'NN',...	0	
windows	642	747	7	0.000150	12.699841	NNS	12	{'WDT', 'VBD', 'PDT', 'VBP', 'IN', 'RB', 'VBZ'...	0	
top	638	752	3	0.000151	12.690217	NN	8	{'NNP', 'VBD', 'VBP', 'IN', 'VB', 'VBZ', 'NN',...	0	
dress	659	730	5	0.000147	12.733053	NN	8	{'VBP', 'IN', 'RB', 'VB', 'VBZ', 'NN', 'NNS', ...	0	
thank	565	849	5	0.000171	12.515185	NN	7	{'NNP', 'VBD', 'VBP', 'IN', 'VB', 'NN', 'JJ'}	0	
confidence	633	757	10	0.000152	12.680656	NN	5	{'NNP', 'VB', 'NNS', 'NN', 'JJ'}	0	
breast	660	729	6	0.000147	12.735030	NN	9	{'VBD', 'VBN', 'VBP', 'IN', 'RB', 'VBZ', 'VB',...	0	
notice	728	644	6	0.000130	12.913889	NN	9	{'NNP', 'VBN', 'VBP', 'IN', 'RB', 'VB', 'NNS',...	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	st
duty	667	721	4	0.000145	12.750950	NN	9	{'NNP', 'VBN', 'VBP', 'IN', 'VB', 'NNS', 'NN',...	0	
bless	639	751	5	0.000151	12.692136	NN	12	{'NNP', 'VBD', 'PDT', 'VBP', 'IN', 'RB', 'VB',...	0	
number	576	837	6	0.000168	12.535722	NN	8	{'NNP', 'CC', 'VBP', 'IN', 'VB', 'NNS', 'NN', ...	0	
stairs	550	878	6	0.000177	12.466728	NN	10	{'NNP', 'VBD', 'PDT', 'VBP', 'RB', 'VBZ', 'VB'...	0	
breath	701	677	6	0.000136	12.841793	NN	7	{'NNP', 'VBD', 'VBN', 'VB', 'NN', 'NNS', 'JJ'}	0	
order	699	682	5	0.000137	12.831177	NN	12	{'NNP', 'PRP', 'VBG', 'VBP', 'IN', 'RB', 'VB',...	0	
office	472	1021	6	0.000205	12.249038	NN	11	{'NNP', 'PRP', 'VBD', 'VBP', 'IN', 'VBZ', 'VB'...	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	st
ladies	378	1336	6	0.000269	11.861101	NNS	12	{'NNP', 'VBD', 'NNPS', 'VBN', 'VBP', 'RB', 'VB...	0	
paper	530	913	5	0.000184	12.410334	NN	9	{'NNP', 'VBP', 'IN', 'RB', 'VBZ', 'VB', 'NNS',...	0	
knowledge	703	675	9	0.000136	12.846062	NN	9	{'NNP', 'VBD', 'VBP', 'VB', 'VBZ', 'NNS', 'NN'...	0	
instant	742	627	7	0.000126	12.952484	NN	6	{'NNP', 'VBP', 'VBZ', 'VB', 'NN', 'JJ'}	0	
mouth	684	701	5	0.000141	12.791535	NN	7	{'NNP', 'VBD', 'VB', 'VBZ', 'NN', 'NNS', '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)
```

```
['sleep', 'windows', 'top', 'thank', 'dress', 'confidence', 'breast', 'notice',
'duty', 'bless', 'number', 'stairs', 'breath', 'order', 'ladies', 'office', 'pap
er', 'knowledge', 'instant', 'mouth']
```

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

book_id	n	tf	tfidf	source_file_path	title
35536	2.144928	0.037376	0.069868	Dickens/35536-the_poems_and_verses_of_charles_...	the poems and verses of charles dickens
1415	2.952430	0.044903	0.053137	Dickens/1415-doctor_marigold.txt	doctor marigold
810	2.757976	0.044831	0.048352	Dickens/810-george_silvermans_explanation.txt	george silvermans explanation
918	2.391451	0.033139	0.042115	Dickens/918-sketches_of_young_gentlemen.txt	sketches of young gentlemen
916	2.559015	0.031080	0.041771	Dickens/916-sketches_of_young_couples.txt	sketches of young couples
824	2.618026	0.029218	0.039669	Dickens/824-speeches_of_charles_dickens.txt	speeches of charles dickens
807	2.997925	0.039372	0.039216	Dickens/807-hunted_down.txt	hunted down
1435	3.055027	0.018592	0.026247	Dickens/1435-miscellaneous_papers.txt	miscellaneous papers
786	4.259530	0.021844	0.026233	Dickens/786-hard_times.txt	hard times
1400	3.487522	0.024428	0.025954	Dickens/1400-great_expectations.txt	great expectations
809	3.556357	0.017571	0.023731	Dickens/809-holiday_romance.txt	holiday romance
564	3.427017	0.018254	0.023636	Dickens/564-the_mystery_of_edwin_drood.txt	the mystery of edwin drood
1413	3.486200	0.019202	0.023386	Dickens/1413-tom_tiddlers_ground.txt	tom tiddlers ground
917	3.090550	0.020404	0.022530	Dickens/917-barnaby_rudge.txt	barnaby rudge

book_id	n	tf	tfidf	source_file_path	title	
98	3.358461	0.019807	0.021985	Dickens/98-a_tale_of_two_cities.txt	a tale of two cities	
730	3.099458	0.017690	0.021117	Dickens/730-oliver_twist.txt	oliver twist	
883	3.887494	0.017771	0.021075	Dickens/883-our_mutual_friend.txt	our mutual friend	
700	3.094307	0.019364	0.021066	Dickens/700-the_old_curiosity_shop.txt	the old curiosity shop	
1467	3.408788	0.018095	0.020612	Dickens/1467-some_christmas_stories.txt	some christmas stories	
872	3.231072	0.013806	0.019746	Dickens/872-reprinted_pieces.txt	reprinted pieces	$\|TH$ $- L$
967	3.605761	0.016061	0.019325	Dickens/967-nicholas_nickleby.txt	nicholas nickleby	
882	3.025790	0.013125	0.019293	Dickens/882-sketches_by_boz.txt	sketches by boz	
2324	4.272909	0.015482	0.019035	Dickens/2324-a_house_to_let.txt	a house to let	$\|TH$
1394	3.814458	0.016505	0.018806	Dickens/1394-the_holly_tree.txt	the holly tree	
1023	4.023658	0.016917	0.018651	Dickens/1023-bleak_house.txt	bleak house	^\
766	4.164960	0.018244	0.018586	Dickens/766-david_copperfield.txt	david copperfield	
914	3.062025	0.012692	0.018262	Dickens/914-the_uncommerical_traveller.txt	the uncommerical traveller	
1421	4.763350	0.015414	0.017159	Dickens/1421-mrs_lirripers_legacy.txt	mrs lirripers legacy	
1414	3.561426	0.012625	0.017034	Dickens/1414-somebodys_luggage.txt	somebodys luggage	
821	3.849948	0.014126	0.016953	Dickens/821-dombey_and_sons.txt	dombey and sons	

book_id	n	tf	tfidf	source_file_path	title	
699	3.808958	0.012132	0.016704	Dickens/699-a_childs_history_of_england.txt	a childs history of england	
1407	3.729649	0.014084	0.016403	Dickens/1407-a_message_from_the_sea.txt	a message from the sea	
912	3.246902	0.010833	0.016324	Dickens/912-the_mudfog_and_other_sketches.txt	the mudfog and other sketches	PU
968	3.960520	0.014513	0.016247	Dickens/968-martin_chuzzlewit.txt	martin chuzzlewit	
580	3.630811	0.011790	0.015913	Dickens/580-the_pickwick_papers.txt	the pickwick papers	
922	2.866613	0.010350	0.015044	Dickens/922-sunday_under_three_heads.txt	sunday under three heads	
588	3.900265	0.011998	0.014644	Dickens/588-master_humphreys_clock.txt	master humphreys clock	'
19337	3.903588	0.012571	0.014323	Dickens/19337-a_christmas_carol.txt	a christmas carol	
1416	5.168994	0.012984	0.014209	Dickens/1416-mrs_lirripers_lodgings.txt	mrs lirripers lodgings	
653	4.204897	0.011646	0.013845	Dickens/653-the_chimes.txt	the chimes	
27924	4.389395	0.012060	0.013294	Dickens/27924-mugby_junction.txt	mugby junction	$\|BA$
927	4.405577	0.011503	0.013199	Dickens/927-the_lamplighter.txt	the lamplighter	
675	3.382945	0.009514	0.012563	Dickens/675-american_notes.txt	american notes	
676	4.515977	0.010164	0.012512	Dickens/676-the_battle_of_life.txt	the battle of life	
650	3.629300	0.007702	0.011398	Dickens/650-pictures_from_italy.txt	pictures from italy	

book_id	n	tf	tfidf	source_file_path	title	
1406	5.571753	0.009851	0.010948	Dickens/1406-the_perils_of_certain_english_pri...	the perils of certain english prisoners	
20795	4.899202	0.009398	0.010577	Dickens/20795-the_cricket_on_the_hearth.txt	the cricket on the hearth	
1289	4.083012	0.009729	0.010370	Dickens/1289-three_ghost_stories.txt	three ghost stories	TH
644	4.975638	0.008870	0.009827	Dickens/644-the_haunted_man_and_the_ghosts_bar...	the haunted man and the ghosts bargain	
888	4.098543	0.007341	0.009703	Dickens/888-the_lazy_tour_of_two_idle_apprenti...	the lazy tour of two idle apprentices	

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	cityb
564.0	**98.0**	dickens 564: the mystery of edwin drood	dickens 98: a tale of two cities	0.817650	2.181992	0.102131	0.705128	0.248603	0.5

doc_a	doc_b	label_a	label_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityb
580.0	**98.0**	dickens 580: the pickwick papers	dickens 98: a tale of two cities	1.002878	2.289080	0.110589	0.812632	0.330185	0.6
588.0	**98.0**	dickens 588: master humphreys clock	dickens 98: a tale of two cities	0.946882	2.587732	0.125556	0.770693	0.296984	0.6
644.0	**98.0**	dickens 644: the haunted man and the ghosts bargain	dickens 98: a tale of two cities	0.678359	2.233104	0.105889	0.811974	0.329651	0.6
650.0	**98.0**	dickens 650: pictures from italy	dickens 98: a tale of two cities	0.871224	2.152545	0.108974	0.795297	0.316249	0.6
653.0	**98.0**	dickens 653: the chimes	dickens 98: a tale of two cities	0.723809	3.706198	0.211257	1.039313	0.540086	1.0
675.0	**98.0**	dickens 675: american notes	dickens 98: a tale of two cities	0.955682	2.137805	0.098926	0.789208	0.311424	0.6
676.0	**98.0**	dickens 676: the battle of life	dickens 98: a tale of two cities	0.820688	4.551075	0.340992	1.198773	0.718528	1.2
699.0	**98.0**	dickens 699: a childs history of england	dickens 98: a tale of two cities	0.769068	3.010372	0.147643	0.945397	0.446888	0.9
700.0	**98.0**	dickens 700: the old curiosity shop	dickens 98: a tale of two cities	0.902024	2.798888	0.154623	0.932323	0.434613	0.7
730.0	**98.0**	dickens 730: oliver twist	dickens 98: a tale of two cities	0.968183	3.252808	0.153514	0.819351	0.335668	0.7

doc_a	doc_b	label_a	label_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityb
766.0	**98.0**	dickens 766: david copperfield	dickens 98: a tale of two cities	0.873235	2.471151	0.110196	0.762618	0.290793	0.6
786.0	**98.0**	dickens 786: hard times	dickens 98: a tale of two cities	0.726354	2.067851	0.095040	0.624151	0.194782	0.4
807.0	**98.0**	dickens 807: hunted down	dickens 98: a tale of two cities	0.886397	3.223907	0.144618	0.872691	0.380794	0.8
809.0	**98.0**	dickens 809: holiday romance	dickens 98: a tale of two cities	1.017582	1.588135	0.085116	0.607068	0.184266	0.4
810.0	**98.0**	dickens 810: george silvermans explanation	dickens 98: a tale of two cities	0.867269	1.997594	0.095671	0.666314	0.221987	0.4
821.0	**98.0**	dickens 821: dombey and sons	dickens 98: a tale of two cities	0.754086	4.478313	0.333909	1.141066	0.651016	1.1
824.0	**98.0**	dickens 824: speeches of charles dickens	dickens 98: a tale of two cities	1.046565	3.688435	0.195294	0.969963	0.470414	0.8
872.0	**98.0**	dickens 872: reprinted pieces	dickens 98: a tale of two cities	0.898753	2.890082	0.156784	0.846636	0.358396	0.6
882.0	**98.0**	dickens 882: sketches by boz	dickens 98: a tale of two cities	0.965408	3.453197	0.165532	1.049204	0.550414	1.0

Compare Z normalized distributions

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

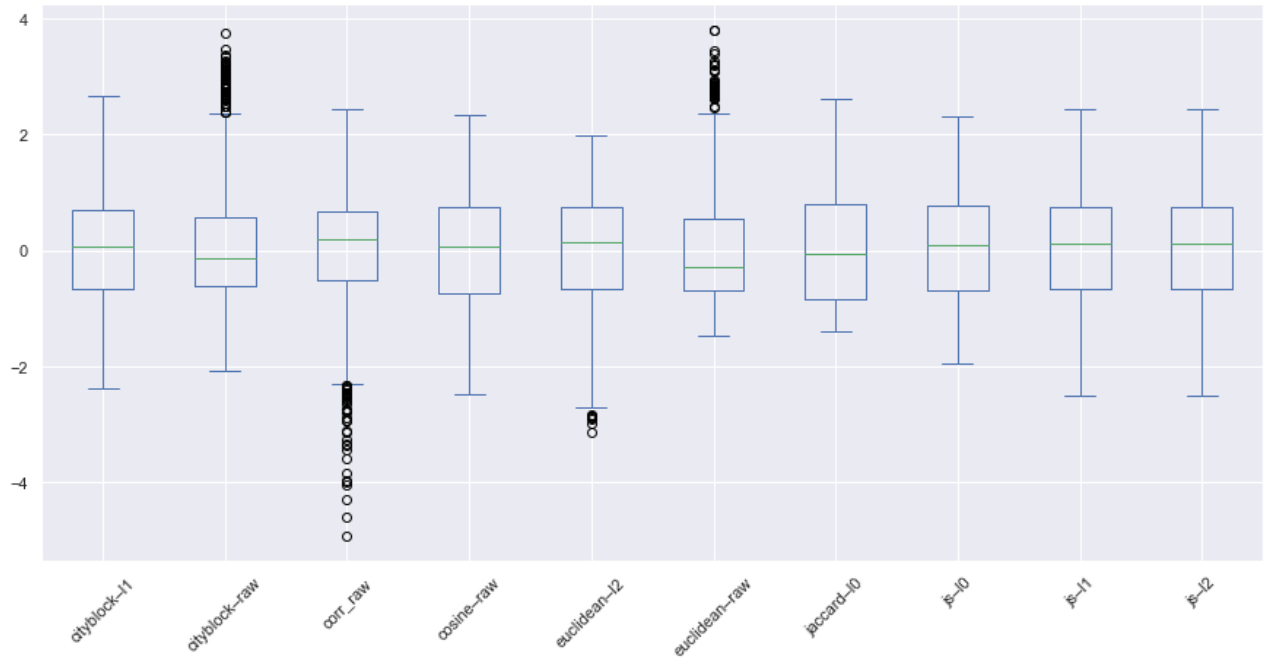
In [78]:
```
ZPAIRS
```

Out[78]:

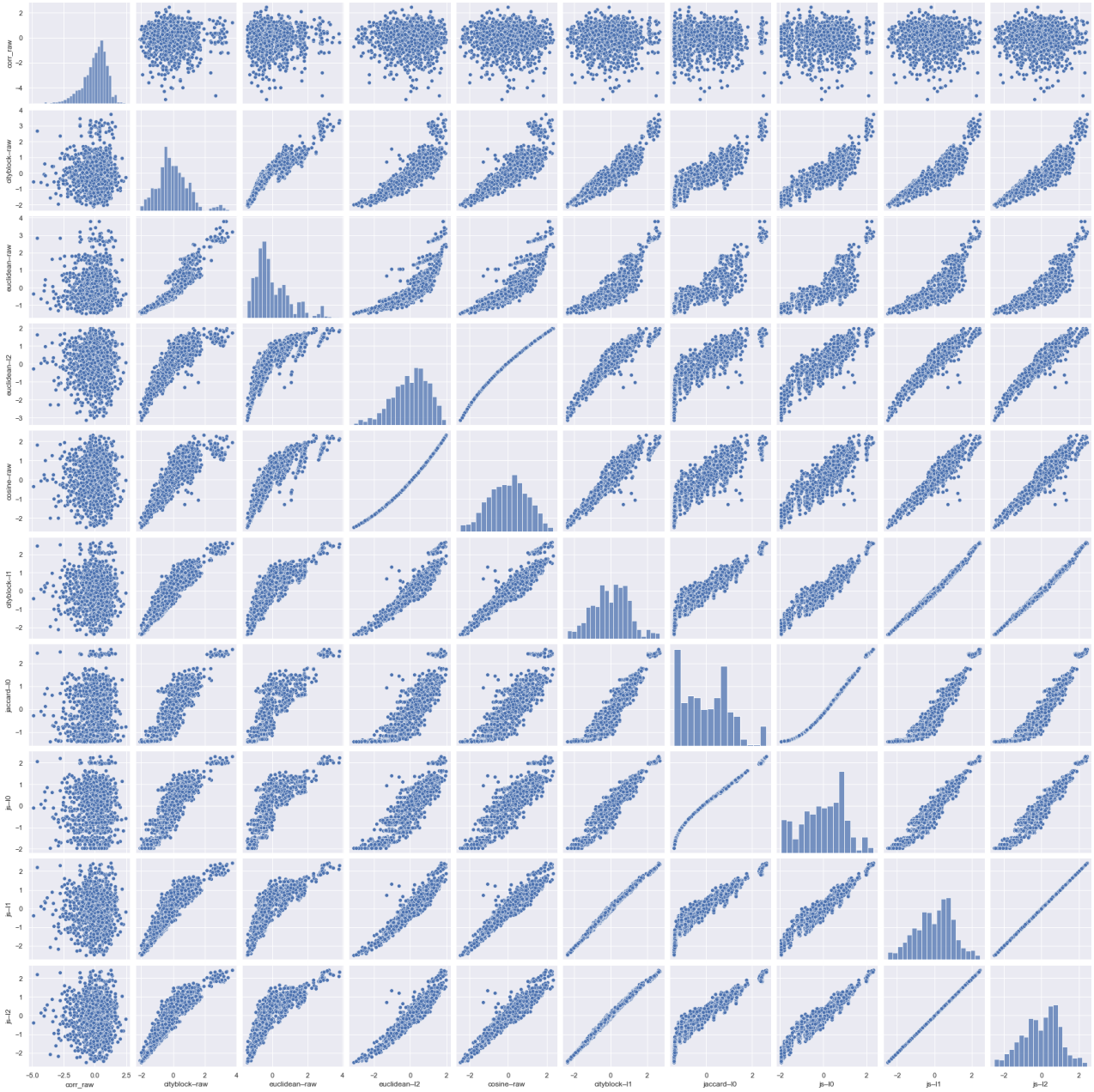
doc_a	doc_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0
564	**98**	-0.793377	-1.379250	-1.117107	-1.680734	-1.584085	-1.578464	-1.319745
580	**98**	0.814111	-1.272129	-1.018769	-1.038081	-1.070754	-1.220194	-1.375355
	564	-0.277098	-0.470971	-0.397164	-0.475308	-0.561506	-0.769479	-1.167842
588	**98**	0.328151	-0.973385	-0.844751	-1.288790	-1.279663	-1.069836	-0.782614
	564	-1.057759	-0.971761	-0.961709	-0.645112	-0.721034	-0.523605	-1.131869
...	**...**	...	...	...	...	...	...	...
35536	**1467**	-0.657405	-0.504204	-0.349307	-0.039757	-0.129099	-0.546136	-0.569091
	2324	0.130957	0.506804	0.090930	0.862431	0.872827	1.082217	0.934007
	19337	0.288699	-0.386278	-0.352837	-0.288472	-0.380110	-0.681064	-0.782686
	20795	0.714052	0.752296	0.177578	0.899015	0.916479	1.025788	0.973376
	27924	0.497169	0.272635	-0.168241	0.616982	0.586053	0.840609	0.850490

1225 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
1023	bleak house	dickens 1023: bleak house	3	0	0	3
1289	three ghost stories	dickens 1289: three ghost stories	3	0	0	0
1394	the holly tree	dickens 1394: the holly tree	3	2	0	3
1400	great expectations	dickens 1400: great expectations	3	0	0	3
1406	the perils of certain english prisoners	dickens 1406: the perils of certain english prisoners	3	2	0	3
1407	a message from the sea	dickens 1407: a message from the sea	3	2	0	0
1413	tom tiddlers ground	dickens 1413: tom tiddlers ground	3	2	0	3
1414	somebodys luggage	dickens 1414: somebodys luggage	3	2	0	0
1415	doctor marigold	dickens 1415: doctor marigold	0	3	2	1
1416	mrs lirripers lodgings	dickens 1416: mrs lirripers lodgings	1	2	3	2
1421	mrs lirripers legacy	dickens 1421: mrs lirripers legacy	1	2	3	2
1435	miscellaneous papers	dickens 1435: miscellaneous papers	3	0	0	0
1467	some christmas stories	dickens 1467: some christmas stories	3	0	0	0
19337	a christmas carol	dickens 19337: a christmas carol	3	0	0	3
20795	the cricket on the hearth	dickens 20795: the cricket on the hearth	3	0	0	3
2324	a house to let	dickens 2324: a house to let	3	0	0	3
27924	mugby junction	dickens 27924: mugby junction	3	0	0	0
35536	the poems and verses of charles dickens	dickens 35536: the poems and verses of charles dickens	3	1	0	0
564	the mystery of edwin drood	dickens 564: the mystery of edwin drood	3	0	0	3

book_id	title	label	y_raw	y_L0	y_L1	y_L2
580	the pickwick papers	dickens 580: the pickwick papers	3	0	0	3
588	master humphreys clock	dickens 588: master humphreys clock	3	0	0	3
644	the haunted man and the ghosts bargain	dickens 644: the haunted man and the ghosts bargain	3	0	0	3
650	pictures from italy	dickens 650: pictures from italy	3	0	0	0
653	the chimes	dickens 653: the chimes	3	0	0	3
675	american notes	dickens 675: american notes	3	0	0	0
676	the battle of life	dickens 676: the battle of life	3	0	0	3
699	a childs history of england	dickens 699: a childs history of england	3	0	0	0
700	the old curiosity shop	dickens 700: the old curiosity shop	3	0	0	3
730	oliver twist	dickens 730: oliver twist	3	0	0	3
766	david copperfield	dickens 766: david copperfield	3	0	0	3
786	hard times	dickens 786: hard times	3	0	0	3
807	hunted down	dickens 807: hunted down	3	2	0	3
809	holiday romance	dickens 809: holiday romance	3	2	1	0
810	george silvermans explanation	dickens 810: george silvermans explanation	3	2	0	3
821	dombey and sons	dickens 821: dombey and sons	3	0	0	3
824	speeches of charles dickens	dickens 824: speeches of charles dickens	3	0	0	0
872	reprinted pieces	dickens 872: reprinted pieces	3	0	0	0
882	sketches by boz	dickens 882: sketches by boz	3	0	0	0
883	our mutual friend	dickens 883: our mutual friend	3	0	0	3
888	the lazy tour of two idle apprentices	dickens 888: the lazy tour of two idle apprentices	3	0	0	0
912	the mudfog and other sketches	dickens 912: the mudfog and other sketches	3	0	0	0
914	the uncommerical traveller	dickens 914: the uncommerical traveller	3	0	0	0
916	sketches of young couples	dickens 916: sketches of young couples	2	0	0	3
917	barnaby rudge	dickens 917: barnaby rudge	3	0	0	3
918	sketches of young gentlemen	dickens 918: sketches of young gentlemen	3	0	0	3
922	sunday under three heads	dickens 922: sunday under three heads	3	2	0	0
927	the lamplighter	dickens 927: the lamplighter	1	3	3	2

book_id	title	label	y_raw	y_L0	y_L1	y_L2
967	nicholas nickleby	dickens 967: nicholas nickleby	3	0	0	3
968	martin chuzzlewit	dickens 968: martin chuzzlewit	3	0	0	3
98	a tale of two cities	dickens 98: a tale of two cities	3	0	0	3

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	0.434304	0.375481	0.416088	0.153934
1	3.000000	0.447782	0.350379	0.424456	0.062581
2	4.000000	0.316780	0.355775	0.405744	0.063818
3	5.000000	0.313854	0.356422	0.046932	0.076669
4	6.000000	0.045889	0.352762	0.247122	0.070491
5	7.000000	0.049817	0.355153	0.252137	0.068152
6	8.000000	0.005861	0.317736	0.066262	0.046934
7	9.000000	0.048023	0.342503	0.253976	0.052131
8	10.000000	0.065307	0.298556	0.082754	0.066405

In [86]:
```
# 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 [87]:
```
max_score_vec
```

Out[87]:
```
'raw_silhouette_score'
```

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_raw
927	dickens 927: the lamplighter	stories	0
1421	dickens 1421: mrs lirripers legacy	stories	0
1416	dickens 1416: mrs lirripers lodgings	stories	0
98	dickens 98: a tale of two cities	novel	1
917	dickens 917: barnaby rudge	stories	1
918	dickens 918: sketches of young gentlemen	stories	1
922	dickens 922: sunday under three heads	non-fiction	1
967	dickens 967: nicholas nickleby	novel	1
968	dickens 968: martin chuzzlewit	novel	1
1023	dickens 1023: bleak house	novel	1
1289	dickens 1289: three ghost stories	stories	1
1394	dickens 1394: the holly tree	stories	1
1406	dickens 1406: the perils of certain english prisoners	stories	1
916	dickens 916: sketches of young couples	stories	1
1407	dickens 1407: a message from the sea	stories	1
1413	dickens 1413: tom tiddlers ground	stories	1
1414	dickens 1414: somebodys luggage	stories	1

book_id	label	type	max_y_raw
1435	dickens 1435: miscellaneous papers	non-fiction	1
1467	dickens 1467: some christmas stories	stories	1
2324	dickens 2324: a house to let	stories	1
19337	dickens 19337: a christmas carol	novel	1
20795	dickens 20795: the cricket on the hearth	novel	1
1400	dickens 1400: great expectations	novel	1
27924	dickens 27924: mugby junction	stories	1
914	dickens 914: the uncommerical traveller	non-fiction	1
888	dickens 888: the lazy tour of two idle apprentices	stories	1
564	dickens 564: the mystery of edwin drood	novel	1
580	dickens 580: the pickwick papers	novel	1
588	dickens 588: master humphreys clock	stories	1
644	dickens 644: the haunted man and the ghosts bargain	stories	1
650	dickens 650: pictures from italy	non-fiction	1
653	dickens 653: the chimes	novel	1
675	dickens 675: american notes	non-fiction	1
676	dickens 676: the battle of life	novel	1
699	dickens 699: a childs history of england	non-fiction	1
912	dickens 912: the mudfog and other sketches	stories	1
700	dickens 700: the old curiosity shop	novel	1
766	dickens 766: david copperfield	novel	1
786	dickens 786: hard times	novel	1
807	dickens 807: hunted down	stories	1
809	dickens 809: holiday romance	stories	1
810	dickens 810: george silvermans explanation	stories	1
821	dickens 821: dombey and sons	novel	1
824	dickens 824: speeches of charles dickens	non-fiction	1
872	dickens 872: reprinted pieces	stories	1
882	dickens 882: sketches by boz	stories	1
883	dickens 883: our mutual friend	novel	1
730	dickens 730: oliver twist	novel	1
35536	dickens 35536: the poems and verses of charles dickens	stories	1
1415	dickens 1415: doctor marigold	stories	2

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

Out[91]:
```
max_y_raw
0     3
1    46
2     1
dtype: int64
```

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

Out[92]:
```
type         max_y_raw
non-fiction  1             7
novel        1            17
stories      0             3
             1            22
             2             1
dtype: int64
```

M07: Features and Components

In [93]:
```
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.

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

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

In [94]:
```
chap_DOC
```

Out[94]:

book_id	chap_id	author	title	type	decade	label	mean_tfidf	n_tokens
98	1	dickens	a tale of two cities	novel	1850	98-dickens--a tale of two cities-1	0.000363	1017
	2	dickens	a tale of two cities	novel	1850	98-dickens--a tale of two cities-2	0.000233	2044
	3	dickens	a tale of two cities	novel	1850	98-dickens--a tale of two cities-3	0.000269	1638
	4	dickens	a tale of two cities	novel	1850	98-dickens--a tale of two cities-4	0.000422	4439
	5	dickens	a tale of two cities	novel	1850	98-dickens--a tale of two cities-5	0.000330	4218
...	...	...	...	...	...	...	...	...

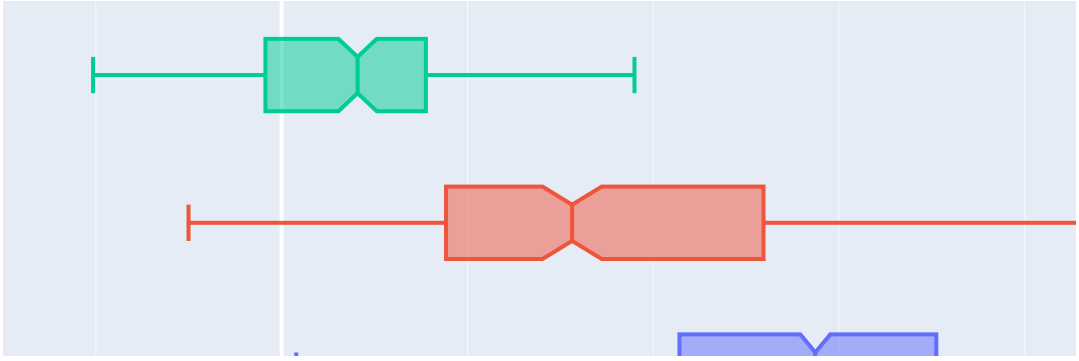
book_id	chap_id	author	title	type	decade	label	mean_tfidf	n_tokens
35536	9	dickens	the poems and verses of charles dickens	stories	1880	35536-dickens--the poems and verses of charles...	0.000507	942
	10	dickens	the poems and verses of charles dickens	stories	1880	35536-dickens--the poems and verses of charles...	0.000797	356
	11	dickens	the poems and verses of charles dickens	stories	1880	35536-dickens--the poems and verses of charles...	0.000327	685
	12	dickens	the poems and verses of charles dickens	stories	1880	35536-dickens--the poems and verses of charles...	0.000338	547
	13	dickens	the poems and verses of charles dickens	stories	1880	35536-dickens--the poems and verses of charles...	0.000671	439

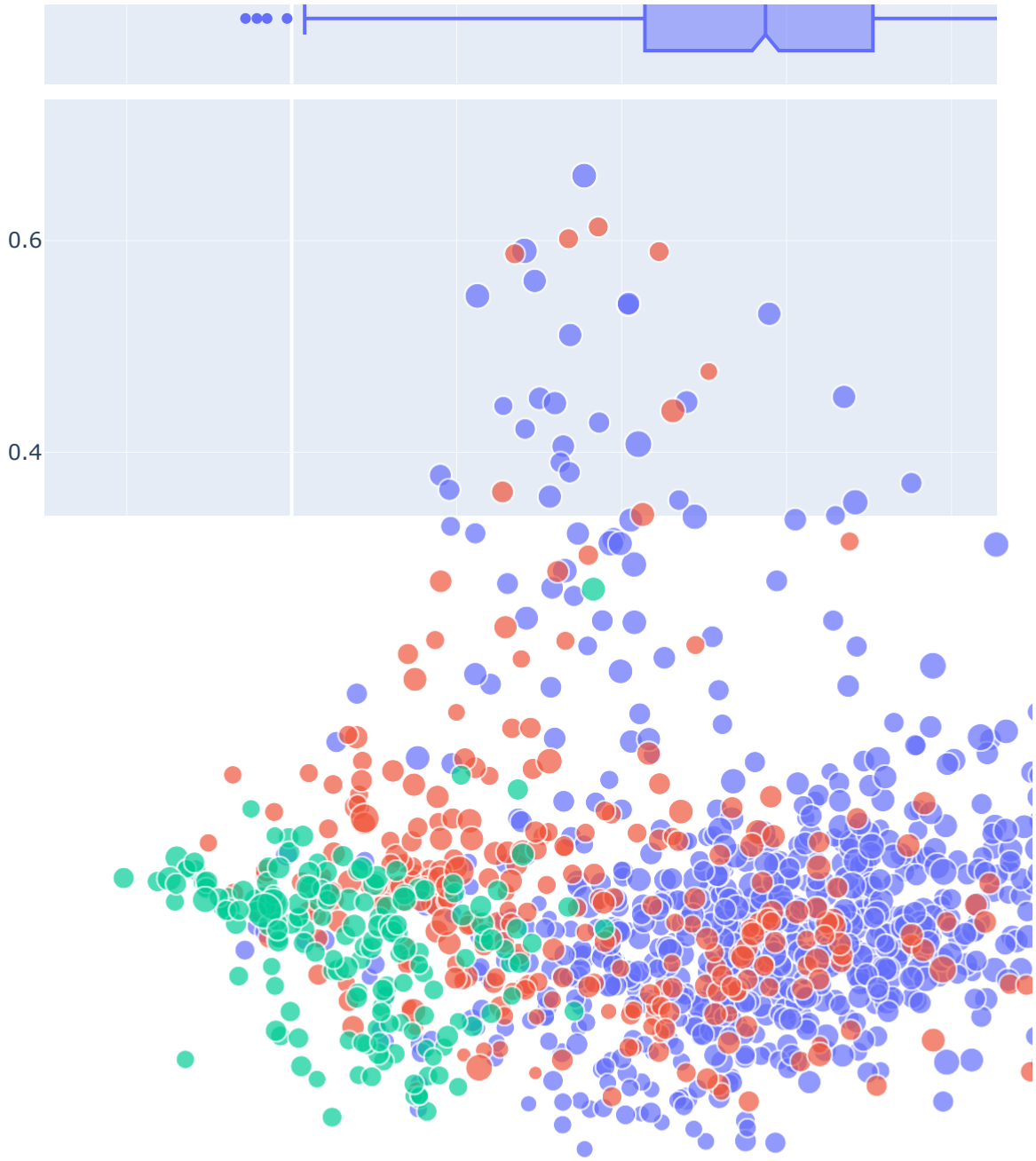
1182 rows × 7 columns

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

In [95]:
```
LOADINGS, DCM, COMPINF = get_pca(TFIDF_sigs, norm_docs = True, center_by_mean =
```

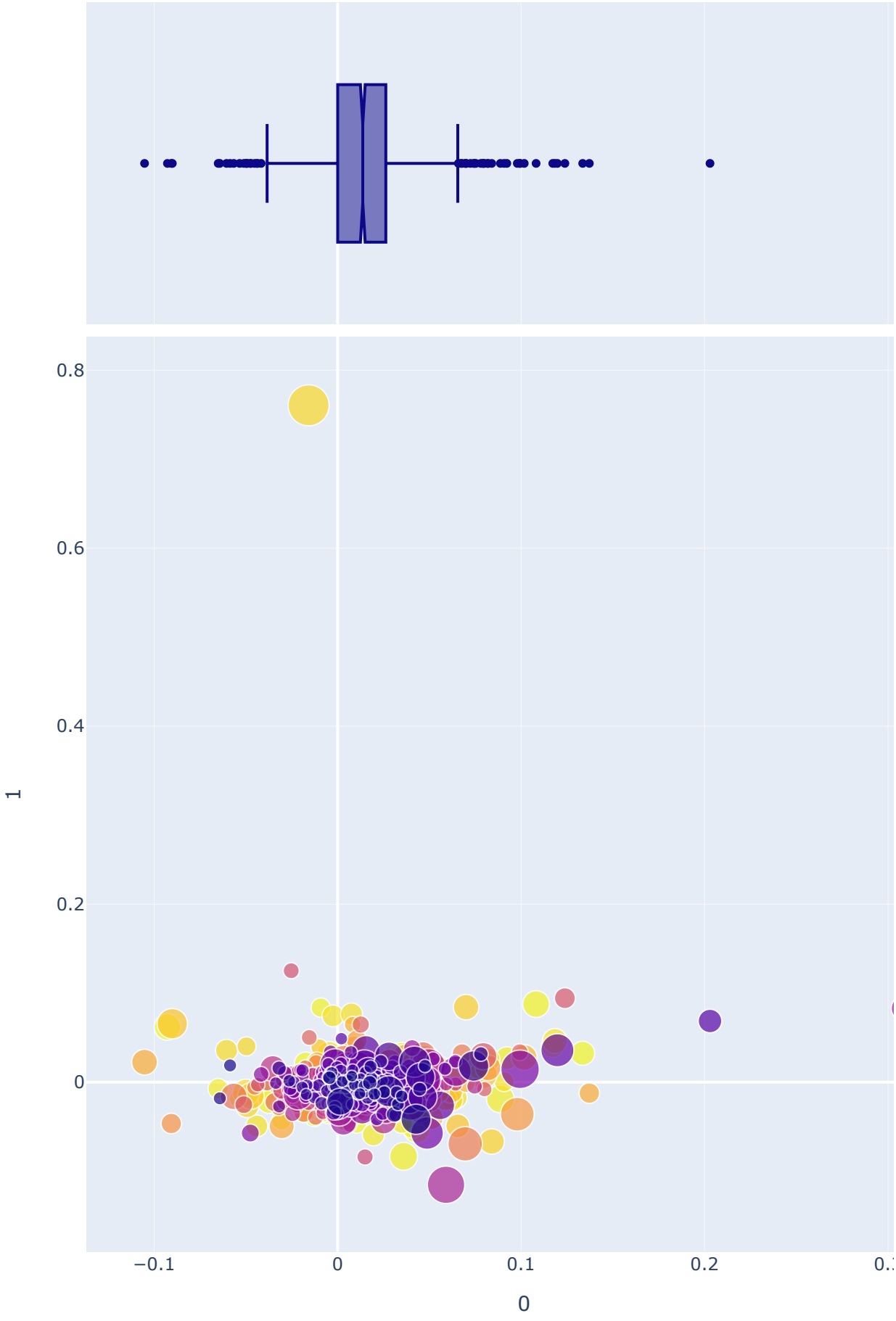
In [96]:
```
px.scatter(DCM, 0, 1, color=chap_DOC.type,
           size=np.abs(chap_DOC.mean_tfidf), hover_name=chap_DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```





In [97]:
```
X = LOADINGS.join(SIGS, how='inner').reset_index()
```

In [98]:
```
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)
```


0.8
0.6
0.4
0.2
0
1
−0.1
0
0.1
0.2
0

In [99]:
```
COMPINF
```

Out[99]:

pc_id	pos	neg	eig_val	exp_var
0	ha maam rejoined inquired aint	public ladies sea gentlemen houses	0.017467	0.156175
1	says takes aint em o	child loved brother father sister	0.016741	0.149683
2	maam sister ladies brother marriage	ha horses crowd road river	0.014116	0.126209
3	ha ladies gentlemen inquired coach	says child father mother loved	0.013638	0.121941
4	coach maam ladies shop town	ha eh brother o rejoined	0.010121	0.090495
5	maam mother child boys boy	ha gentlemen brother court office	0.009784	0.087477
6	brother son maam court father	child ladies love happy girl	0.008110	0.072511
7	sister boy aint o boys	ha maam gentlemen ladies says	0.007541	0.067423
8	ladies boys father boy son	maam letter office river reference	0.007492	0.066984
9	ha sea sister school children	crowd girl gentlemen rejoined inquired	0.006834	0.061103

Prince PCA Method with entire TFIDF

In [139…
```
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 [140…
```
pca = pca.fit(TFIDF)
```

In [141…
```
dcm = pca.transform(TFIDF)
```

In [142…
```
dcm
```

Out[142…

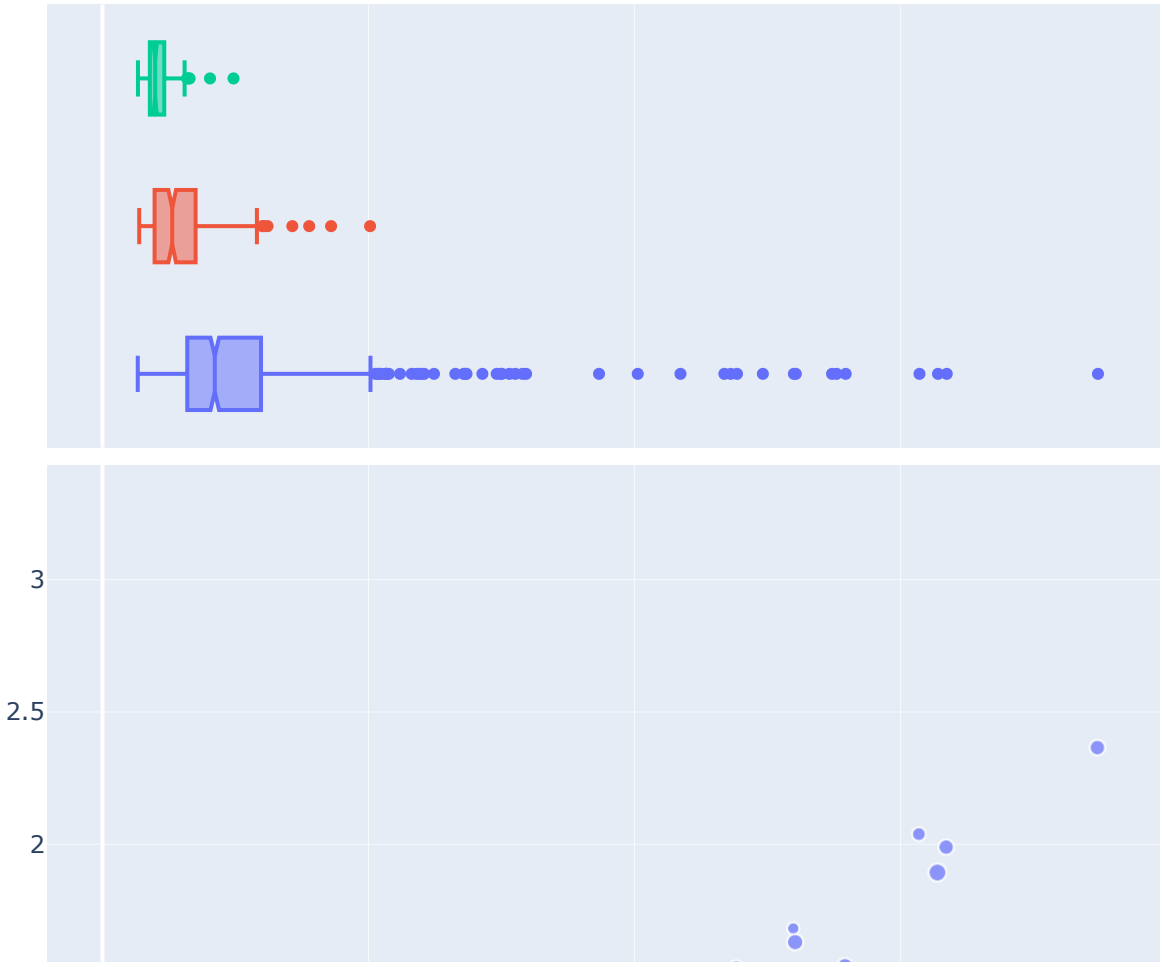
book_id	chap_id	0	1	2	3	4	5
98	1	0.079602	-0.047823	-0.032369	0.022100	-0.028963	-0.002443
	2	0.080926	-0.050889	-0.033654	0.017720	-0.035551	0.008480
	3	0.077527	-0.043145	-0.028152	0.018199	-0.025712	0.004466

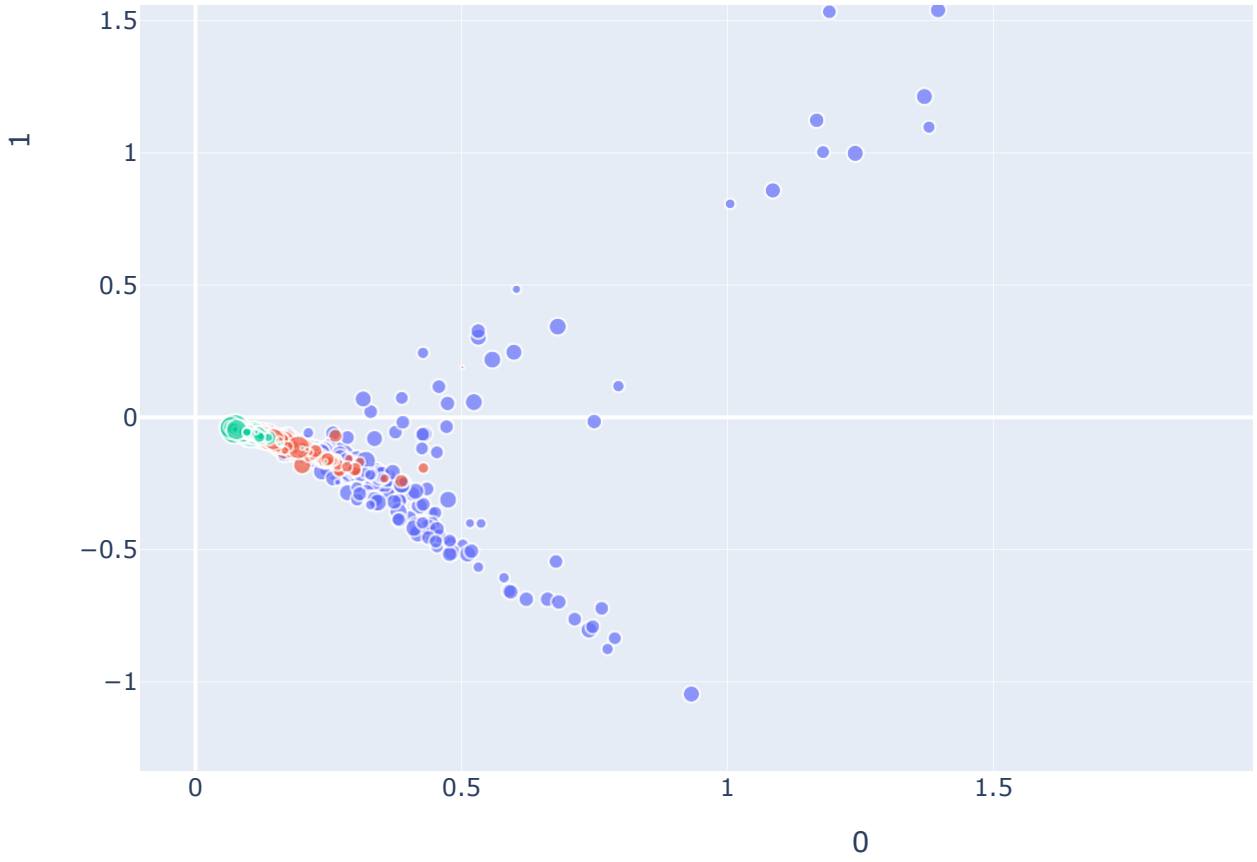
book_id	chap_id	0	1	2	3	4	5
	4	0.176733	-0.106757	-0.057149	0.045436	-0.043284	0.005294
	5	0.108728	-0.070882	-0.018355	0.023081	-0.020364	0.021739
...	...	...	...	...	...	...	...
35536	9	0.125069	-0.072843	-0.055778	0.018766	-0.043503	0.008264
	10	0.503005	0.189780	-0.045538	-0.070940	-0.269298	0.248859
	11	0.082916	-0.048396	-0.036694	0.014311	-0.034964	-0.007043
	12	0.079417	-0.044509	-0.030858	0.015558	-0.033919	0.009589
	13	0.211297	-0.120470	-0.095485	0.067477	-0.114600	0.044557

1182 rows × 6 columns

In [143…

```
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')
```





Remove Outliers

In [105…

```
# 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 [106…

```
# upper fences for pc 0
dickens_0_upper_fence = upper_fence(dcm, LIB.index.values, 0)

# upper fences for pc 1
dickens_1_upper_fence = upper_fence(dcm, LIB.index.values, 1)
```

In [107…

```
# outliers the chapters in books with PC 0 or PC 1 greater than the max of the u
outliers = dcm.loc[(dcm[0] > dickens_0_upper_fence) | (dcm[1] > dickens_1_upper_
```

In [108…

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

In [117…

```
# 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)]

# remove numbers
small_VOCAB = small_VOCAB.loc[~small_VOCAB.index.str.contains('[0-9]', regex = T
```

In [118…

```
# remove ~15% of VOCAB data

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

Out[118…

```
0.15425531914893617
```

In [119…

```
# remove proper nouns and numbers from corpus
small_CORPUS = small_CORPUS.loc[small_CORPUS.term_str.isin(small_VOCAB.index.val
```

In [120…

```
# remove ~11% of data

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

Out[120…

```
0.10609553711053038
```

In [121…

```
small_BOW = create_bow(small_CORPUS, CHAPS)
```

In [122…

```
# suppress chained assignment warning
pd.options.mode.chained_assignment = None
```

In [123…

```
small_DTCM, small_TFIDF, small_BOW, small_DFIDF, small_VOCAB = get_tfidf(small_B
```

In [124…

```
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 [125…

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

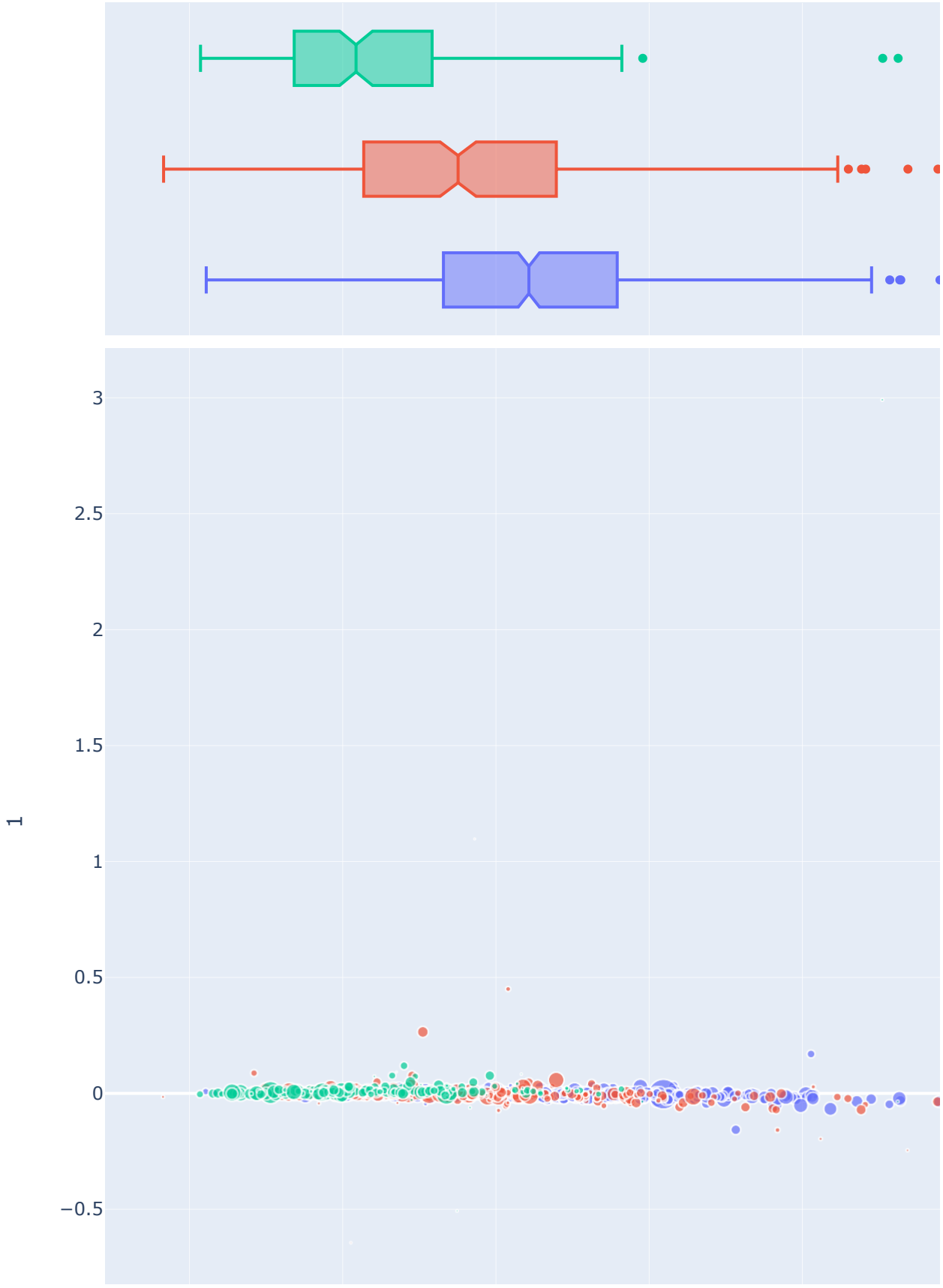
In [126…

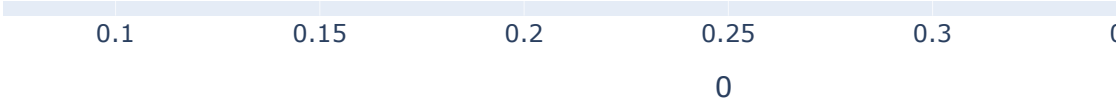
```
small_dcm = pca.transform(small_TFIDF)
```

In [127…

```
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 1000 most significant terms excluding proper nouns (`TFIDF_sigs`)

In [128…
```
pca_sigs = pca.fit(TFIDF_sigs)
```

In [129…
```
dcm_sigs = pca_sigs.transform(TFIDF_sigs)
```

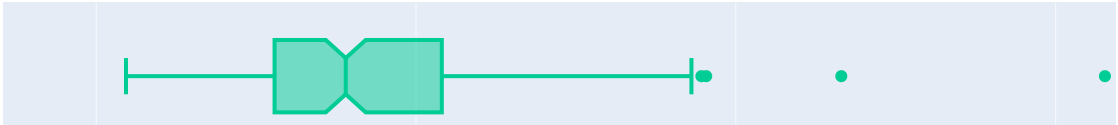
In [130…
```
dcm_sigs
```

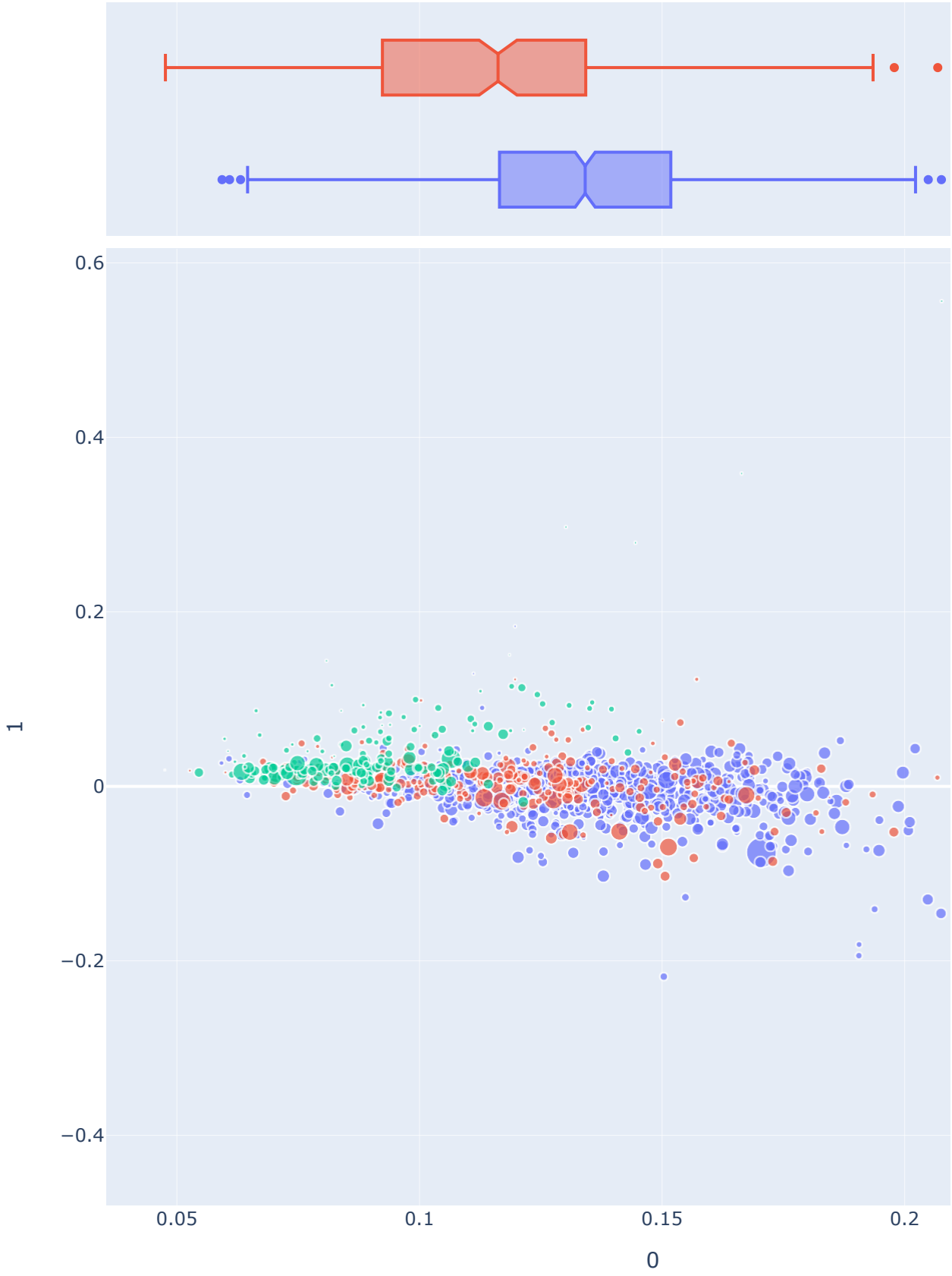
Out[130…

book_id	chap_id	0	1	2	3	4	5
98	1	0.059308	0.027044	-0.004335	0.002380	-0.007692	-0.010751
	2	0.064570	-0.009571	-0.008245	-0.023042	0.019243	0.009329
	3	0.083818	0.014368	-0.013850	-0.018862	-0.002281	0.017569
	4	0.144831	0.012586	-0.011949	-0.017702	-0.018918	0.001366
	5	0.099830	-0.004260	-0.004859	-0.025189	-0.001643	0.008036
...	...	...	...	...	...	...	...
35536	9	0.088392	0.011102	-0.007005	-0.002917	-0.019529	-0.001646
	10	0.184596	-0.012318	0.001723	-0.034009	0.035516	-0.121150
	11	0.072715	0.016359	-0.004793	-0.000527	-0.023610	0.017840
	12	0.072537	0.022408	-0.005806	0.002997	-0.015566	-0.019207
	13	0.107308	0.026861	-0.012891	-0.000272	-0.013886	-0.025155

1182 rows × 6 columns

In [131…
```
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 [132…

```
# save BOW for topic modeling

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

In [ ]: