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

DS 5001: Exploratory Text Analytics

Cecily Wolfe (cew4pf)

Spring 2022

In [219…

```
# 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]:
```
# read in csv files

dickens_LIB = pd.read_csv('dickens_pre_LIB.csv')

dickens_CORPUS = pd.read_csv('dickens_pre_CORPUS.csv')

twain_LIB = pd.read_csv('twain_pre_LIB.csv')

twain_CORPUS = pd.read_csv('twain_pre_CORPUS.csv')
```

LIB Table

In [6]:
```
# combined LIB
LIB = pd.concat([dickens_LIB, twain_LIB]).set_index(BOOKS).sort_index()
```

In [7]:
```
LIB.head()
```

Out[7]:

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

CORPUS Table

In [8]:
```
# combined corpus

CORPUS = pd.concat([dickens_CORPUS, twain_CORPUS]).set_index(OHCO)
```

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

In [9]:

```
CORPUS
```

Out[9]:

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
98	1	0	0	0	('The', 'DT')	DT	The	the
				1	('Period', 'NN')	NN	Period	period
		1	0	0	('It', 'PRP')	PRP	It	it
				1	('was', 'VBD')	VBD	was	was
				2	('the', 'DT')	DT	the	the
...	...	...	...	...	...	...	...	...
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

7940320 rows × 4 columns

In [10]:

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

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

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

In [11]:

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

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

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

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

    # number of characters in each term
```

```
    vocab['n_chars'] = vocab.index.str.len()

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

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

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

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

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

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

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

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

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

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

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

    return vocab
```

In [12]:
```
VOCAB = create_vocab(CORPUS)
```

M03: Language Models

Create Training Vocab (V_{train})

In [13]:
```
V_TRAIN = sorted(list(set(VOCAB.index)))
```

In [14]:
```
len(V_TRAIN)
```

Out[14]:
```
80292
```

Generate Training Sentences

In [15]:
```
# 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 [16]:
```
S_TRAIN = list(CORPUS.groupby(OHCO[:-1]).term_str.apply(lambda x: ' '.join(x)).v
```

In [17]:
```
len(S_TRAIN)
```

Out[17]:
```
393987
```

Generate and Count Ngrams

In [19]:
```
train = NgramCounter(S_TRAIN, V_TRAIN)
train.generate()
```

n-gram token table

In [20]:
```
train.I
```

Out[20]:

		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>
...	...	...	...	...
393986	11	soliloquy	by	mark
	12	by	mark	twain
	13	mark	twain	</s>
	14	twain	</s>	NaN
	15	</s>	NaN	NaN

9122281 rows × 3 columns

Unigram table

In [418…
```
stop_words = VOCAB.loc[VOCAB.stop == 1].index.values
```

In [419…
```
unigram_df = train.LM[0].sort_values('n', ascending = False)
```

```
unigram_df
```

Out[419…

w0	n	mle	p	log_p
<s>	787974	8.637905e-02	8.637905e-02	-3.533175
the	418963	4.592744e-02	4.592744e-02	-4.444500
</s>	393987	4.318953e-02	4.318953e-02	-4.533175
and	310105	3.399424e-02	3.399424e-02	-4.878566
of	218996	2.400671e-02	2.400671e-02	-5.380418
…	…	…	…	…
jewry	1	1.096217e-07	1.096217e-07	-23.120963
jewsand	1	1.096217e-07	1.096217e-07	-23.120963
jewsharp	1	1.096217e-07	1.096217e-07	-23.120963
jezabels	1	1.096217e-07	1.096217e-07	-23.120963
ĕngine	1	1.096217e-07	1.096217e-07	-23.120963

80294 rows × 4 columns

In [421…

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

Out[421…

w0	n	mle	p	log_p
the	418963	4.592744e-02	4.592744e-02	-4.444500
and	310105	3.399424e-02	3.399424e-02	-4.878566
of	218996	2.400671e-02	2.400671e-02	-5.380418
to	206700	2.265881e-02	2.265881e-02	-5.463784
a	189310	2.075249e-02	2.075249e-02	-5.590572
…	…	…	…	…
jewry	1	1.096217e-07	1.096217e-07	-23.120963
jewsand	1	1.096217e-07	1.096217e-07	-23.120963
jewsharp	1	1.096217e-07	1.096217e-07	-23.120963
jezabels	1	1.096217e-07	1.096217e-07	-23.120963
ĕngine	1	1.096217e-07	1.096217e-07	-23.120963

80292 rows × 4 columns

Bigram table

In [22]:

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

```
bigram_df
```

Out[22]:

w0	w1	n	mle
<s>	<s>	393987	4.318953e-02
</s>	<s>	393986	4.318942e-02
of	the	46463	5.093354e-03
<s>	i	38136	4.180534e-03
in	the	36914	4.046576e-03
...	...	...	...
hindered	not	1	1.096217e-07
	it	1	1.096217e-07
	him	1	1.096217e-07
	by	1	1.096217e-07
ěngine	driver	1	1.096217e-07

1534725 rows × 2 columns

In [423…

```
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[423…

w0	w1	n	mle
said	mr	4835	5.300210e-04
mr	pickwick	2062	2.260400e-04
dont	know	1810	1.984153e-04
old	man	1453	1.592804e-04
said	mrs	1393	1.527031e-04
...	...	...	...
hindoo	african	1	1.096217e-07
hindmost	wagon	1	1.096217e-07
	portion	1	1.096217e-07
hindering	flurry	1	1.096217e-07
ěngine	driver	1	1.096217e-07

749667 rows × 2 columns

Trigram table

In [23]:
```
trigram_df = train.LM[2].sort_values('n', ascending = False)
trigram_df
```

Out[23]:

			n	mle
w0	w1	w2		
</s>	<s>	<s>	393986	4.318943e-02
<s>	<s>	i	38136	4.180534e-03
		the	27928	3.061516e-03
		he	17736	1.944251e-03
		it	16086	1.763375e-03
...	...	...	...	...
have	worked	like	1	1.096217e-07
		me	1	1.096217e-07
		much	1	1.096217e-07
		myself	1	1.096217e-07
ěngine	driver	whom	1	1.096217e-07

4490726 rows × 2 columns

In [417…
```
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[417…

			n	mle
w0	w1	w2		
said	mr	pickwick	598	6.555379e-05
sir	said	mr	323	3.540782e-05
said	mr	pecksniff	247	2.707657e-05
		boffin	203	2.225321e-05
		dombey	201	2.203397e-05
...	...	...	...	...

			n	mle
w0	**w1**	**w2**		
havent	**come**	**along**	1	1.096217e-07
	damaged	**yet**	1	1.096217e-07
	decoyed	**us**	1	1.096217e-07
	done	**anything**	1	1.096217e-07
		much	1	1.096217e-07

426308 rows × 2 columns

Create language model

In [24]:
```
# 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 [25]:
```
# 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 [26]:
```
model.generate_text()
```

```
01. I WOULD SHOW BY WAY OF RIVER AN IGNORANT CREW LIKE THAT.

02. A LITTLE PAINTED STAND FOR SPINAL MENINGITIS SNODGRASS.

03. FOR FURTHER OBSERVATIONS ON HUMAN SUFFERING AND AGONY WHICH THIS CLUB IN SOM
E DEGREE ATTRIBUTABLE TO CUFFY.

04. HE SAW MY MISTAKE WAS IMMENSELY SATISFIED.

05. ALL THE STOCK AND WHENEVER THE DOCTOR CAME ACROSS THE FACE OF A LIME KILN.

06. THEY JOURNEY TO POUGHKEEPSIE.

07. BUT I SEEM TO.
```

```
08. HOW ARE YOU JEALOUS NOW SAID MR OMER IN COMPLIANCE WITH THIS POOR FELLOWS FA
CE NARROWLY WHILE HE DID VERY OFTEN SAY TO THE AMOUNT OF WEAZEN LIFE MAY PERHAPS
BE MY PARTNER.

09. HE IS WELL.

10. HAS HE NOT THOUGHT OF PUTTING THE NEW CANDIDATE FOR MAYOR WOULD FOLLOW HIS C
URIOUS TONE CAUSED BELLA TO ME.

11. DONT TURN MY EYE AND LIMB AND FEATURE THAT IT WAS BUT ONE CHURCH NEAR IT.

12. THE FATHER EARL OF MARCH KING BUT HIM.

13. HER FATHER WAS DEVOTED TO DISCOVERY THOUGHT BY SOME NEW OCCASION TO REMARK T
ESTILY THAT WHEN HE HAD COMMUNICATED SINCE THE MINISTRY OF THE REWARD PLACARD BE
FORE HE WAS A STRONG FOUNDATION OF THESE HERE WOTERS DOWN AND RESTED THERE IN A
SHADY SOLITUDE WHERE NEWS FROM SAINTS REST FOXS MARTYRS TUPPERS PROVERBIAL PHILO
SOPHY BOUND COPIES OF THE HOLY SEPULCHRE.

14. AFTER GLANCING AT THEM MORE AND MORE STERNLY.

15. WE ALL ECHOED A VERY LARGE STATION YET WHEN THE CANOES ARRIVED AT THE DOOR.

16. BECAUSE IT STILL UPON THE INDIVIDUAL WHO DESIRES TO GO AND SIT DOWN.

17. APPROACHING VANCOUVER THROUGH A HUNDRED YEARS MUST PASS.

18. LOUISA I CAN DO THAT AS I FIND THAT MR PICKWICK EYEING THE PLAN.

19. LET ME STAY TOO.

20. ONE THING WAS SO MUCH THAT THE CLEAR SKY HARDLY BROAD ENOUGH TO BE VERY MUCH
EVEN IF IT HAD TANGLED ME ALL THE MONOTONY OF SAMENESS IN THEM CARRY OFF EVERYTH
ING THEY COULD HELP IT.
```

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 [27]:
```
V = len(VOCAB)
```

In [28]:
```
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 [29]:
```
R
```

Out[29]:
```
[43, 51, 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 [30]:
```
w0 = ['he', 'she']

w1 = ['said', 'heard']

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

In [31]:
```
LM[1].loc[bigram_pairs].n.unstack()
```

Out[31]:

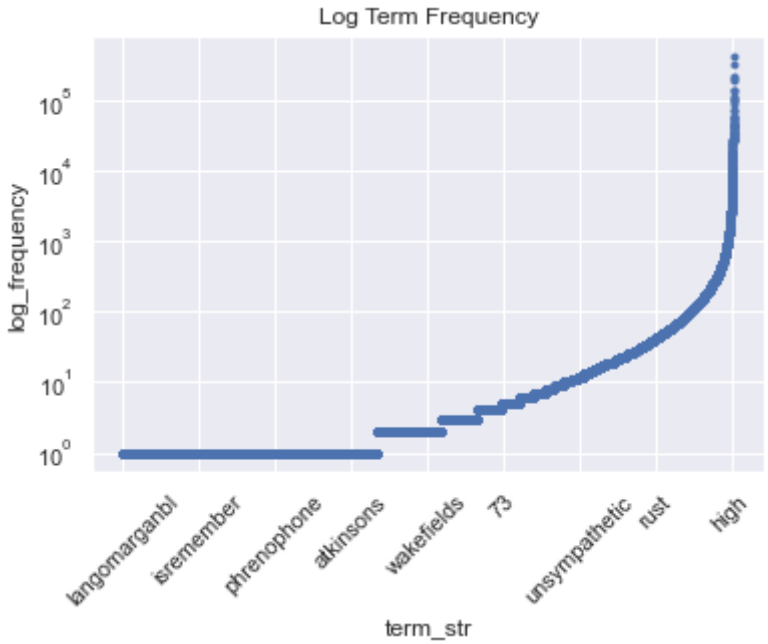
w0 \ w1	said	heard
he	3499	263
she	1225	82

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

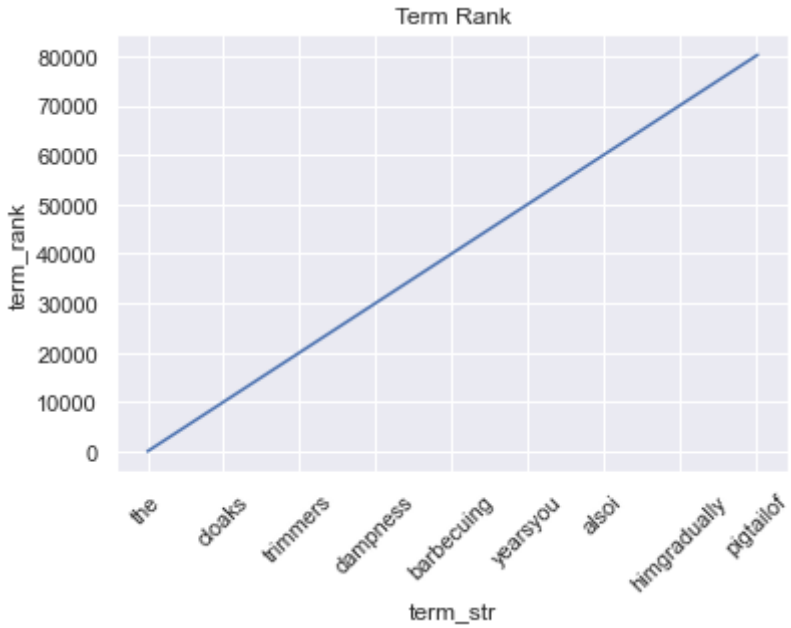
In [32]:
```
VOCAB.n.sort_values().plot(ylabel = "log_frequency", logy=True, style = '.', rot
```



Add Term Rank *r* to VOCAB

In [33]:
```
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 [34]:
```
VOCAB.term_rank.plot(ylabel = "term_rank", logx = False, rot = 45, title = "Term
```



In [35]:
```
VOCAB.head()
```

Out[35]:

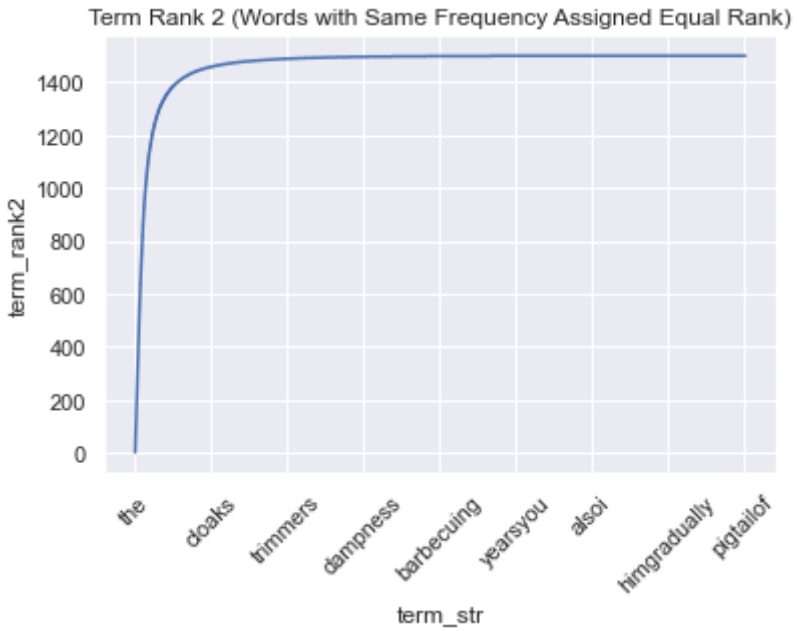
term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
the	1	418963	3	0.052764	4.244302	DT	22	{PRP, FW, RB, NN, JJS, NNP, VBZ, IN, VBG, VBD,...	1	
and	2	310105	3	0.039054	4.678368	CC	20	{PRP, FW, RB, PDT, NN, NNP, VBZ, IN, VBD, POS,...	1	
of	3	218996	2	0.027580	5.180221	IN	19	{PRP, FW, RB, PDT, NN, NNP, VBZ, IN, VBD, POS,...	1	
to	4	206700	2	0.026032	5.263587	TO	23	{WDT, FW, RB, PDT, NN, NNP, VBZ, IN, VBG, VBD,...	1	
a	5	189310	1	0.023842	5.390375	DT	21	{RBR, PRP, FW, RB, NN, NNP, VBZ, IN, VBD, POS,...	1	

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

In [36]:

```
# 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 [37]:
```
VOCAB.term_rank2.plot(ylabel = 'term_rank2', logx = False, rot = 45, title = "Te
```

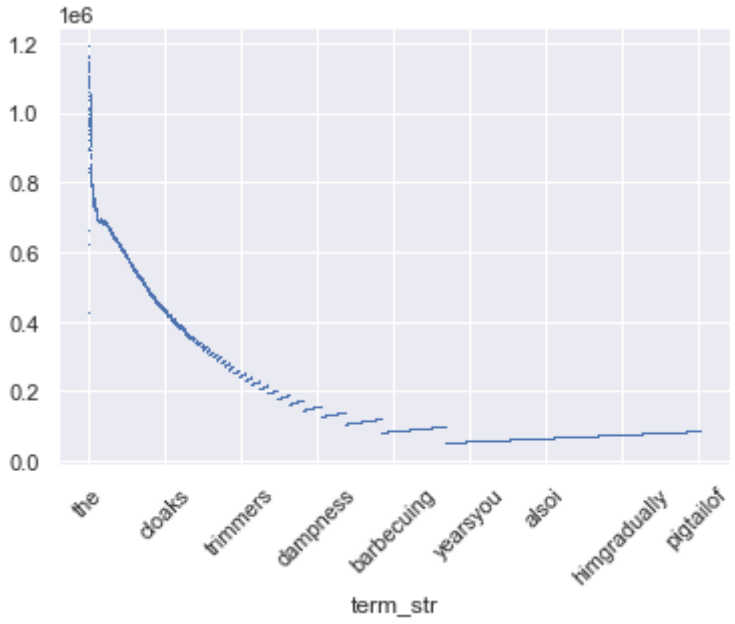


Compute Zipf's k using `term_rank` and `term_rank2`

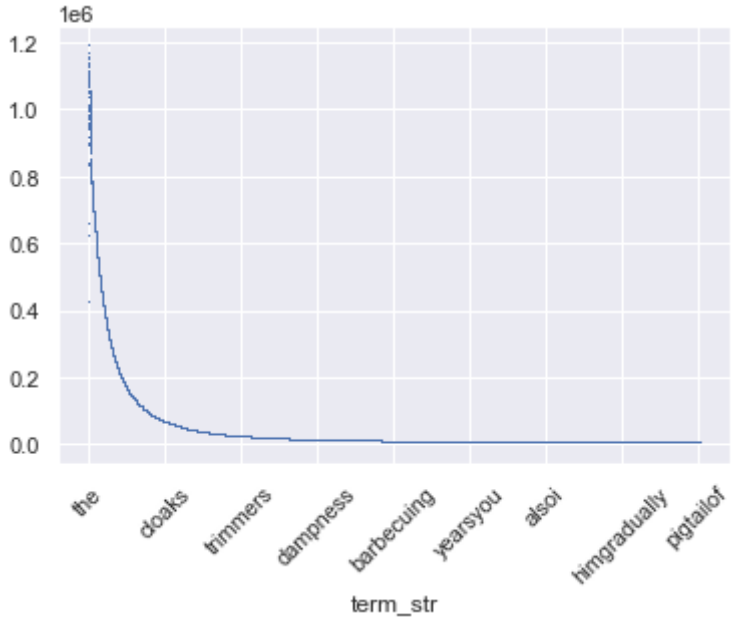
- **Equation**: $k = fr$

In [38]:
```
VOCAB['zipf_k'] = VOCAB.n * VOCAB.term_rank
VOCAB['zipf_k2'] = VOCAB.n * VOCAB.term_rank2
```

In [39]:
```
VOCAB.zipf_k.plot(style = ',', rot = 45);
```



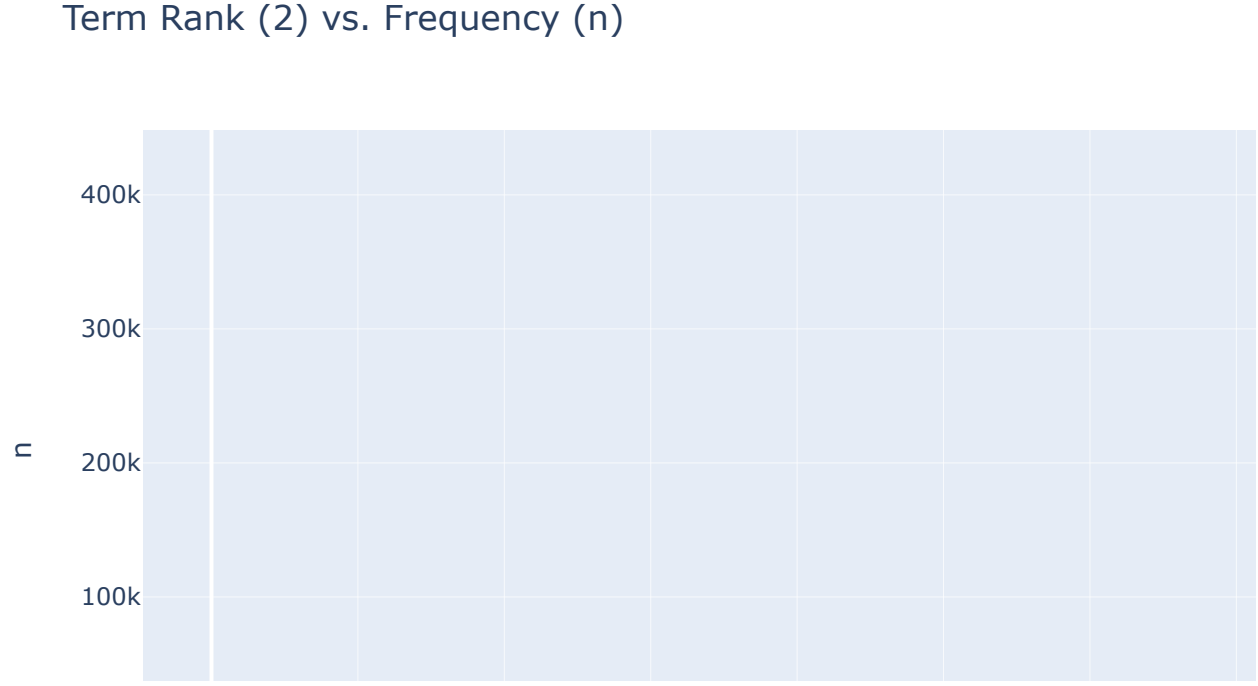
In [40]:
```
VOCAB.zipf_k2.plot(style = ',', rot = 45);
```

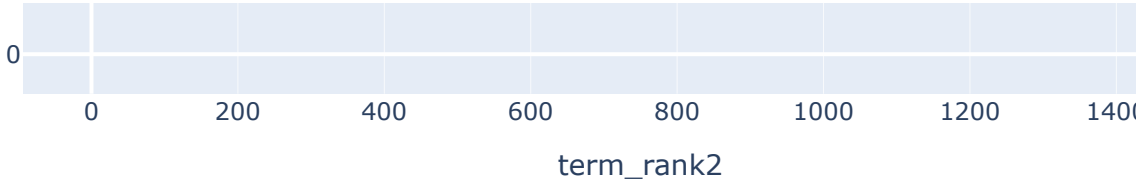


Rank vs. N (frequency n)

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

In [41]:
```
# 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 [42]:
```
BOW = create_bow(CORPUS, CHAPS)
```

In [43]:
```
DTCM, TFIDF, BOW, DFIDF, VOCAB = get_tfidf(BOW, VOCAB, tf_method = 'max', idf_me
```

In [44]:
```
VOCAB
```

Out[44]:

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop
the	1	418963	3	5.276399e-02	4.244302	DT	22	{PRP, FW, RB, NN, JJS, NNP, VBZ, IN, VBG, VBD,...	1
and	2	310105	3	3.905447e-02	4.678368	CC	20	{PRP, FW, RB, PDT, NN, NNP, VBZ, IN, VBD, POS,...	1
of	3	218996	2	2.758025e-02	5.180221	IN	19	{PRP, FW, RB, PDT, NN, NNP, VBZ, IN, VBD, POS,...	1
to	4	206700	2	2.603170e-02	5.263587	TO	23	{WDT, FW, RB, PDT, NN, NNP, VBZ, IN, VBG, VBD,...	1

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop
a	5	189310	1	2.384161e-02	5.390375	DT	21	{RBR, PRP, FW, RB, NN, NNP, VBZ, IN, VBD, POS,...	1
...	...	...	...	...	...	...	...	...	...
pipean	80288	1	6	1.259395e-07	22.920766	NN	1	{NN}	0
pipeand	80289	1	7	1.259395e-07	22.920766	NN	1	{NN}	0
pipeau	80290	1	6	1.259395e-07	22.920766	NNP	1	{NNP}	0
pipebut	80291	1	7	1.259395e-07	22.920766	NN	1	{NN}	0
ĕngine	80292	1	6	1.259395e-07	22.920766	NNP	1	{NNP}	0

80292 rows × 20 columns

Document-Term Count Matrix `DTCM`

In [45]:
```
DTCM
```

Out[45]:

book_id	chap_id	term_str	0	00	01	02	03	04	05	06	07	08	...	éternumens	étouffante	étr
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	
	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	
	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	
	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	
	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	
...	**...**		...	...	...	...	...	...	...	...	...	...	...	...	...	
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	
	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	
	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	
	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	
	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	

2290 rows × 80292 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 [46]:
```
# 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 [47]:
```
SIGS
```

Out[47]:

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
seem	528	1409	4	0.000177	12.460310	VB	10	{VBN, RB, JJ, NN, CC, NNP, VBZ, VBP, VBD, VB}	0	
sound	514	1453	5	0.000183	12.415947	NN	11	{VBN, RB, NNS, JJ, NN, NNP, VBZ, IN, VBP, VBD,...	0	
quiet	508	1470	5	0.000185	12.399165	JJ	14	{RBR, VBN, NNS, RB, JJ, RP, NN, WP, NNP, VBZ, ...	0	
knows	504	1475	5	0.000186	12.394266	VBZ	9	{VBN, NNS, JJ, NN, NNP, VBZ, VBP, VBD, VB}	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
pleasant	522	1415	8	0.000178	12.454179	JJ	10	{VBN, RB, NNS, JJ, NN, NNP, VBZ, IN, VBP, VB}	0	
...	...	...	...	...	...	...	...	...	...	
glanced	1441	476	7	0.000060	14.025948	VBD	6	{VBN, JJ, NN, VBP, VBD, VB}	0	
utmost	1460	470	6	0.000059	14.044249	JJ	4	{JJ, NNP, NN, RB}	0	
command	1325	517	7	0.000065	13.906745	NN	8	{JJ, NN, NNP, VBZ, VBG, VBP, VBD, VB}	0	
loose	1508	458	5	0.000058	14.081562	JJ	13	{RBR, VBN, RB, NNS, JJ, FW, NN, CC, NNP, VBZ, ...	0	
humble	1547	443	6	0.000056	14.129603	JJ	6	{VBN, RB, JJ, NN, NNP, VB}	0	

1000 rows × 20 columns

In [425…
```
SIGS.head(10).index.values
```

Out[425…
```
array(['seem', 'sound', 'quiet', 'knows', 'pleasant', 'red', 'past',
       'minutes', 'change', 'bring'], 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 [48]:
```
TFIDF
```

Out[48]:

book_id	chap_id	term_str 0	00	01	02	03	04	05	06	07	08	...	éternumens	étouffante	étr
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	
	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	
	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	
	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	
	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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	
	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	
	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	
	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	
	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	

2290 rows × 80292 columns

In [49]:
```
TFIDF_sigs = TFIDF[SIGS.index]
```

In [50]:
```
TFIDF_sigs
```

Out[50]:

book_id	chap_id	term_str seem	sound	quiet	knows	pleasant	red	past	min
70	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	2	0.005280	0.000000	0.001056	0.005273	0.002114	0.002107	0.002107	0.00
	3	0.000000	0.000000	0.000000	0.020499	0.000000	0.000000	0.013650	0.000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	5	0.012662	0.008441	0.004221	0.000000	0.004226	0.029474	0.004211	0.00
...	...	...	...	...	...	...	...	...	
62739	2	0.008019	0.004010	0.004010	0.016019	0.000000	0.000000	0.008000	0.000
	3	0.045108	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	4	0.005101	0.000000	0.000000	0.000000	0.000000	0.000000	0.005088	0.000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000

2290 rows × 1000 columns

In [51]:
```
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: 49.906167661726485
max TFIDF term: sir
```

In [52]:

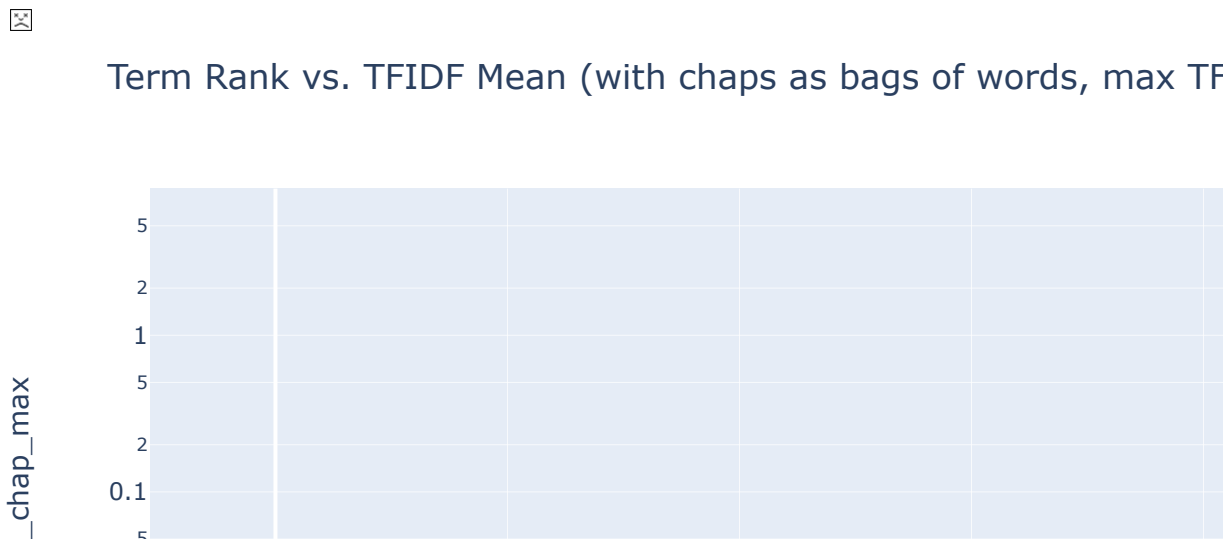
```
print('Max total TFIDF value by book and chapter:', max(TFIDF_sigs.sum(axis=1)))

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: 9.135380806295675
(Book_id, chap_id) with max total TFIDF:  (142, 7)
Title of book with max total TFIDF: The 30000 Bequest And Other Stories
```

Rank and TFIDF Mean

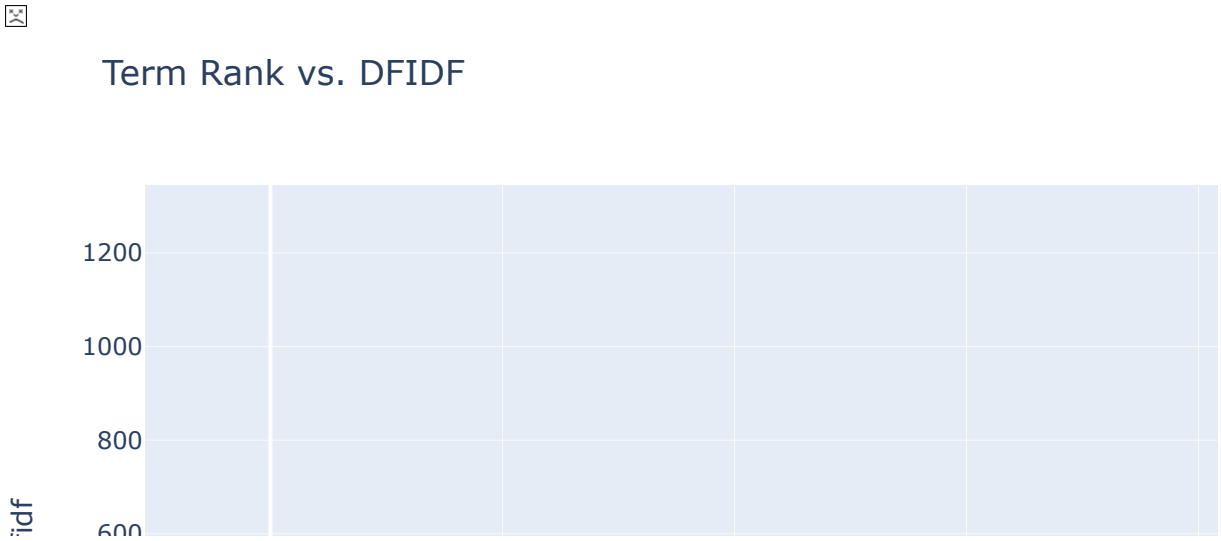
In [53]:

```
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 [54]:
```
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 [55]:
```
# group by book_id (BOOKS = OHCO[:1])
mean_TFIDF = TFIDF.groupby(BOOKS).mean()

mean_TFIDF
```

Out[55]:

term_str	0	00	01	02	03	04	05	06	07	08	...	éternumens	étouffante	étranger	év

term_str book_id	0	00	01	02	03	04	05	06	07	08	...	éternumens	étouffante	étranger	év
70	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
74	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
76	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
86	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
91	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
35536	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
60900	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.01442	0.0	
61522	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
62636	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	
62739	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000	0.0	

95 rows × 80292 columns

Mean TFIDF for all book for 1000 most significant terms only

In [56]:
```
mean_TFIDF_sigs = TFIDF_sigs.groupby(BOOKS).mean()

mean_TFIDF_sigs
```

Out[56]:

term_str book_id	seem	sound	quiet	knows	pleasant	red	past	minutes	cl
70	0.005686	0.006651	0.001146	0.003327	0.002894	0.006775	0.004189	0.003501	0.0
74	0.005353	0.008716	0.004168	0.002367	0.001992	0.006359	0.002387	0.005633	0.0
76	0.008142	0.007334	0.003783	0.004438	0.000445	0.002152	0.002387	0.004956	0.0
86	0.009043	0.006658	0.002853	0.001392	0.003136	0.002880	0.003008	0.004065	0.0(
91	0.012518	0.005794	0.005207	0.015241	0.000751	0.004442	0.006828	0.006162	0.0
...	...	...	...	...	...	...	...	...	
35536	0.000000	0.003818	0.009571	0.004909	0.005534	0.000000	0.013158	0.000000	0.0(
60900	0.011662	0.012412	0.002266	0.008758	0.004153	0.006389	0.004252	0.008771	0.0(
61522	0.008924	0.003971	0.001714	0.003060	0.001408	0.002623	0.007394	0.007150	0.0
62636	0.000000	0.000000	0.000000	0.022181	0.005558	0.000000	0.000000	0.000000	0.0(
62739	0.009705	0.000668	0.006837	0.002670	0.000000	0.000000	0.002181	0.000000	0.0(

95 rows × 1000 columns

DOC Table

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

In [57]:
```
book_DOC = pd.DataFrame(index = mean_TFIDF.index)
```

In [58]:
```
book_DOC = book_DOC.join(LIB[['author', 'title']])

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

In [59]:
```
book_DOC
```

Out[59]:

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
...	...	...	...
35536	dickens	the poems and verses of charles dickens	dickens 35536: the poems and verses of charles...
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

95 rows × 3 columns

Normalized Tables for Clustering

In [60]:
```
# 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 [61]:
```
assert round(L1.sum(1).sum()) == len(mean_TFIDF_sigs)
```

In [62]:
```
assert round(((L2.T)**2).sum().sum()) == len(mean_TFIDF_sigs)
```

Create table of book pairs (doc pair table PAIRS)

In [63]:
```
mean_TFIDF_sigs.T.corr().stack()
```

Out[63]:
```
book_id  book_id
70       70         1.000000
         74         0.155905
         76         0.063526
         86         0.226229
         91         0.071208
                       ...
62739    35536      0.028754
         60900      0.029163
         61522      0.072251
         62636      0.121954
         62739      1.000000
Length: 9025, dtype: float64
```

In [64]:
```
# 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 [65]:
```
PAIRS
```

Out[65]:

		corr_raw
doc_a	doc_b	
74	70	0.844095
76	70	0.936474
	74	0.524728
86	70	0.773771
	74	0.779680
...	...	...
62739	33077	0.804115
	35536	0.971246
	60900	0.970837
	61522	0.927749
	62636	0.878046

4465 rows × 1 columns

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

In [66]:
```
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 [67]:
```
for X, metric, label in combos:
    PAIRS[label] = pdist(X, metric)
```

Compare Distributions

In [68]:
```
PAIRS.head(20)
```

Out[68]:

		corr_raw	cityblock-raw	euclidean-raw	euclidean-l2	cosine-raw	cityblock-l1	jaccard-l0	
doc_a	doc_b								
74	70	0.844095	3.571013	0.193548	0.953050	0.454152	0.823824	0.059416	0.1
76	70	0.936474	3.937929	0.259376	1.121754	0.629166	1.023973	0.184211	0.2
	74	0.524728	2.720471	0.141738	0.840126	0.352906	0.668990	0.033199	0.
86	70	0.773771	3.871357	0.259483	1.139019	0.648683	1.061109	0.292683	0.3
	74	0.779680	4.328619	0.276506	1.165533	0.679233	1.170515	0.372967	0.3
	76	0.787390	3.417256	0.162168	0.955848	0.456823	0.826704	0.029000	0.1
91	70	0.928792	3.664634	0.211988	0.980174	0.480371	0.783302	0.063636	0.
	74	0.518159	2.362593	0.119425	0.778301	0.302877	0.637118	0.023092	0.0
	76	0.162367	2.836389	0.170178	0.897120	0.402413	0.685036	0.036290	0.
	86	0.762499	2.541155	0.164262	0.915180	0.418778	0.656441	0.027081	0.0
93	70	0.963467	4.005373	0.183762	0.965004	0.465616	0.843175	0.031031	0.1
	74	0.581724	3.722674	0.215505	1.093317	0.597671	0.904213	0.026000	0.0
	76	0.148472	4.492781	0.222441	1.042723	0.543636	0.937451	0.064000	0.
	86	0.786666	4.500904	0.220663	1.061680	0.563583	1.003266	0.141283	0.2
	91	0.221026	3.323835	0.158690	0.990244	0.490291	0.935656	0.056169	0.
98	70	1.049178	4.799437	0.237051	1.038975	0.539734	0.952565	0.128257	0.
	74	1.049602	3.133375	0.149915	0.933864	0.436051	0.814625	0.032000	0.
	76	1.059715	5.123451	0.287079	1.095881	0.600478	0.962716	0.115346	0.2

doc_a	doc_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0	
	86	0.983786	2.905040	0.148502	0.984102	0.484228	0.881336	0.051102	0.1
	91	1.082516	3.797932	0.183327	0.985437	0.485543	0.837634	0.026000	0.0

Hierarchical agglomerative cluster diagrams for the distance measures

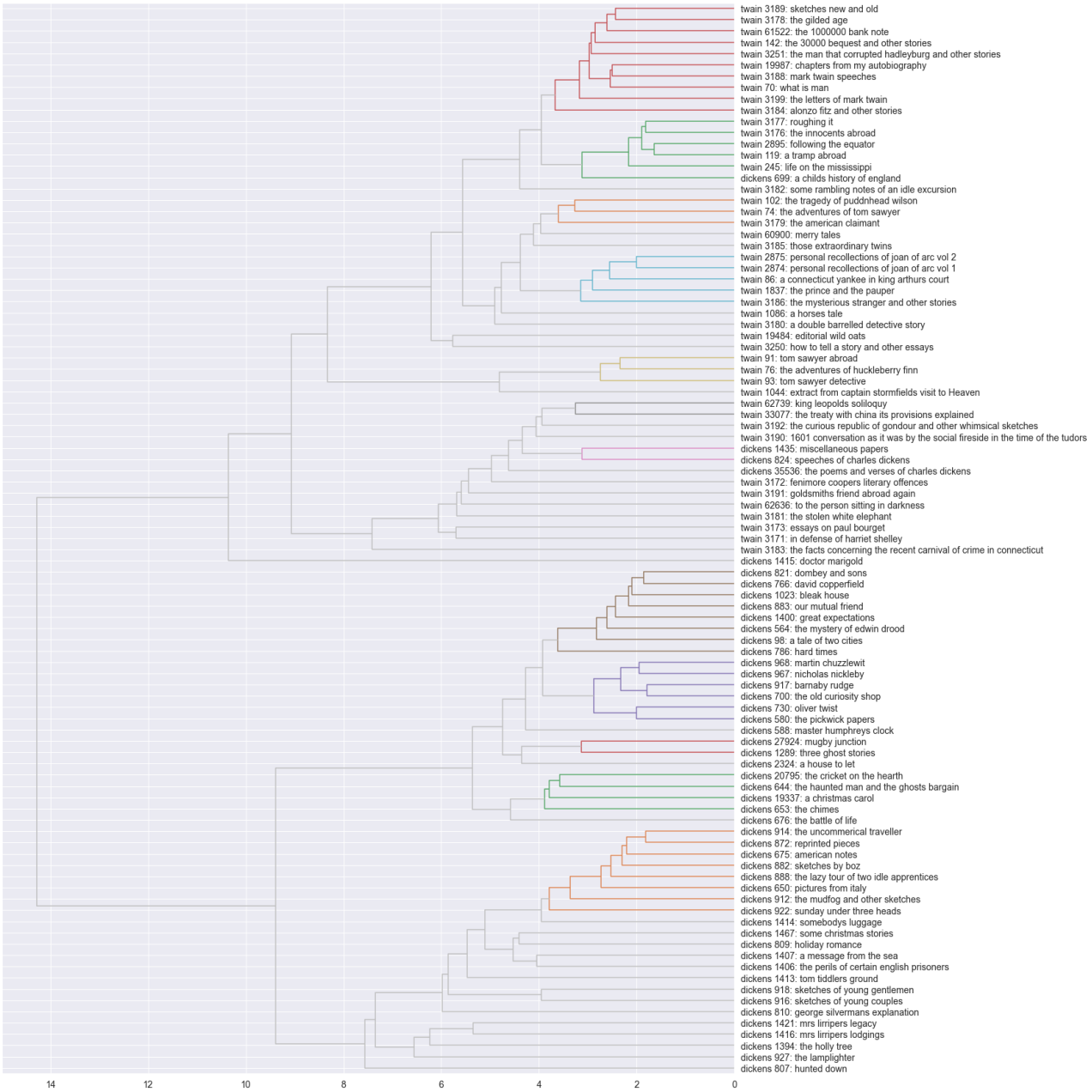
In [69]:
```
LIB['label'] = book_DOC['label']
```

In [213…
```
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 [214…
```
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)
```

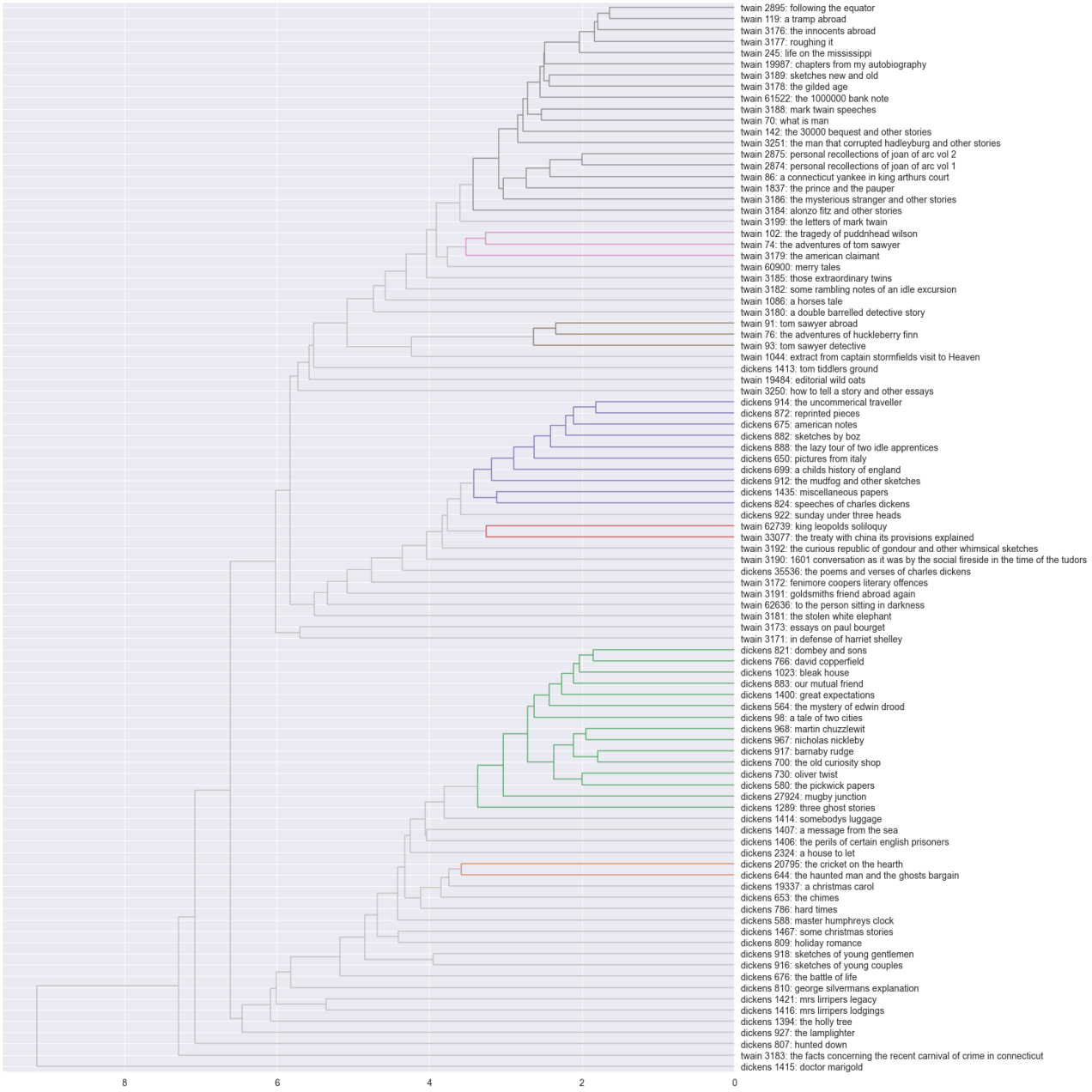
```
/var/folders/3n/4b11y5qn5cn20kztppfbsxq40000gn/T/ipykernel_13153/3242109510.py:1
4: RuntimeWarning:

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>
```

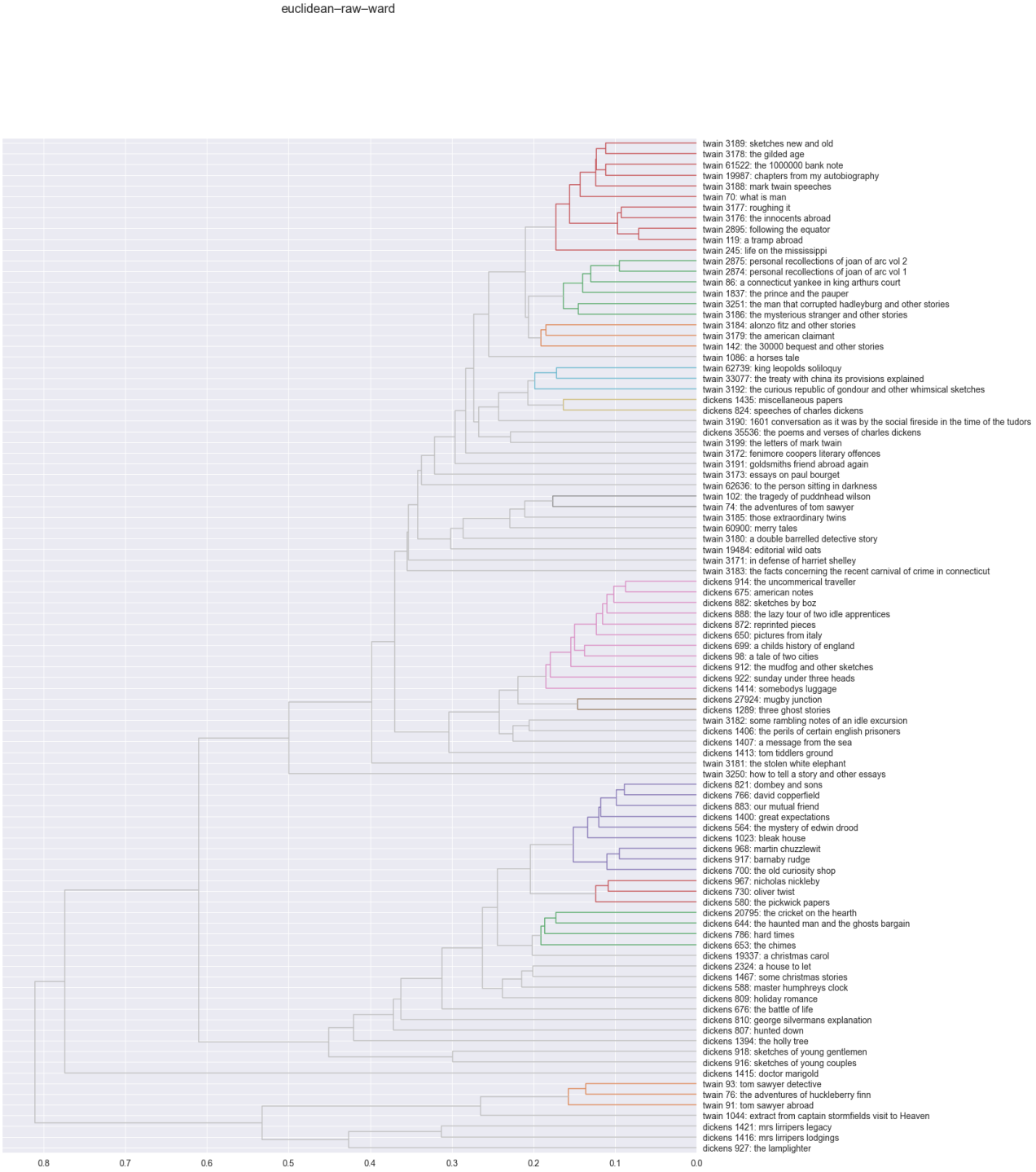



```
<Figure size 432x288 with 0 Axes>
```

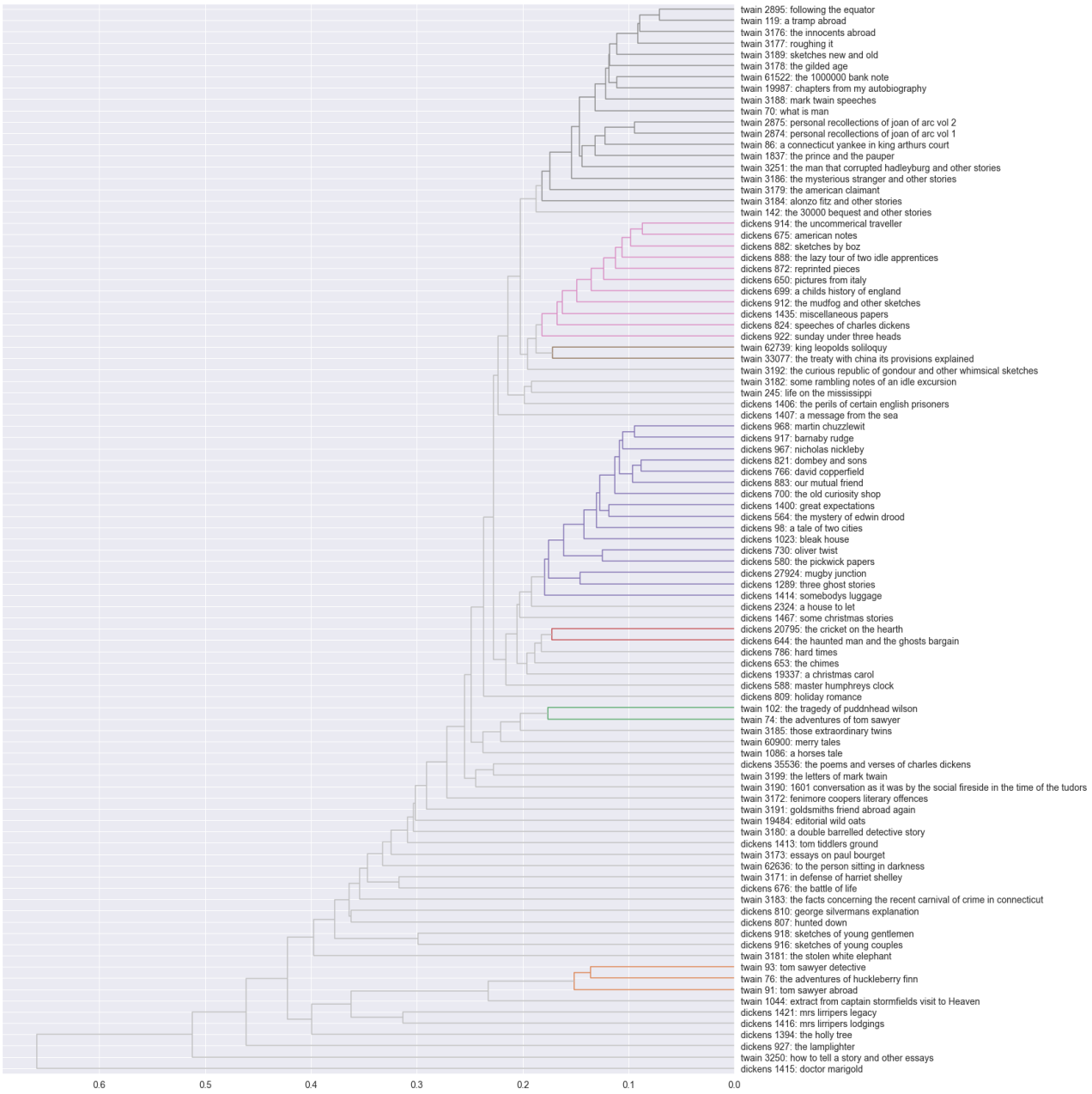
cityblock–raw–weighted



<Figure size 432x288 with 0 Axes>

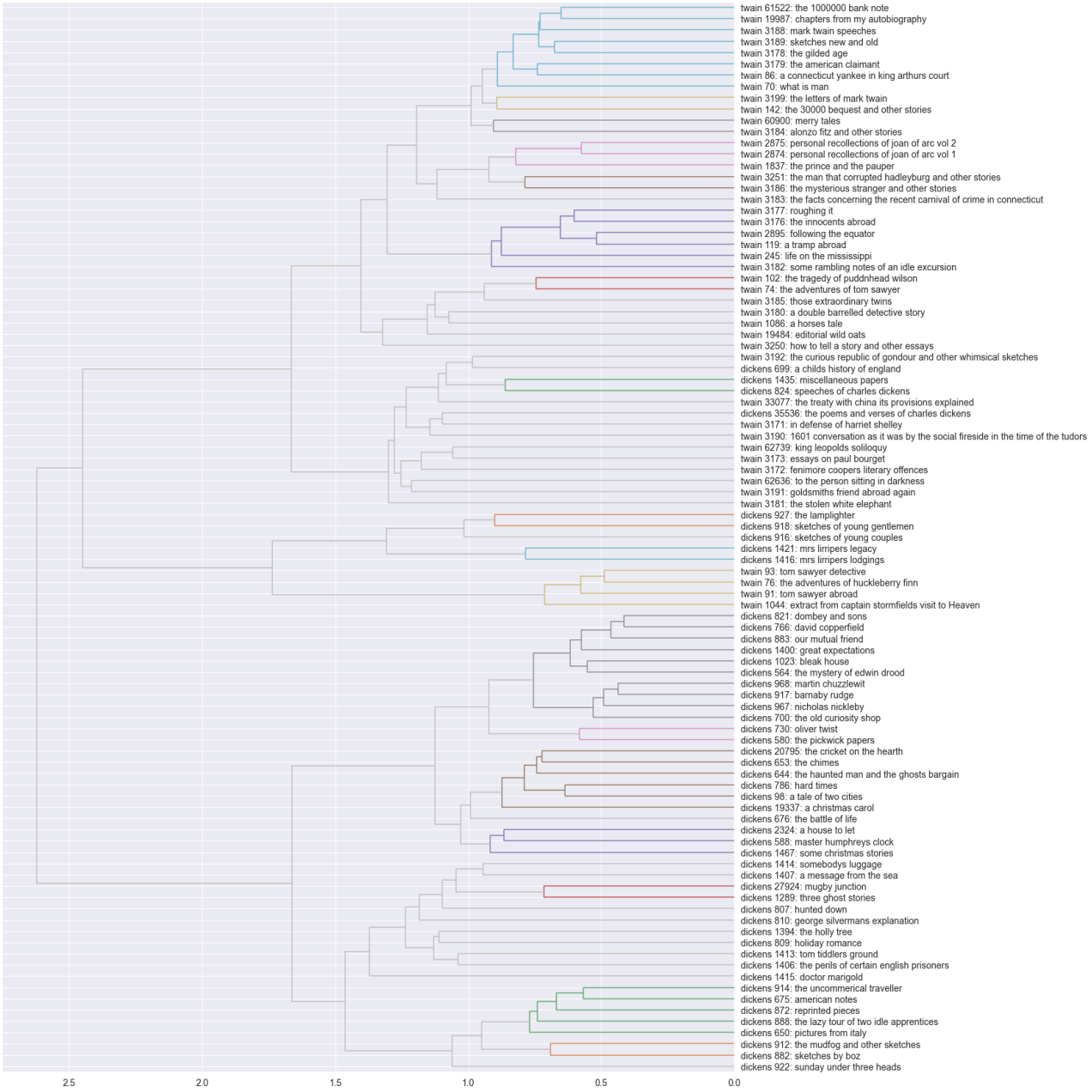


<Figure size 432x288 with 0 Axes>

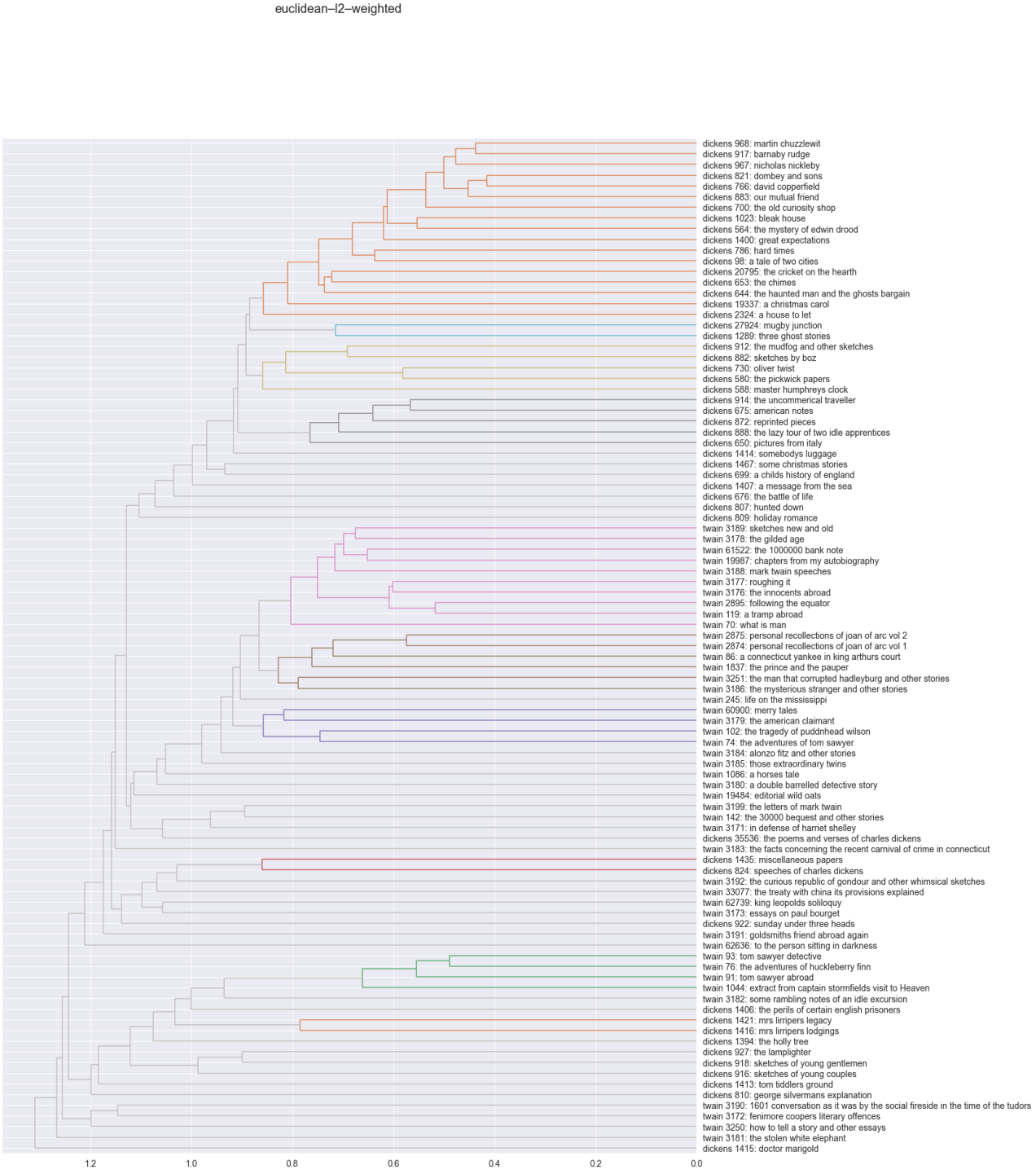


<Figure size 432x288 with 0 Axes>

euclidean–l2–ward



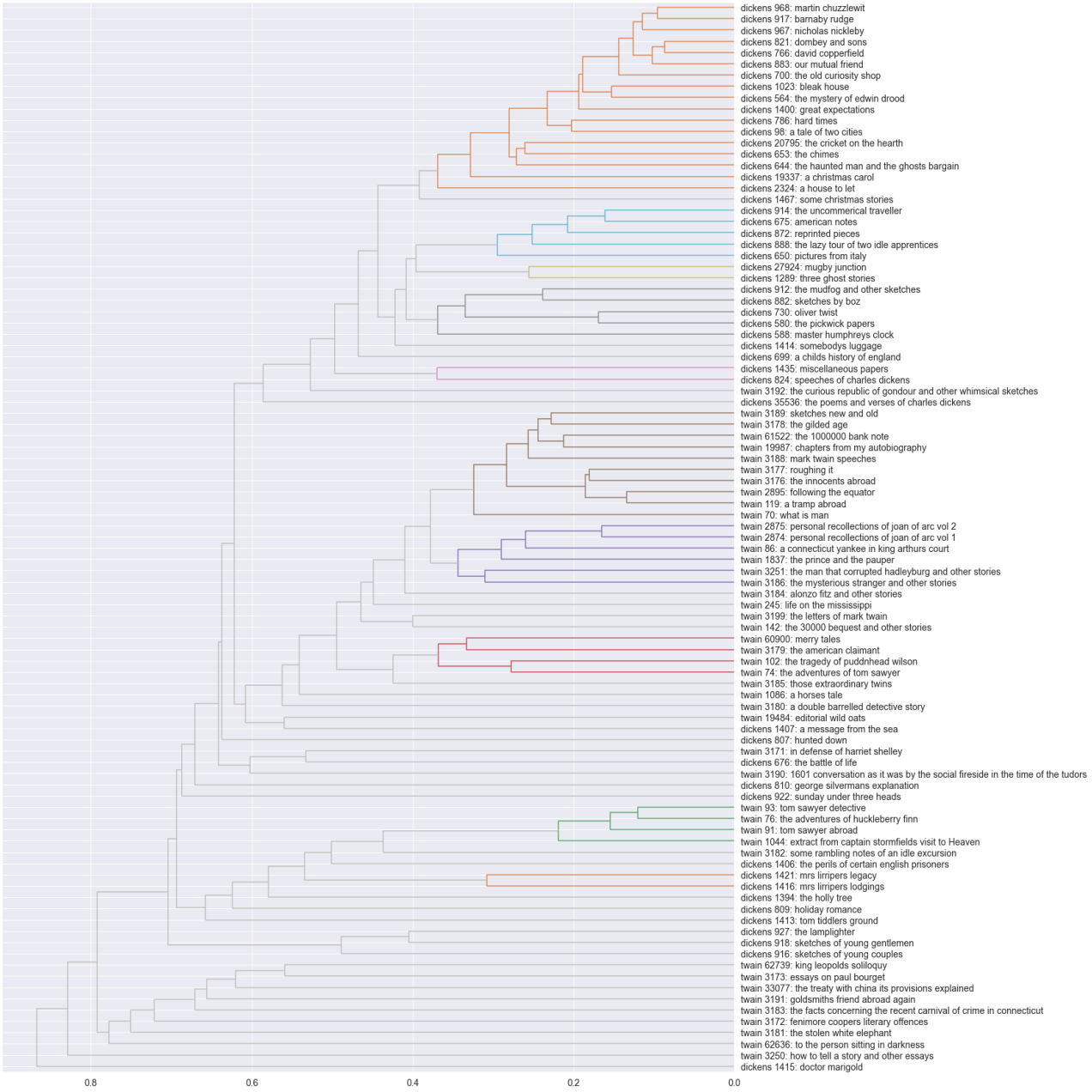
<Figure size 432x288 with 0 Axes>




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

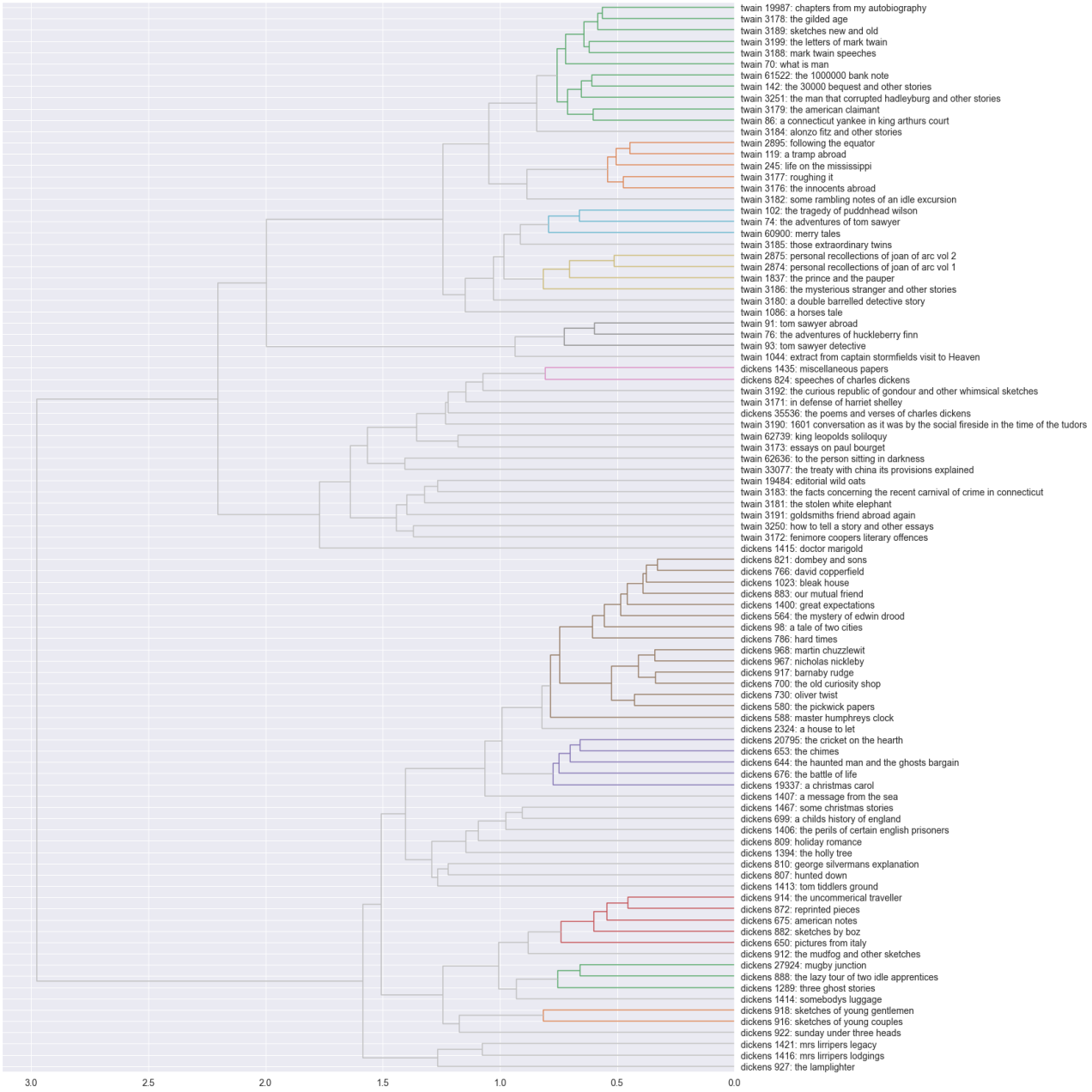
<Figure size 432x288 with 0 Axes>

cosine–raw–weighted

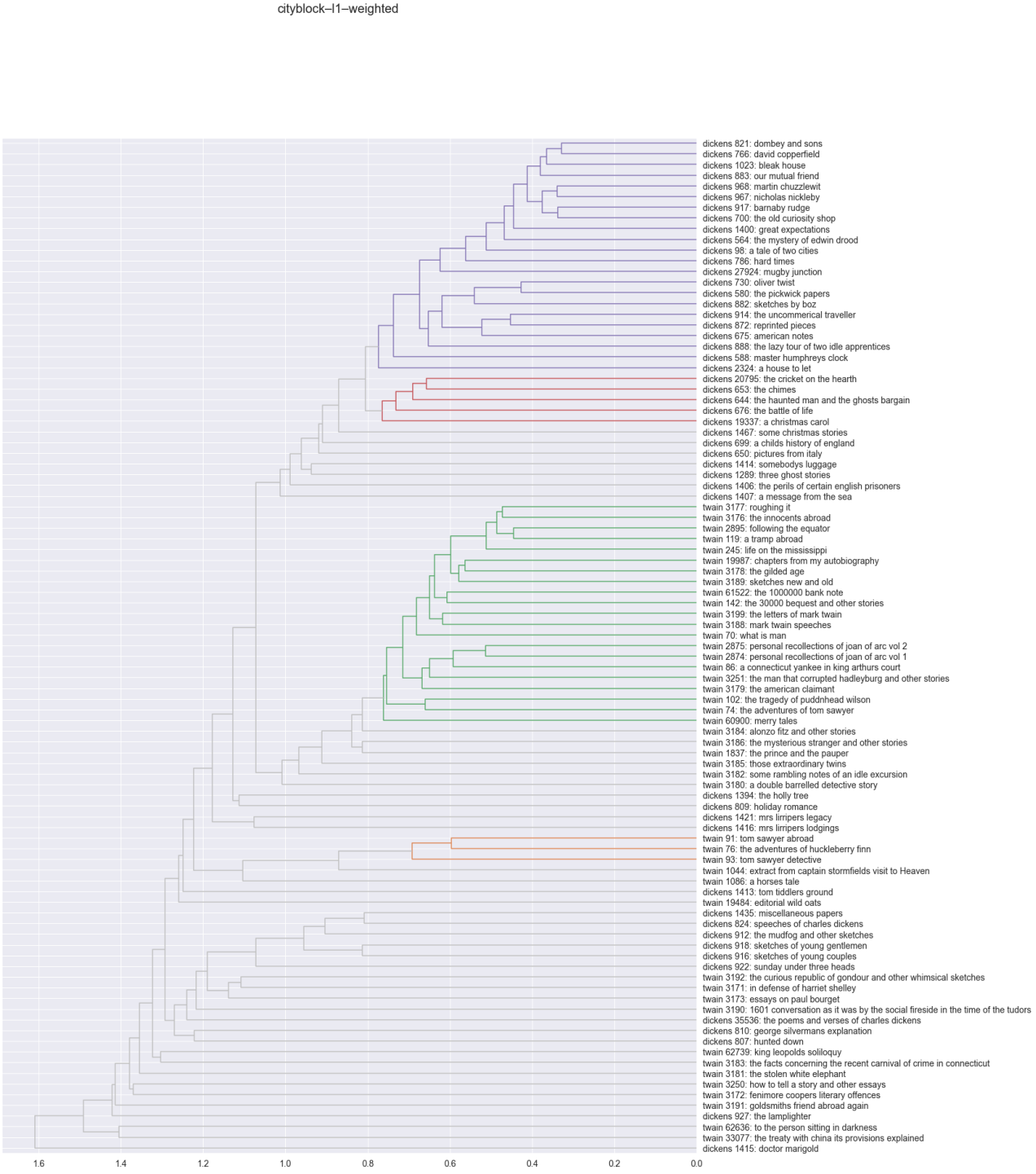


<Figure size 432x288 with 0 Axes>

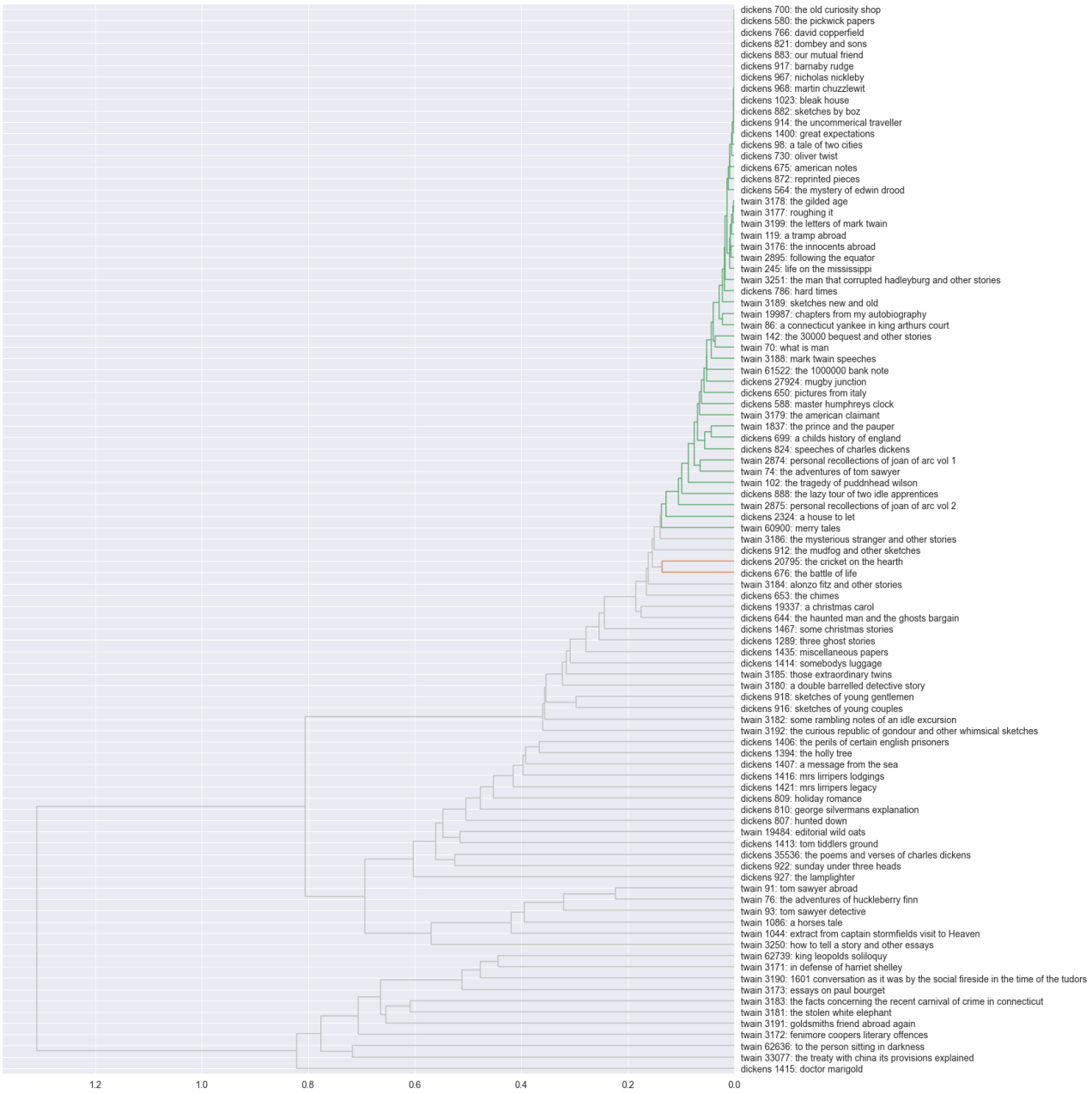
cityblock–l1–ward



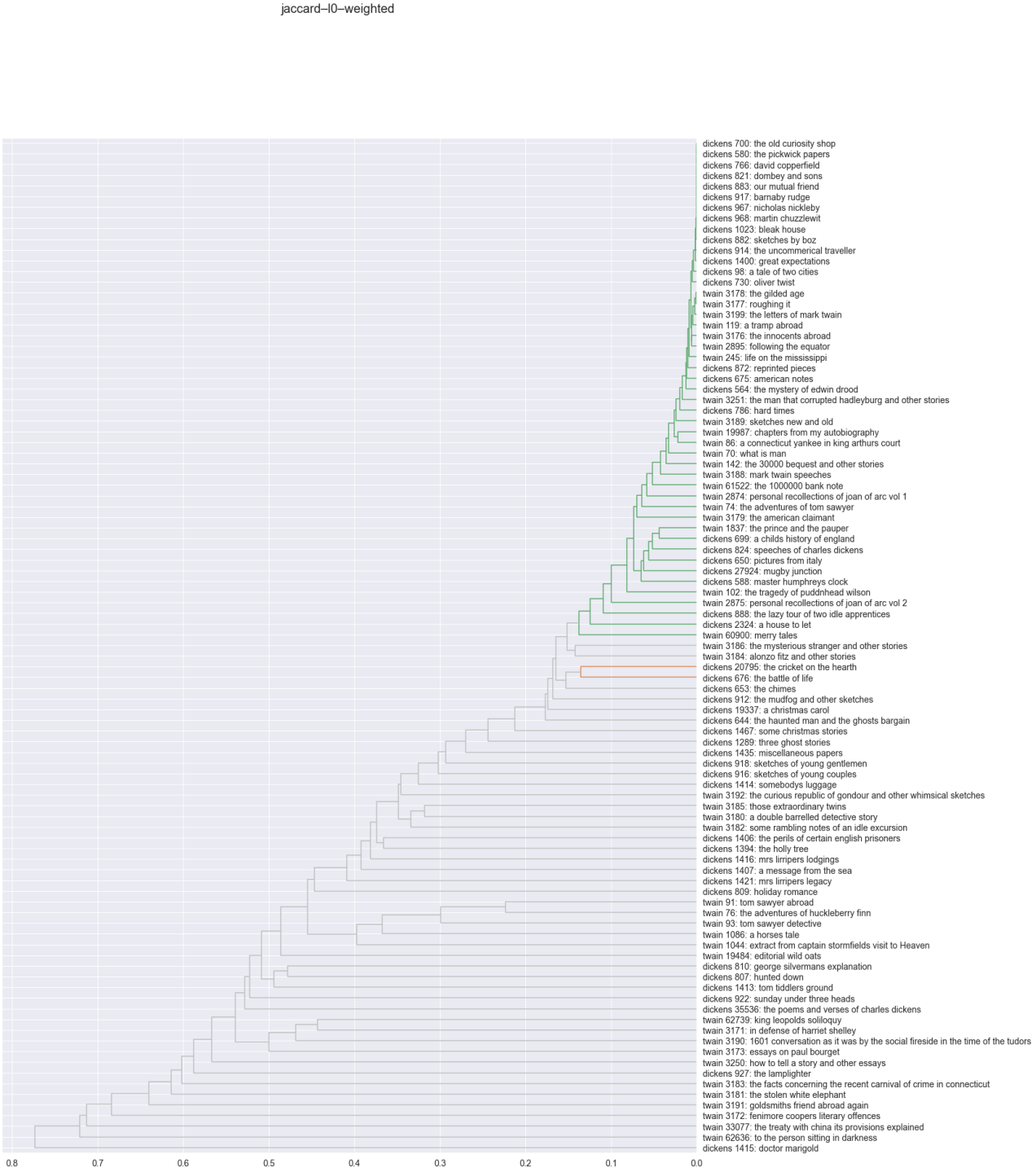
<Figure size 432x288 with 0 Axes>



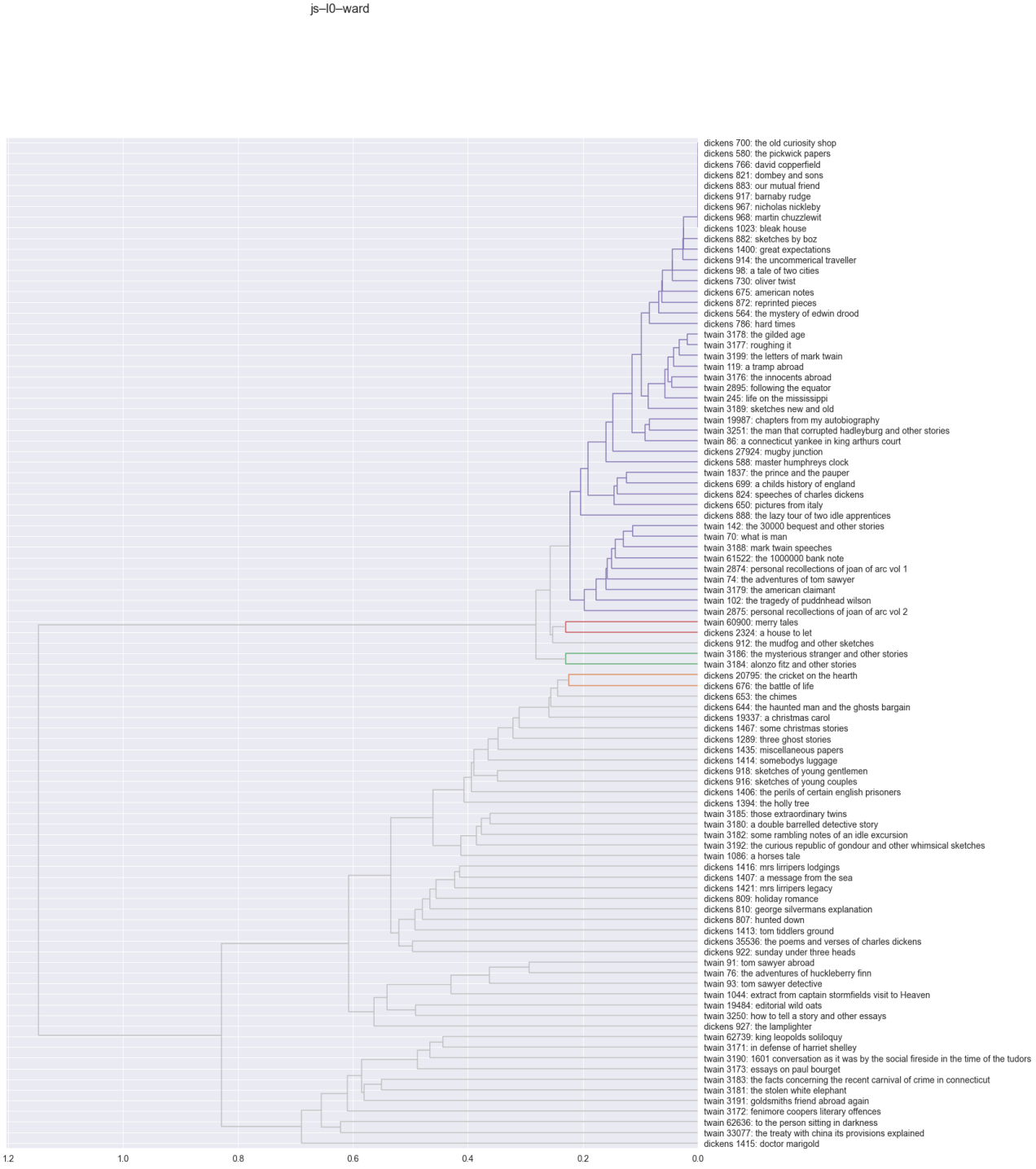
<Figure size 432x288 with 0 Axes>



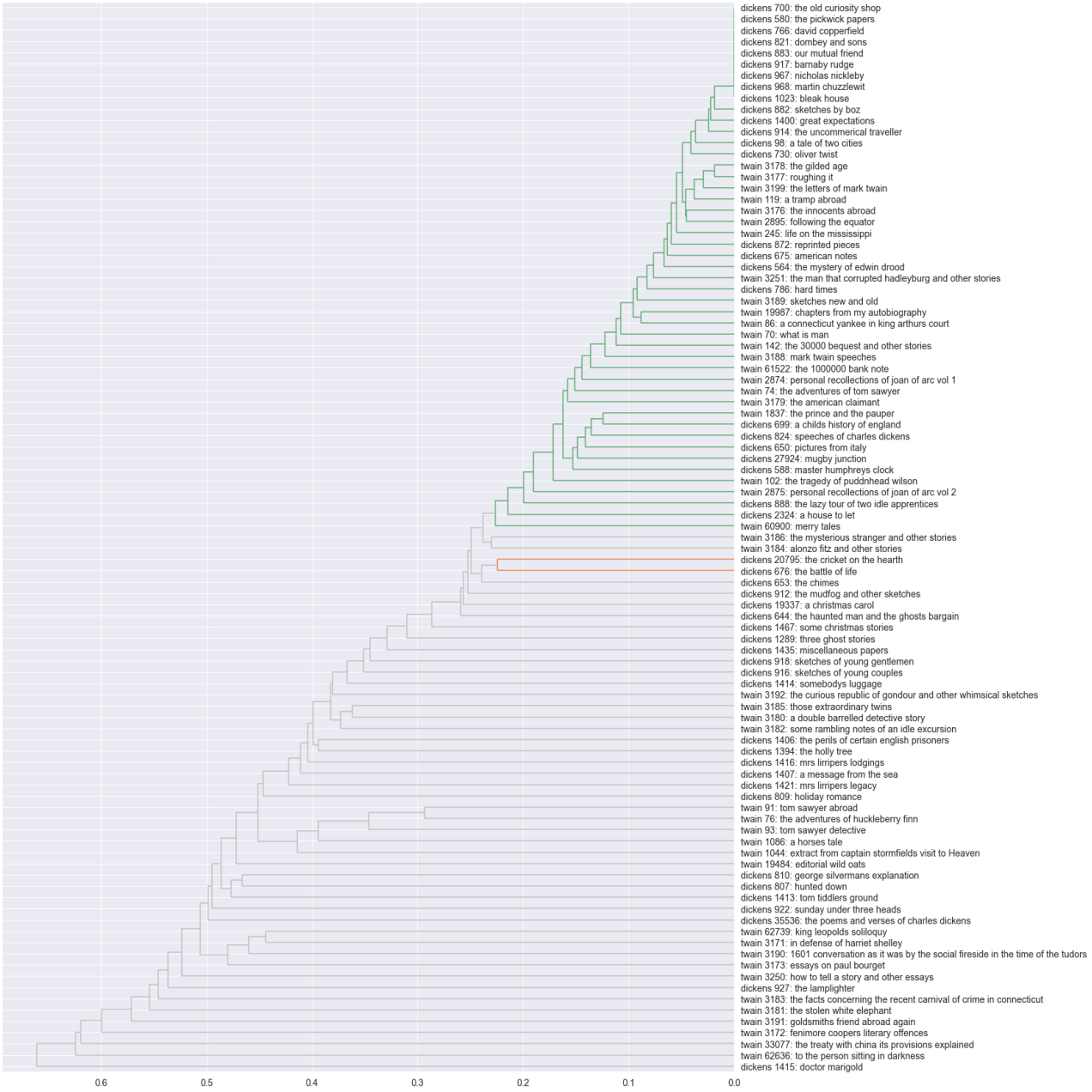
<Figure size 432x288 with 0 Axes>



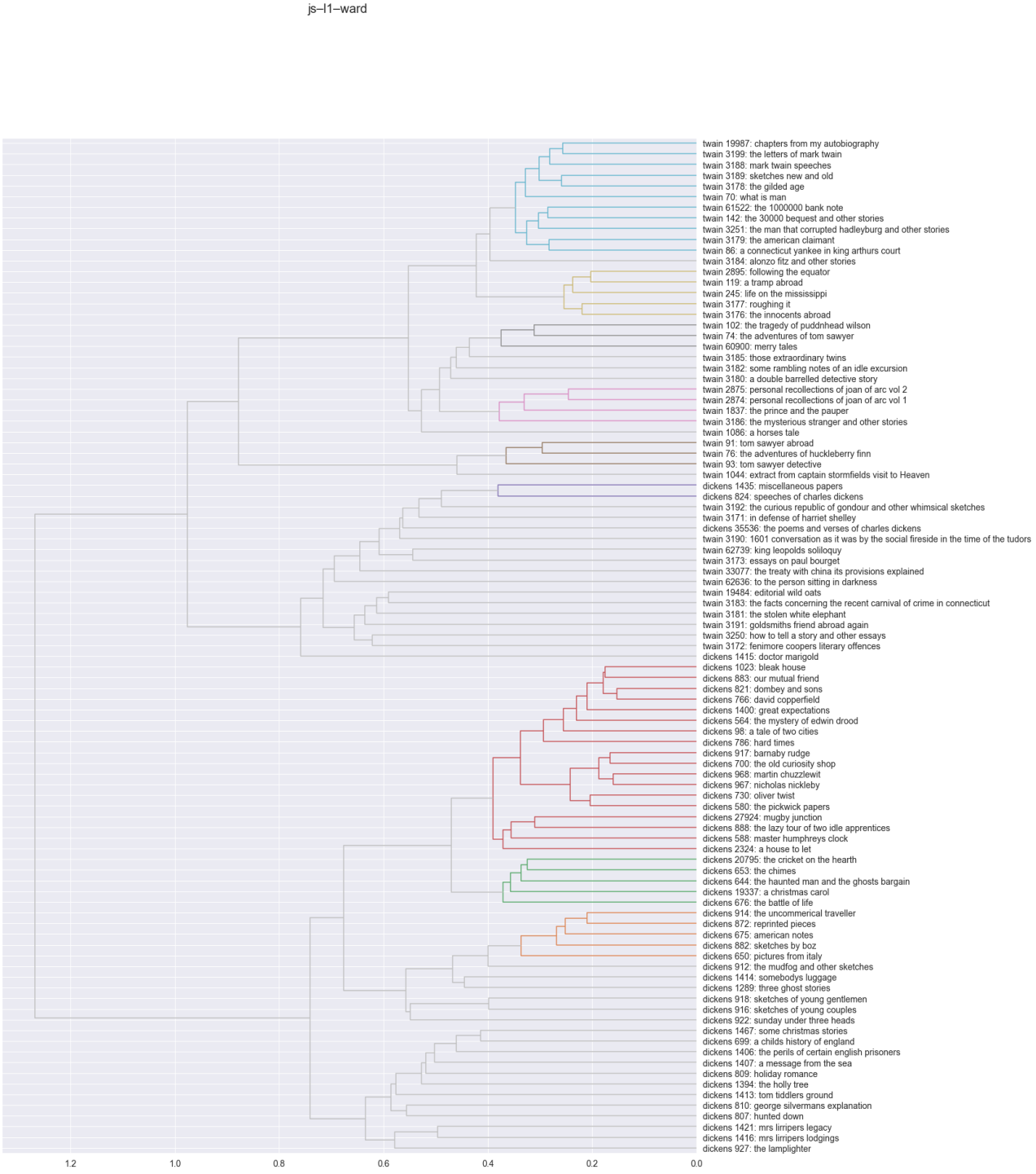
<Figure size 432x288 with 0 Axes>



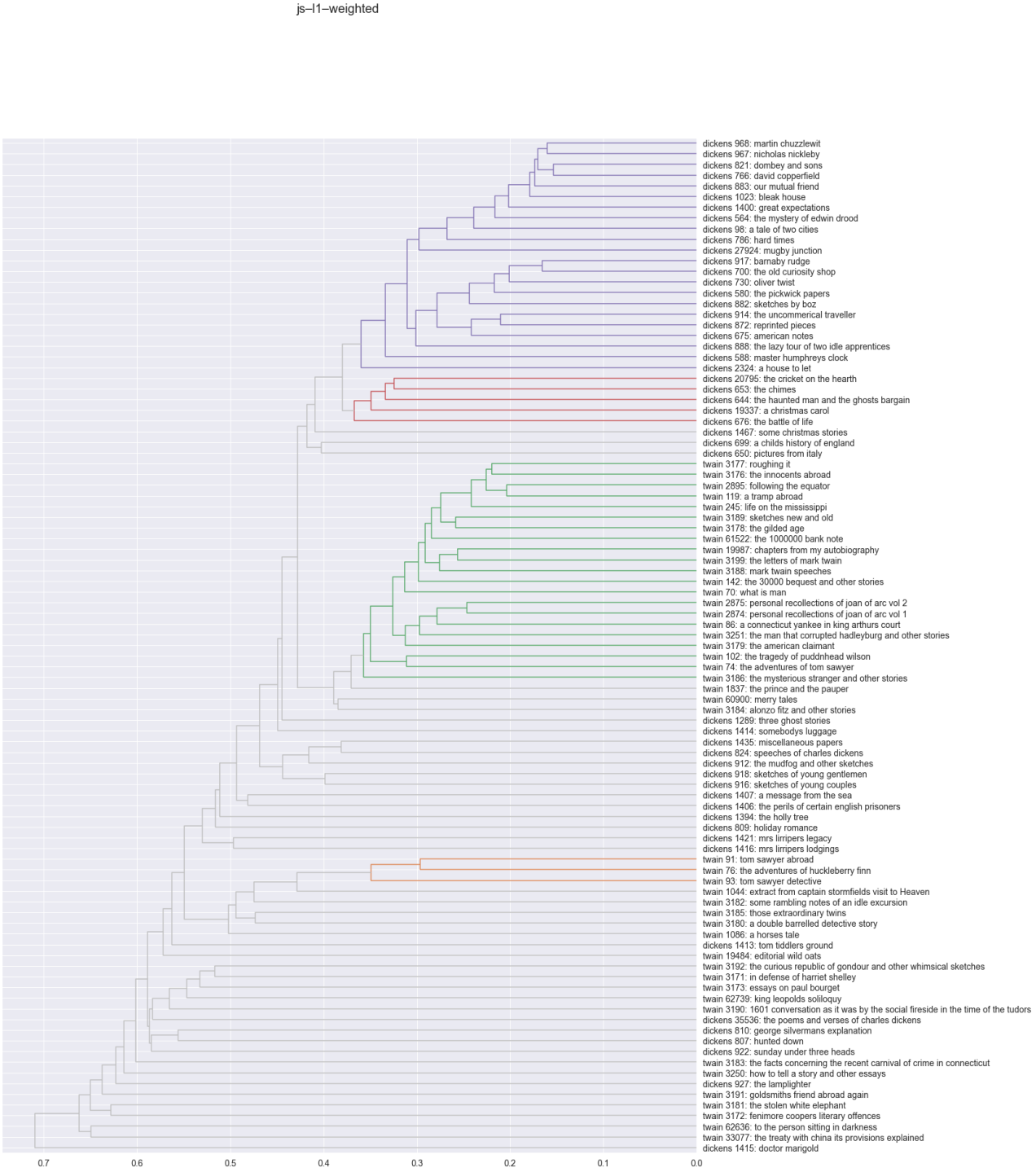

```
<Figure size 432x288 with 0 Axes>
```



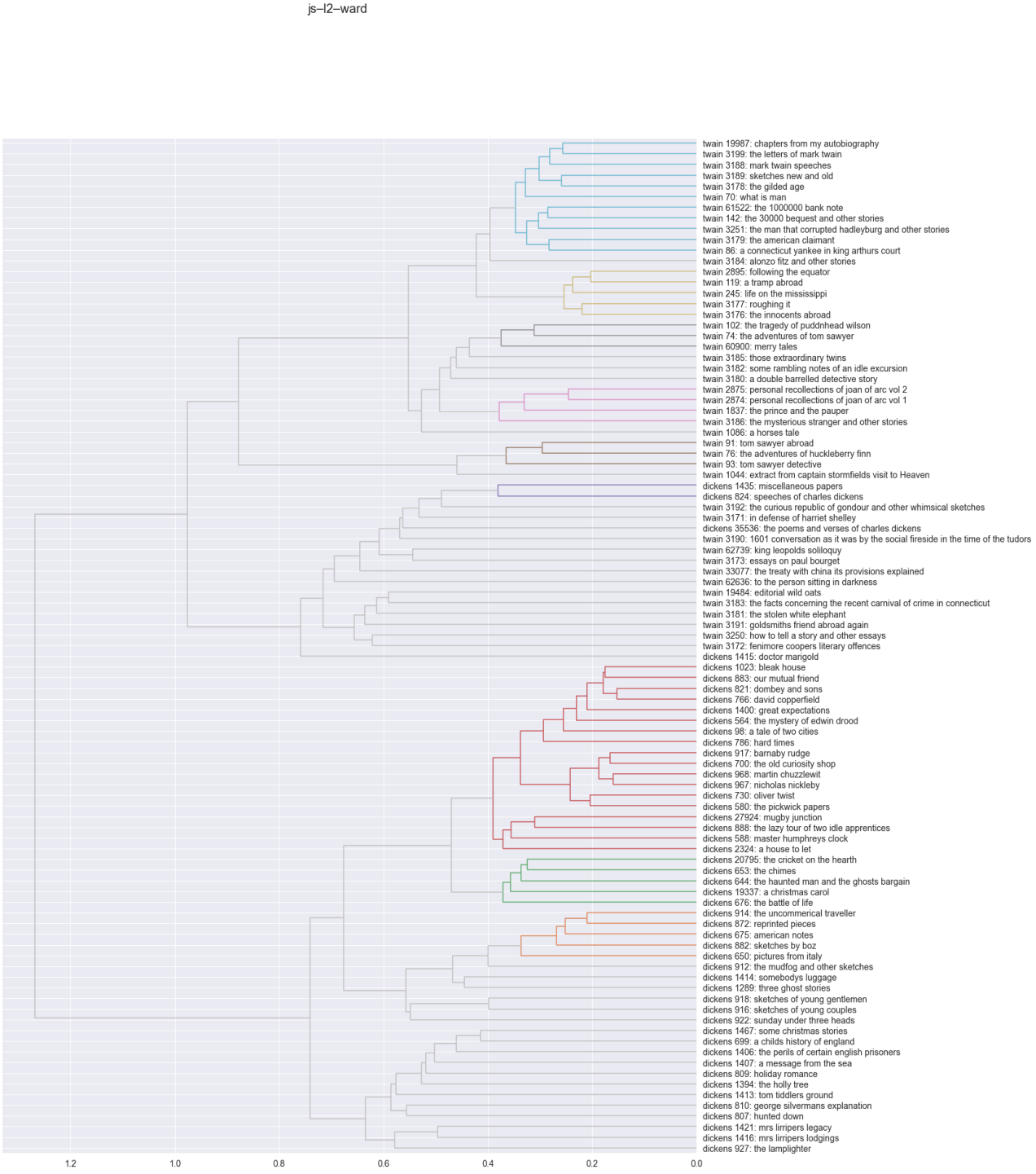
<Figure size 432x288 with 0 Axes>



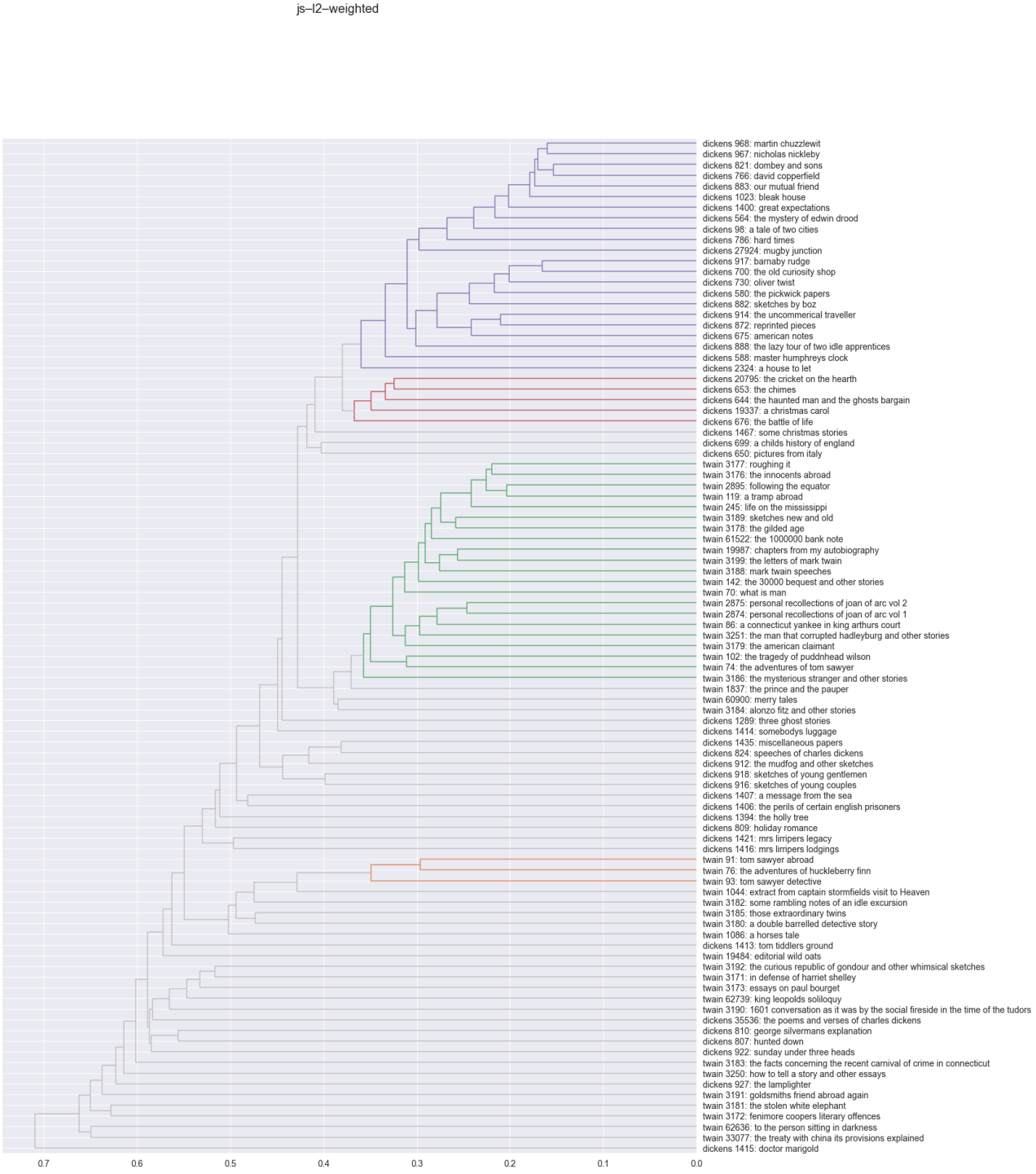
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



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

In [72]:

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

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

Out[72]:

term_str	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
sound	514	1453	5	0.000183	12.415947	NN	11	{VBN, RB, NNS, JJ, NN, NNP, VBZ, IN, VBP, VBD,...	0	
minutes	490	1510	7	0.000190	12.360433	NNS	10	{FW, NNS, JJ, CD, NN, NNP, VBZ, IN, VBP, VBD}	0	
change	501	1480	6	0.000186	12.389384	NN	11	{NNS, JJ, NN, CC, NNP, VBZ, VBG, VBP, VBD, VB,...	0	
purpose	529	1409	7	0.000177	12.460310	NN	9	{NNS, RB, JJ, NN, NNP, VBZ, VBP, VBD, VB}	0	
character	498	1495	9	0.000188	12.374836	NN	11	{VBN, NNS, RB, JJ, NN, NNP, VBZ, VBP, VBD, JJR...	0	
father	262	3097	6	0.000390	11.324110	NN	15	{RBR, PRP$, PRP, RB, NNS, JJ, NN, CC, NNP, VBZ...	0	
foot	523	1415	4	0.000178	12.454179	NN	12	{VBN, NNS, FW, JJ, PDT, NN, RB, NNP, VBZ, VBP,...	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
attention	574	1277	9	0.000161	12.602223	NN	11	{NNS, RB, JJ, RP, NN, NNP, VBZ, IN, VBP, VBD, VB}	0	
mother	266	3070	6	0.000387	11.336743	NN	18	{RBR, PRP$, PRP, FW, NNS, RB, JJ, MD, NN, CD, ...	0	
arm	414	1882	3	0.000237	12.042715	NN	11	{RB, NNS, JJ, RP, NN, NNP, VBZ, IN, VBP, VBD, VB}	0	
order	550	1345	5	0.000169	12.527375	NN	15	{PRP, FW, NNS, RB, JJ, RP, NN, VBN, NNP, VBZ, ...	0	
feeling	547	1354	7	0.000171	12.517754	NN	11	{VBN, NNS, JJ, PDT, NN, NNP, VBZ, VBG, VBP, VB...	0	
silence	556	1323	7	0.000167	12.551168	NN	12	{VBN, NNS, RB, JJ, NN, CC, NNP, VBZ, IN, VBP, ...	0	
anybody	536	1390	7	0.000175	12.479897	NN	15	{PRP, VBN, NNS, RB, JJ, RP, NN, CD, NNP, VBZ, ...	0	

term_str	term_rank	n	n_chars	p	i	max_pos	n_pos	cat_pos	stop	ste
chair	378	2058	5	0.000259	11.913738	NN	7	{JJ, NN, NNP, VBZ, VBP, VBD, VB}	0	
forth	559	1312	5	0.000165	12.563214	NN	14	{RBR, VBN, RB, NNS, JJ, RP, NN, PDT, NNP, VBZ,...	0	
corner	457	1659	6	0.000209	12.224668	NN	10	{FW, NNS, VBN, JJ, NN, NNP, VBZ, VBP, VBD, VB}	0	
truth	534	1397	5	0.000176	12.472649	NN	11	{FW, NNS, RB, JJ, PDT, NN, CC, NNP, VBZ, IN, VB}	0	
theres	418	1840	6	0.000232	12.075276	NN	20	{FW, RB, PDT, NN, NNP, VBZ, IN, VBG, VBD, POS,...	0	
book	346	2265	4	0.000285	11.775470	NN	10	{NNS, RB, JJ, NN, NNP, VBZ, IN, VBP, VBD, VB}	0	

In [73]:
```
top_20_nouns = list(VOCAB.loc[VOCAB.max_pos.isin(noun_tags)].sort_values('dfidf'
```

In [74]:
```
print(top_20_nouns)
```

```
['sound', 'minutes', 'change', 'purpose', 'character', 'father', 'foot', 'attent
```

```
ion', 'mother', 'arm', 'order', 'feeling', 'silence', 'anybody', 'chair', 'fort
h', 'corner', 'truth', 'theres', 'book']
```

Most "Significant" Book based on mean TFIDF

In [75]:
```
BOW.groupby(BOOKS).mean().sort_values('tfidf', ascending = False).join(LIB, on =
```

Out[75]:

book_id	n	tf	tfidf	source_file_path	title	
3188	2.540928	0.054027	0.076034	Twain/3188-mark_twain_speeches.txt	mark twain speeches	INTR(
35536	2.144928	0.037376	0.074466	Dickens/35536-the_poems_and_verses_of_charles_...	the poems and verses of charles dickens	THE V $\|THE$
3191	2.330049	0.043699	0.059526	Twain/3191-goldsmiths_friend_abroad_again.txt	goldsmiths friend abroad again	
1415	2.952430	0.044903	0.059090	Dickens/1415-doctor_marigold.txt	doctor marigold	
1086	2.659144	0.040938	0.056994	Twain/1086-a_horses_tale.txt	a horses tale	
...	...	...	...	...	...	
1406	5.571753	0.009851	0.011944	Dickens/1406-the_perils_of_certain_english_pri...	the perils of certain english prisoners	^C
1289	4.083012	0.009729	0.011733	Dickens/1289-three_ghost_stories.txt	three ghost stories	HOl
644	4.975638	0.008870	0.011279	Dickens/644-the_haunted_man_and_the_ghosts_bar...	the haunted man and the ghosts bargain	^CH

book_id	n	tf	tfidf	source_file_path	title	
888	4.098543	0.007341	0.010750	Dickens/888-the_lazy_tour_of_two_idle_apprenti...	the lazy tour of two idle apprentices	CH
61522	4.577857	0.007410	0.010719	Twain/61522-the_1000000_bank_note.txt	the 1000000 bank note	^_TH

95 rows × 13 columns

Compare Distributions

In [76]:
```
# 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 [77]:
```
# 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 [78]:
```
DISTS.head(20).style.background_gradient(cmap='YlGnBu', high=.5, axis=0)
```

Out[78]:

doc_a	doc_b	label_a	label_b	corr_raw	cityblock_raw	euclidean_raw	euclidean_l2	cosine_raw	city
74.0	**70.0**	twain 74: the adventures of tom sawyer	twain 70: what is man	0.844095	3.571013	0.193548	0.953050	0.454152	0.8
76.0	**70.0**	twain 76: the adventures of huckleberry finn	twain 70: what is man	0.936474	3.937929	0.259376	1.121754	0.629166	1.(
86.0	**70.0**	twain 86: a connecticut yankee in king arthurs court	twain 70: what is man	0.773771	3.871357	0.259483	1.139019	0.648683	1.(

doc_a	doc_b	label_a	label_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	city
91.0	**70.0**	twain 91: tom sawyer abroad	twain 70: what is man	0.928792	3.664634	0.211988	0.980174	0.480371	0.7
93.0	**70.0**	twain 93: tom sawyer detective	twain 70: what is man	0.963467	4.005373	0.183762	0.965004	0.465616	0.8
98.0	**70.0**	dickens 98: a tale of two cities	twain 70: what is man	1.049178	4.799437	0.237051	1.038975	0.539734	0.9
102.0	**70.0**	twain 102: the tragedy of puddnhead wilson	twain 70: what is man	0.896696	4.135842	0.181786	0.922717	0.425703	0.8
119.0	**70.0**	twain 119: a tramp abroad	twain 70: what is man	0.739475	3.009917	0.150472	0.917769	0.421150	0.7
142.0	**70.0**	twain 142: the 30000 bequest and other stories	twain 70: what is man	0.766430	4.455942	0.331842	1.184606	0.701645	1.0
245.0	**70.0**	twain 245: life on the mississippi	twain 70: what is man	0.793997	5.677724	0.358178	1.188755	0.706570	1.1
564.0	**70.0**	dickens 564: the mystery of edwin drood	twain 70: what is man	1.079489	4.185885	0.197324	0.994248	0.494265	0.9
580.0	**70.0**	dickens 580: the pickwick papers	twain 70: what is man	1.115897	2.911195	0.142001	0.848816	0.360244	0.7
588.0	**70.0**	dickens 588: master humphreys clock	twain 70: what is man	1.086118	4.643432	0.243657	1.140053	0.649861	1.2
644.0	**70.0**	dickens 644: the haunted man and the ghosts bargain	twain 70: what is man	1.065740	2.779993	0.137146	0.831188	0.345437	0.6

doc_a	doc_b	label_a	label_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	city
650.0	**70.0**	dickens 650: pictures from italy	twain 70: what is man	0.965629	4.788468	0.250481	1.065405	0.567544	0.9
653.0	**70.0**	dickens 653: the chimes	twain 70: what is man	1.057439	4.748378	0.245228	1.148615	0.659658	1.(
675.0	**70.0**	dickens 675: american notes	twain 70: what is man	1.007192	4.428126	0.239206	1.040649	0.541476	1.(
676.0	**70.0**	dickens 676: the battle of life	twain 70: what is man	1.068210	3.300123	0.186275	0.929429	0.431920	0.7
699.0	**70.0**	dickens 699: a childs history of england	twain 70: what is man	0.925559	5.105826	0.268420	1.131675	0.640344	1.2
700.0	**70.0**	dickens 700: the old curiosity shop	twain 70: what is man	1.092769	4.469155	0.266474	1.139351	0.649060	1.(

Compare Z normalized distributions

In [79]:
```
ZPAIRS = (PAIRS - PAIRS.mean()) / PAIRS.std()
```

In [80]:
```
ZPAIRS
```

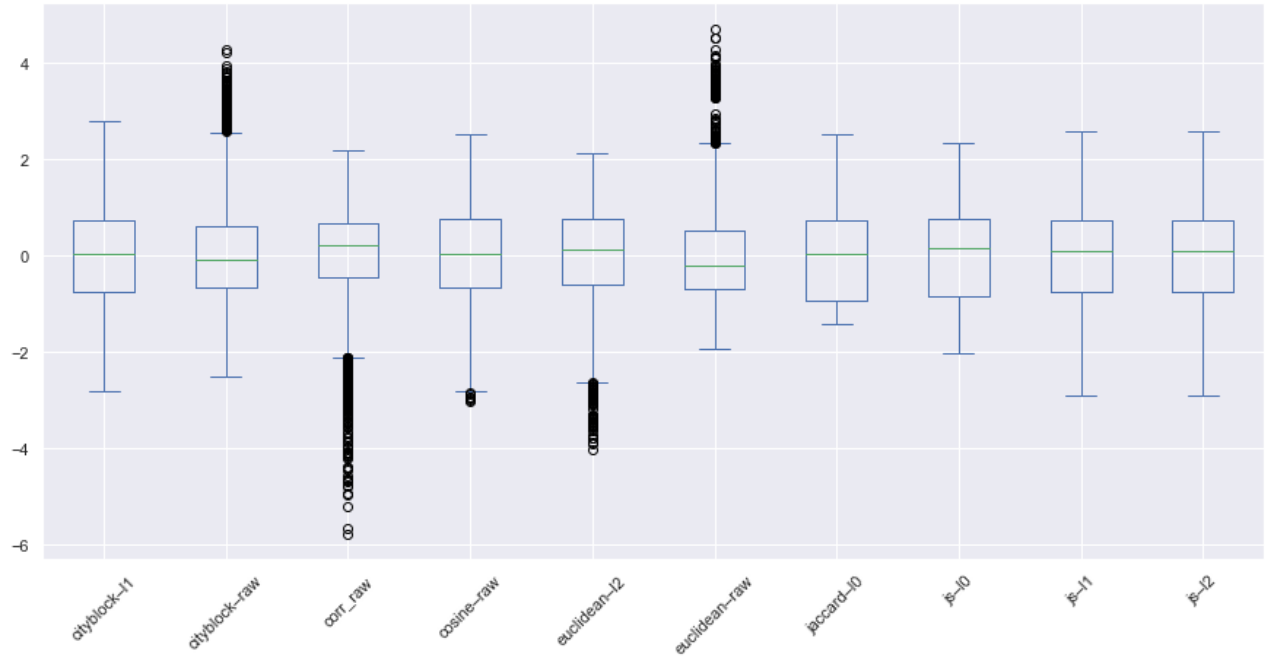
Out[80]:

doc_a	doc_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0
74	**70**	-0.515520	-0.835652	-0.617753	-0.442434	-0.524221	-0.745584	-1.144794
76	**70**	0.183248	-0.513548	0.097841	0.680280	0.669488	0.089873	-0.532739
	74	-2.931244	-1.582315	-1.180975	-1.193936	-1.214787	-1.391887	-1.273374
86	**70**	-1.047459	-0.571989	0.099009	0.795182	0.802606	0.244886	-0.000736
	74	-1.002766	-0.170574	0.284066	0.971627	1.010983	0.701563	0.393019
...	**...**	...	...	...	...	...	...	...
62739	**33077**	-0.817934	1.275295	0.522098	0.820098	0.831741	1.624017	1.914503
	35536	0.446262	0.230380	-0.123133	0.732090	0.729260	0.757307	0.554787

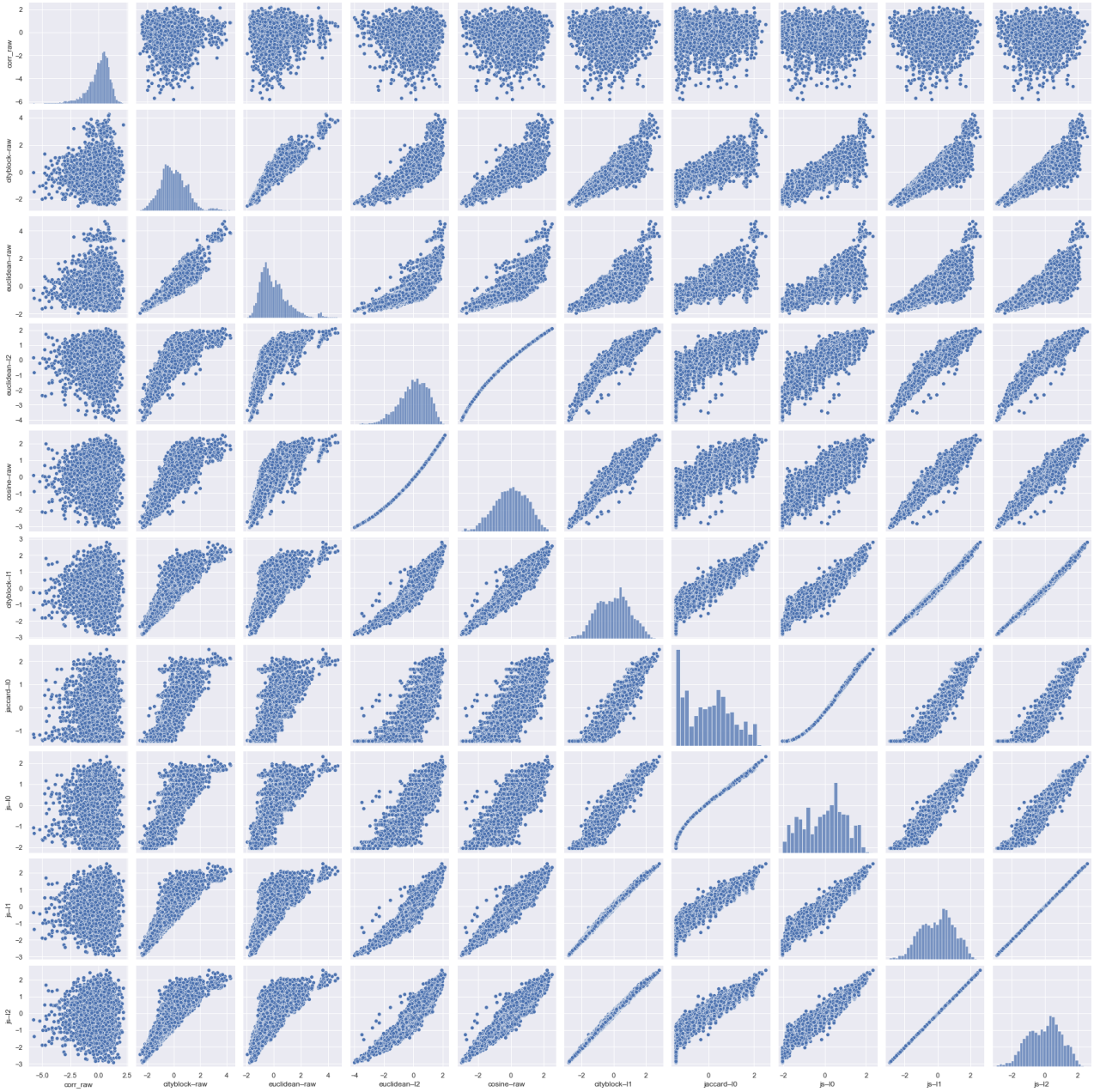
doc_a	doc_b	corr_raw	cityblock–raw	euclidean–raw	euclidean–l2	cosine–raw	cityblock–l1	jaccard–l0
	60900	0.443173	0.528037	0.104629	0.562216	0.534825	1.374205	1.885876
	61522	0.117250	-0.625254	-0.863275	0.216809	0.153182	0.534637	0.508427
	62636	-0.258709	-0.180451	0.221931	1.023905	1.073641	1.423334	1.823807

4465 rows × 10 columns

In [81]:
```
ZPAIRS.T.sort_index().T.plot.box(rot = 45, figsize = (15,7));
```



In [82]:
```
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 [83]:
```
n_clusters = 4
```

In [84]:
```
# 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 [217…
```
book_DOC.iloc[:,2:].sort_values('label').style.background_gradient(cmap = 'RdBu'
```

Out[217…

book_id	label	y_raw	y_L0	y_L1	y_L2
1023	dickens 1023: bleak house	3	1	0	0
1289	dickens 1289: three ghost stories	3	1	0	0
1394	dickens 1394: the holly tree	3	0	0	0
1400	dickens 1400: great expectations	3	1	0	0
1406	dickens 1406: the perils of certain english prisoners	0	0	0	0
1407	dickens 1407: a message from the sea	3	0	0	0
1413	dickens 1413: tom tiddlers ground	3	3	0	0
1414	dickens 1414: somebodys luggage	3	0	0	0
1415	dickens 1415: doctor marigold	2	2	2	1
1416	dickens 1416: mrs lirripers lodgings	1	0	1	3
1421	dickens 1421: mrs lirripers legacy	1	0	1	3
1435	dickens 1435: miscellaneous papers	0	1	0	1
1467	dickens 1467: some christmas stories	3	1	0	0
19337	dickens 19337: a christmas carol	3	1	0	0
20795	dickens 20795: the cricket on the hearth	3	1	0	0
2324	dickens 2324: a house to let	3	1	0	0
27924	dickens 27924: mugby junction	3	1	0	0
35536	dickens 35536: the poems and verses of charles dickens	0	3	0	1
564	dickens 564: the mystery of edwin drood	3	1	0	0
580	dickens 580: the pickwick papers	3	1	0	0
588	dickens 588: master humphreys clock	3	1	0	0
644	dickens 644: the haunted man and the ghosts bargain	3	1	0	0

book_id	label	y_raw	y_L0	y_L1	y_L2
650	dickens 650: pictures from italy	0	1	0	1
653	dickens 653: the chimes	3	1	0	0
675	dickens 675: american notes	0	1	0	0
676	dickens 676: the battle of life	3	1	0	0
699	dickens 699: a childs history of england	0	1	0	1
700	dickens 700: the old curiosity shop	3	1	0	0
730	dickens 730: oliver twist	3	1	0	0
766	dickens 766: david copperfield	3	1	0	0
786	dickens 786: hard times	3	1	0	0
807	dickens 807: hunted down	3	3	0	0
809	dickens 809: holiday romance	0	0	0	0
810	dickens 810: george silvermans explanation	3	3	0	0
821	dickens 821: dombey and sons	3	1	0	0
824	dickens 824: speeches of charles dickens	0	1	0	1
872	dickens 872: reprinted pieces	0	1	0	0
882	dickens 882: sketches by boz	0	1	0	0
883	dickens 883: our mutual friend	3	1	0	0
888	dickens 888: the lazy tour of two idle apprentices	0	1	0	0
912	dickens 912: the mudfog and other sketches	0	1	0	0
914	dickens 914: the uncommerical traveller	0	1	0	0
916	dickens 916: sketches of young couples	3	0	0	0
917	dickens 917: barnaby rudge	3	1	0	0
918	dickens 918: sketches of young gentlemen	3	0	0	3
922	dickens 922: sunday under three heads	0	3	0	1
927	dickens 927: the lamplighter	1	2	1	3
967	dickens 967: nicholas nickleby	3	1	0	0
968	dickens 968: martin chuzzlewit	3	1	0	0
98	dickens 98: a tale of two cities	3	1	0	0
102	twain 102: the tragedy of puddnhead wilson	0	1	0	1
1044	twain 1044: extract from captain stormfields visit to Heaven	0	0	1	2
1086	twain 1086: a horses tale	0	0	0	1
119	twain 119: a tramp abroad	0	1	0	1
142	twain 142: the 30000 bequest and other stories	0	1	0	1
1837	twain 1837: the prince and the pauper	0	1	0	1

book_id	label	y_raw	y_L0	y_L1	y_L2
19484	twain 19484: editorial wild oats	0	0	0	1
19987	twain 19987: chapters from my autobiography	0	1	0	1
245	twain 245: life on the mississippi	0	1	0	1
2874	twain 2874: personal recollections of joan of arc vol 1	0	1	0	1
2875	twain 2875: personal recollections of joan of arc vol 2	0	1	0	1
2895	twain 2895: following the equator	0	1	0	1
3171	twain 3171: in defense of harriet shelley	0	3	0	1
3172	twain 3172: fenimore coopers literary offences	0	3	0	1
3173	twain 3173: essays on paul bourget	0	3	0	1
3176	twain 3176: the innocents abroad	0	1	0	1
3177	twain 3177: roughing it	0	1	0	1
3178	twain 3178: the gilded age	0	1	0	1
3179	twain 3179: the american claimant	0	1	0	1
3180	twain 3180: a double barrelled detective story	0	0	0	1
3181	twain 3181: the stolen white elephant	0	2	0	1
3182	twain 3182: some rambling notes of an idle excursion	0	0	0	1
3183	twain 3183: the facts concerning the recent carnival of crime in connecticut	0	2	0	1
3184	twain 3184: alonzo fitz and other stories	0	1	0	1
3185	twain 3185: those extraordinary twins	0	0	0	1
3186	twain 3186: the mysterious stranger and other stories	0	1	0	1
3188	twain 3188: mark twain speeches	0	1	0	1
3189	twain 3189: sketches new and old	0	1	0	1
3190	twain 3190: 1601 conversation as it was by the social fireside in the time of the tudors	0	3	0	1
3191	twain 3191: goldsmiths friend abroad again	0	3	0	1
3192	twain 3192: the curious republic of gondour and other whimsical sketches	0	0	0	1
3199	twain 3199: the letters of mark twain	0	1	0	1
3250	twain 3250: how to tell a story and other essays	0	3	3	1
3251	twain 3251: the man that corrupted hadleyburg and other stories	0	1	0	1
33077	twain 33077: the treaty with china its provisions explained	0	3	0	1
60900	twain 60900: merry tales	0	1	0	1
61522	twain 61522: the 1000000 bank note	0	1	0	1
62636	twain 62636: to the person sitting in darkness	0	2	0	1

book_id	label	y_raw	y_L0	y_L1	y_L2
62739	twain 62739: king leopolds soliloquy	0	3	0	1
70	twain 70: what is man	0	1	0	1
74	twain 74: the adventures of tom sawyer	0	1	0	1
76	twain 76: the adventures of huckleberry finn	0	0	1	2
86	twain 86: a connecticut yankee in king arthurs court	0	1	0	1
91	twain 91: tom sawyer abroad	0	0	1	2
93	twain 93: tom sawyer detective	0	0	1	2

In [318…

```
# 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 [428…

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

Out[428…

	k	raw_silhouette_score	L0_silhouette_score	L1_silhouette_score	L2_silhouette_score
0	2.000000	0.322780	0.359987	0.312638	0.060449
1	3.000000	0.085965	0.323238	0.065156	0.080510
2	4.000000	0.078566	0.311382	0.321294	0.080087
3	5.000000	0.051705	0.324306	0.070481	0.043220
4	6.000000	0.100967	0.312791	0.078695	0.063616
5	7.000000	0.097534	0.308273	0.072923	0.063055
6	8.000000	0.041944	0.317302	0.078148	0.071406

	k	raw_silhouette_score	L0_silhouette_score	L1_silhouette_score	L2_silhouette_score
7	9.000000	0.090916	0.322025	0.038684	0.050611
8	10.000000	0.085972	0.323470	0.085550	0.044141

In [320…

```
# 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 [333…

```
# 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 [323…

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

In [430…

```
book_DOC[['label', 'author', 'type', max_col_name]].sort_values(max_col_name).st
```

Out[430…

book_id	label	author	type	max_y_L0
70	twain 70: what is man	twain	non-fiction	0
918	dickens 918: sketches of young gentlemen	dickens	stories	0
967	dickens 967: nicholas nickleby	dickens	novel	0
968	dickens 968: martin chuzzlewit	dickens	novel	0
1023	dickens 1023: bleak house	dickens	novel	0
1289	dickens 1289: three ghost stories	dickens	stories	0
1414	dickens 1414: somebodys luggage	dickens	stories	0
1435	dickens 1435: miscellaneous papers	dickens	non-fiction	0
1467	dickens 1467: some christmas stories	dickens	stories	0
1837	twain 1837: the prince and the pauper	twain	novel	0
2324	dickens 2324: a house to let	dickens	stories	0
2874	twain 2874: personal recollections of joan of arc vol 1	twain	non-fiction	0

book_id	label	author	type	max_y_L0
2875	twain 2875: personal recollections of joan of arc vol 2	twain	non-fiction	0
2895	twain 2895: following the equator	twain	non-fiction	0
3176	twain 3176: the innocents abroad	twain	non-fiction	0
3177	twain 3177: roughing it	twain	novel	0
3178	twain 3178: the gilded age	twain	novel	0
3179	twain 3179: the american claimant	twain	novel	0
61522	twain 61522: the 1000000 bank note	twain	stories	0
60900	twain 60900: merry tales	twain	stories	0
27924	dickens 27924: mugby junction	dickens	stories	0
20795	dickens 20795: the cricket on the hearth	dickens	novel	0
19987	twain 19987: chapters from my autobiography	twain	non-fiction	0
19337	dickens 19337: a christmas carol	dickens	novel	0
917	dickens 917: barnaby rudge	dickens	stories	0
3251	twain 3251: the man that corrupted hadleyburg and other stories	twain	stories	0
3189	twain 3189: sketches new and old	twain	stories	0
3188	twain 3188: mark twain speeches	twain	non-fiction	0
3186	twain 3186: the mysterious stranger and other stories	twain	stories	0
3185	twain 3185: those extraordinary twins	twain	stories	0
3184	twain 3184: alonzo fitz and other stories	twain	stories	0
3180	twain 3180: a double barrelled detective story	twain	stories	0
3199	twain 3199: the letters of mark twain	twain	non-fiction	0
916	dickens 916: sketches of young couples	dickens	stories	0
1400	dickens 1400: great expectations	dickens	novel	0
912	dickens 912: the mudfog and other sketches	dickens	stories	0
564	dickens 564: the mystery of edwin drood	dickens	novel	0
644	dickens 644: the haunted man and the ghosts bargain	dickens	stories	0
650	dickens 650: pictures from italy	dickens	non-fiction	0
653	dickens 653: the chimes	dickens	novel	0
675	dickens 675: american notes	dickens	non-fiction	0

book_id	label	author	type	max_y_L0
676	dickens 676: the battle of life	dickens	novel	0
914	dickens 914: the uncommerical traveller	dickens	non-fiction	0
245	twain 245: life on the mississippi	twain	non-fiction	0
142	twain 142: the 30000 bequest and other stories	twain	stories	0
699	dickens 699: a childs history of england	dickens	non-fiction	0
700	dickens 700: the old curiosity shop	dickens	novel	0
730	dickens 730: oliver twist	dickens	novel	0
766	dickens 766: david copperfield	dickens	novel	0
786	dickens 786: hard times	dickens	novel	0
580	dickens 580: the pickwick papers	dickens	novel	0
119	twain 119: a tramp abroad	twain	non-fiction	0
588	dickens 588: master humphreys clock	dickens	stories	0
98	dickens 98: a tale of two cities	dickens	novel	0
888	dickens 888: the lazy tour of two idle apprentices	dickens	stories	0
883	dickens 883: our mutual friend	dickens	novel	0
74	twain 74: the adventures of tom sawyer	twain	novel	0
76	twain 76: the adventures of huckleberry finn	twain	novel	0
882	dickens 882: sketches by boz	dickens	stories	0
872	dickens 872: reprinted pieces	dickens	stories	0
86	twain 86: a connecticut yankee in king arthurs court	twain	novel	0
821	dickens 821: dombey and sons	dickens	novel	0
102	twain 102: the tragedy of puddnhead wilson	twain	novel	0
824	dickens 824: speeches of charles dickens	dickens	non-fiction	0
3250	twain 3250: how to tell a story and other essays	twain	non-fiction	1
91	twain 91: tom sawyer abroad	twain	novel	1
93	twain 93: tom sawyer detective	twain	novel	1
19484	twain 19484: editorial wild oats	twain	stories	1
3190	twain 3190: 1601 conversation as it was by the social fireside in the time of the tudors	twain	stories	1
3191	twain 3191: goldsmiths friend abroad again	twain	stories	1

book_id	label	author	type	max_y_L0
3192	twain 3192: the curious republic of gondour and other whimsical sketches	twain	stories	1
35536	dickens 35536: the poems and verses of charles dickens	dickens	stories	1
33077	twain 33077: the treaty with china its provisions explained	twain	non-fiction	1
3173	twain 3173: essays on paul bourget	twain	non-fiction	1
3182	twain 3182: some rambling notes of an idle excursion	twain	non-fiction	1
922	dickens 922: sunday under three heads	dickens	non-fiction	1
927	dickens 927: the lamplighter	dickens	stories	1
1044	twain 1044: extract from captain stormfields visit to Heaven	twain	stories	1
1086	twain 1086: a horses tale	twain	novel	1
1394	dickens 1394: the holly tree	dickens	stories	1
62636	twain 62636: to the person sitting in darkness	twain	non-fiction	1
1406	dickens 1406: the perils of certain english prisoners	dickens	stories	1
1407	dickens 1407: a message from the sea	dickens	stories	1
3183	twain 3183: the facts concerning the recent carnival of crime in connecticut	twain	stories	1
1413	dickens 1413: tom tiddlers ground	dickens	stories	1
1415	dickens 1415: doctor marigold	dickens	stories	1
1416	dickens 1416: mrs lirripers lodgings	dickens	stories	1
1421	dickens 1421: mrs lirripers legacy	dickens	stories	1
809	dickens 809: holiday romance	dickens	stories	1
807	dickens 807: hunted down	dickens	stories	1
3171	twain 3171: in defense of harriet shelley	twain	non-fiction	1
3172	twain 3172: fenimore coopers literary offences	twain	non-fiction	1
3181	twain 3181: the stolen white elephant	twain	stories	1
810	dickens 810: george silvermans explanation	dickens	stories	1
62739	twain 62739: king leopolds soliloquy	twain	stories	1

In [335…
```
book_DOC.groupby(max_col_name).size()
```

Out[335…
```
max_y_L0
0    64
```

```
1     31
dtype: int64
```

In [336…
```
# cluster breakdown by author
book_DOC.groupby(['author', max_col_name]).size()
```

Out[336…
```
author   max_y_L0
dickens  0           37
         1           13
twain    0           27
         1           18
dtype: int64
```

In [337…
```
# cluster breakdown by type
book_DOC.groupby(['type', max_col_name]).size()
```

Out[337…
```
type         max_y_L0
non-fiction  0           16
             1            8
novel        0           25
             1            3
stories      0           23
             1           20
dtype: int64
```

In [338…
```
# cluster breakdown by author and type
book_DOC.groupby(['author', 'type', max_col_name]).size()
```

Out[338…
```
author   type         max_y_L0
dickens  non-fiction  0            6
                      1            1
         novel        0           17
         stories      0           14
                      1           12
twain    non-fiction  0           10
                      1            7
         novel        0            8
                      1            3
         stories      0            9
                      1            8
dtype: int64
```

M07: Features and Components

In [86]:
```
TFIDF_sigs
```

Out[86]:

	term_str	seem	sound	quiet	knows	pleasant	red	past	min
book_id	chap_id								
70	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	2	0.005280	0.000000	0.001056	0.005273	0.002114	0.002107	0.002107	0.00
	3	0.000000	0.000000	0.000000	0.020499	0.000000	0.000000	0.013650	0.000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000

	term_str	seem	sound	quiet	knows	pleasant	red	past	min
book_id	chap_id								
	5	0.012662	0.008441	0.004221	0.000000	0.004226	0.029474	0.004211	0.00
...	...	...	...	...	...	...	...	...	
62739	2	0.008019	0.004010	0.004010	0.016019	0.000000	0.000000	0.008000	0.00(
	3	0.045108	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
	4	0.005101	0.000000	0.000000	0.000000	0.000000	0.000000	0.005088	0.00(
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(

2290 rows × 1000 columns

In [166…

```
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.a

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

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

In [167…

```
chap_DOC
```

Out[167…

		author	title	type	decade	label	mean_tfidf	n_tokens
book_id	chap_id							
70	1	twain	what is man	non-fiction	1900	70-twain-what_is_man-1	0.000351	7(
	2	twain	what is man	non-fiction	1900	70-twain-what_is_man-2	0.000392	26698
	3	twain	what is man	non-fiction	1900	70-twain-what_is_man-3	0.000322	4367
	4	twain	what is man	non-fiction	1900	70-twain-what_is_man-4	0.000243	3669
	5	twain	what is man	non-fiction	1900	70-twain-what_is_man-5	0.000273	5465
...	...	...	...	...	...	...	...	..
62739	2	twain	king leopolds soliloquy	stories	1900	62739-twain-king_leopolds_soliloquy-2	0.000350	6390
	3	twain	king leopolds soliloquy	stories	1900	62739-twain-king_leopolds_soliloquy-3	0.000523	686

book_id	chap_id	author	title	type	decade	label	mean_tfidf	n_tokens
	4	twain	king leopolds soliloquy	stories	1900	62739-twain-king_leopolds_soliloquy-4	0.000277	3033
	5	twain	king leopolds soliloquy	stories	1900	62739-twain-king_leopolds_soliloquy-5	0.000226	1129
	6	twain	king leopolds soliloquy	stories	1900	62739-twain-king_leopolds_soliloquy-6	0.000286	356

2290 rows × 7 columns

In [168…
```
TFIDF_sigs
```

Out[168…

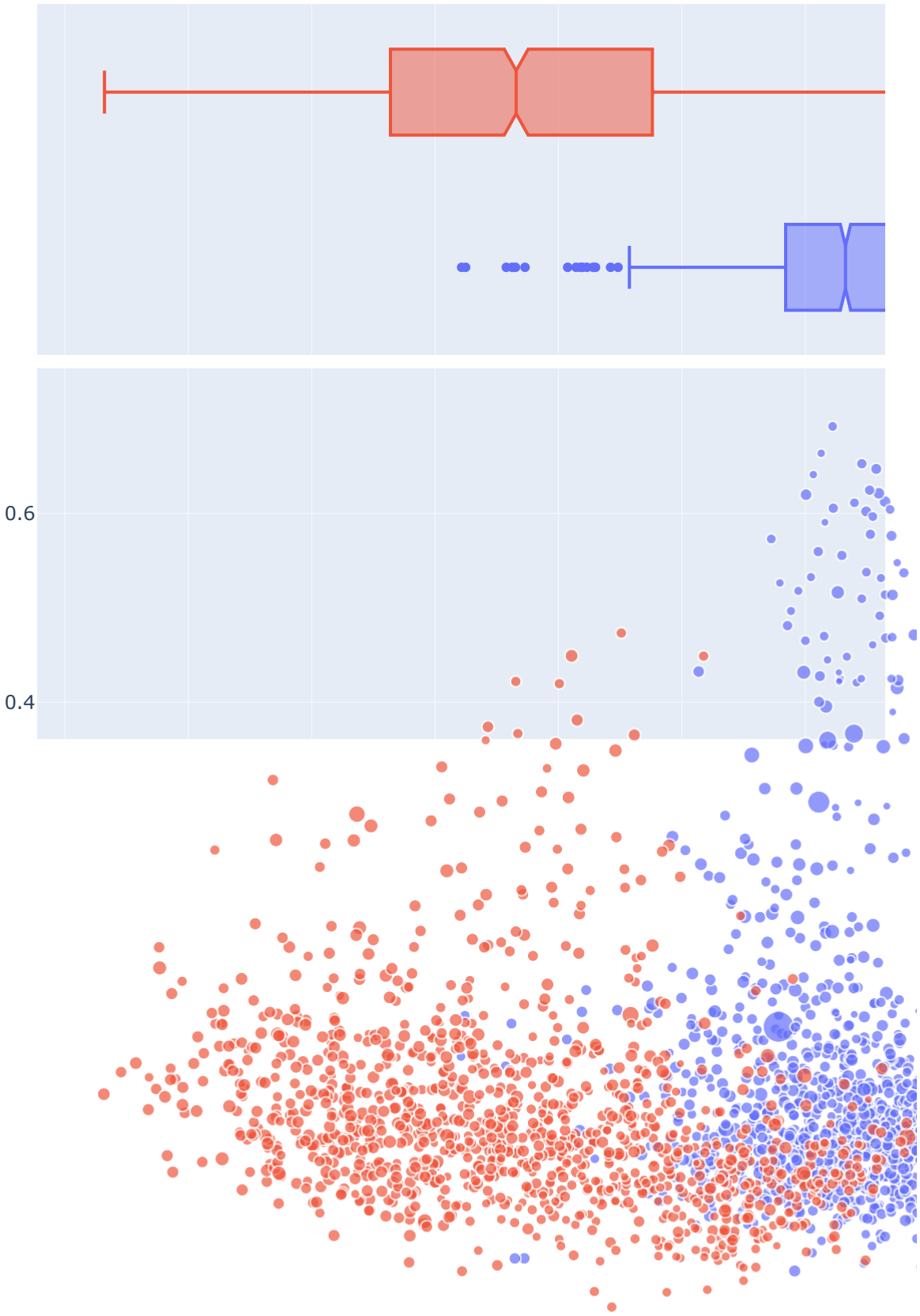
book_id	chap_id	term_str	seem	sound	quiet	knows	pleasant	red	past	min
70	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	2	0.005280	0.000000	0.001056	0.005273	0.002114	0.002107	0.002107	0.00
	3	0.000000	0.000000	0.000000	0.020499	0.000000	0.000000	0.013650	0.000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	5	0.012662	0.008441	0.004221	0.000000	0.004226	0.029474	0.004211	0.00
...	...	...	...	...	...	...	...	...	...	
62739	2	0.008019	0.004010	0.004010	0.016019	0.000000	0.000000	0.008000	0.000
	3	0.045108	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	4	0.005101	0.000000	0.000000	0.000000	0.000000	0.000000	0.005088	0.000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000

2290 rows × 1000 columns

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

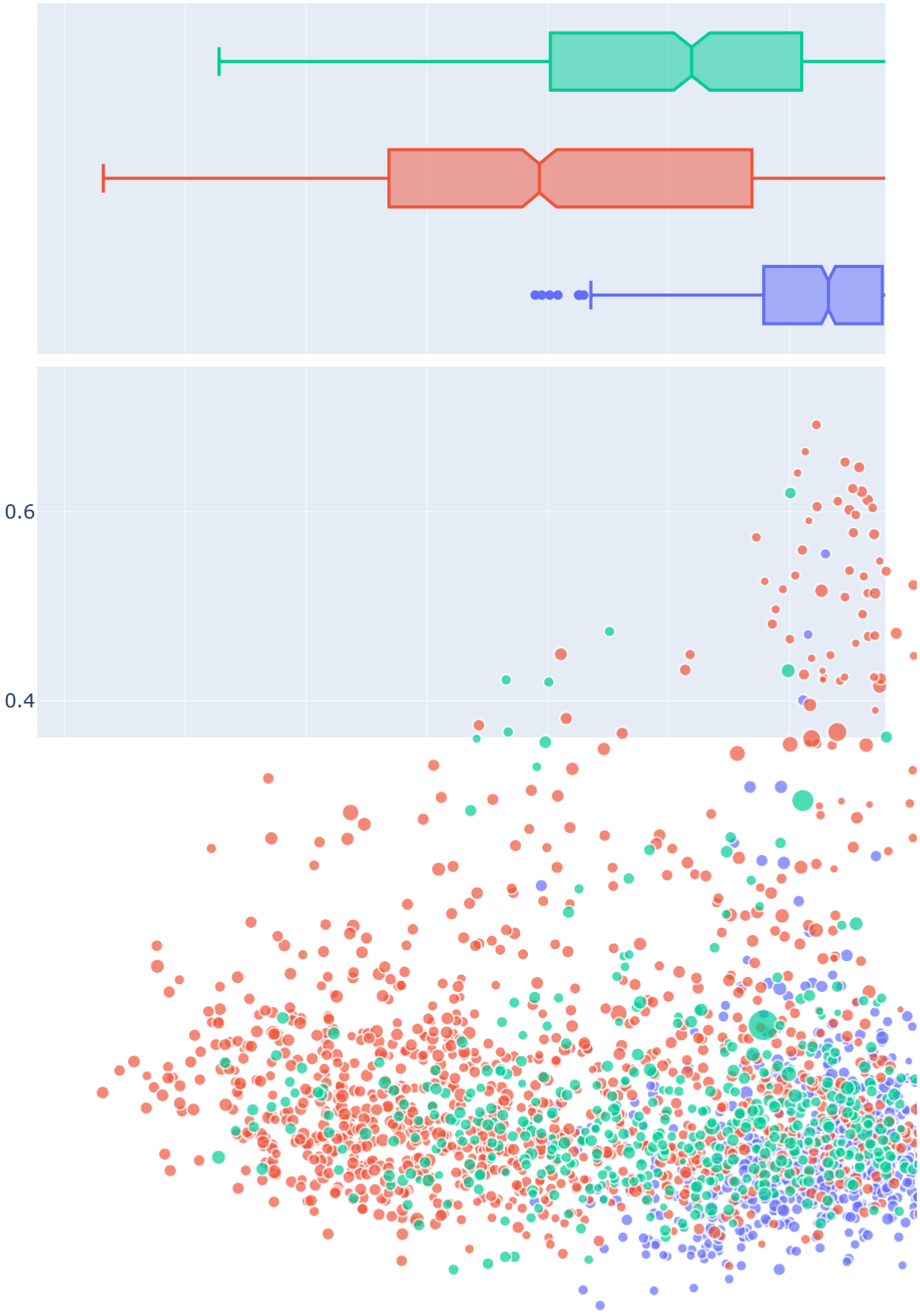
In [169…
```
LOADINGS, DCM, COMPINF = get_pca(TFIDF_sigs, norm_docs = True, center_by_mean =
```

In [170…
```
px.scatter(DCM, 0, 1, color=chap_DOC.author,
           size=np.abs(chap_DOC.mean_tfidf), hover_name=chap_DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```


0.6
0.4

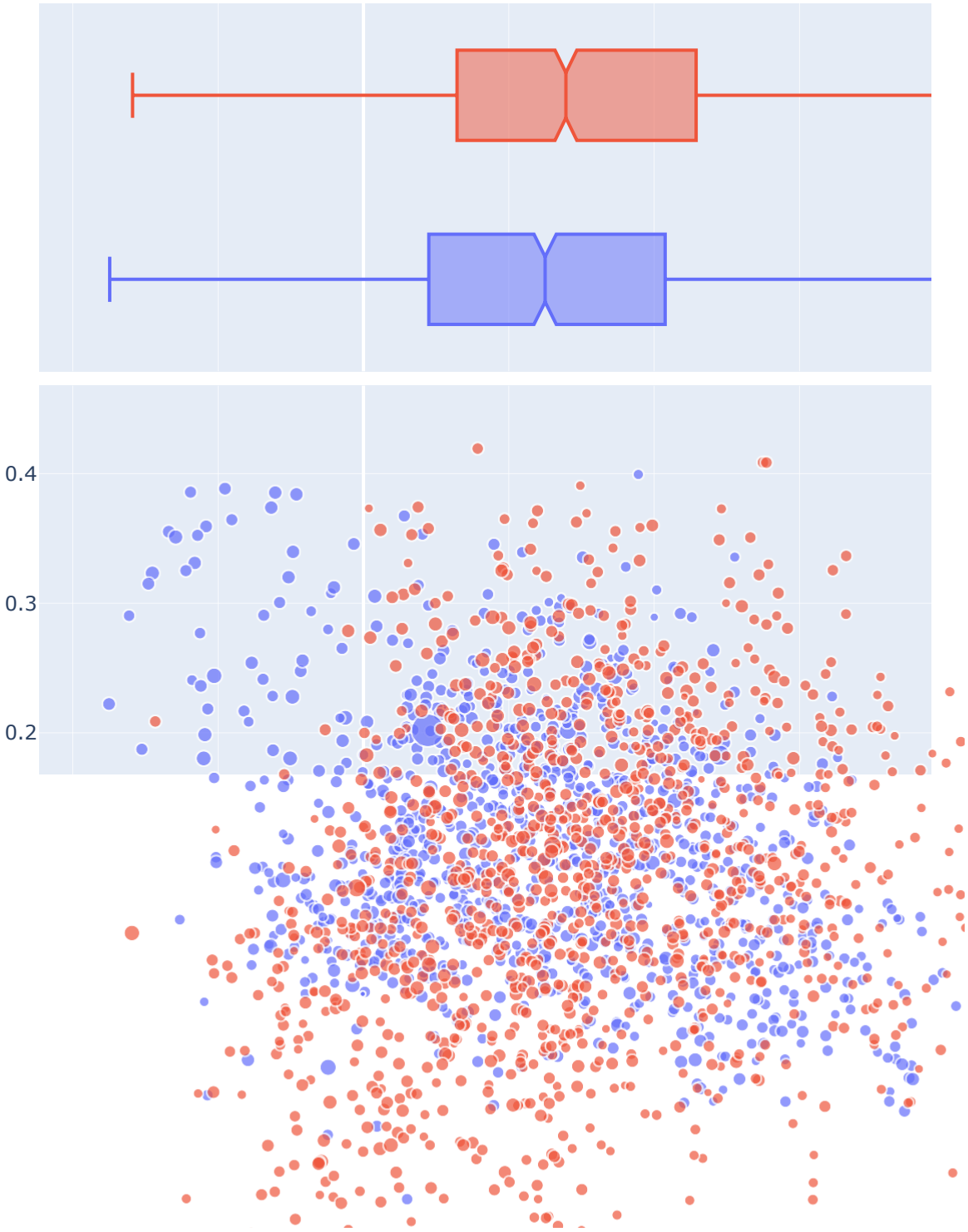
In [202…

```
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 [171…

```
px.scatter(DCM, 2, 3, color=chap_DOC.author,
           size=np.abs(chap_DOC.mean_tfidf), hover_name=chap_DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```



In [172…
```
X = LOADINGS.join(SIGS, how='inner').reset_index()
```

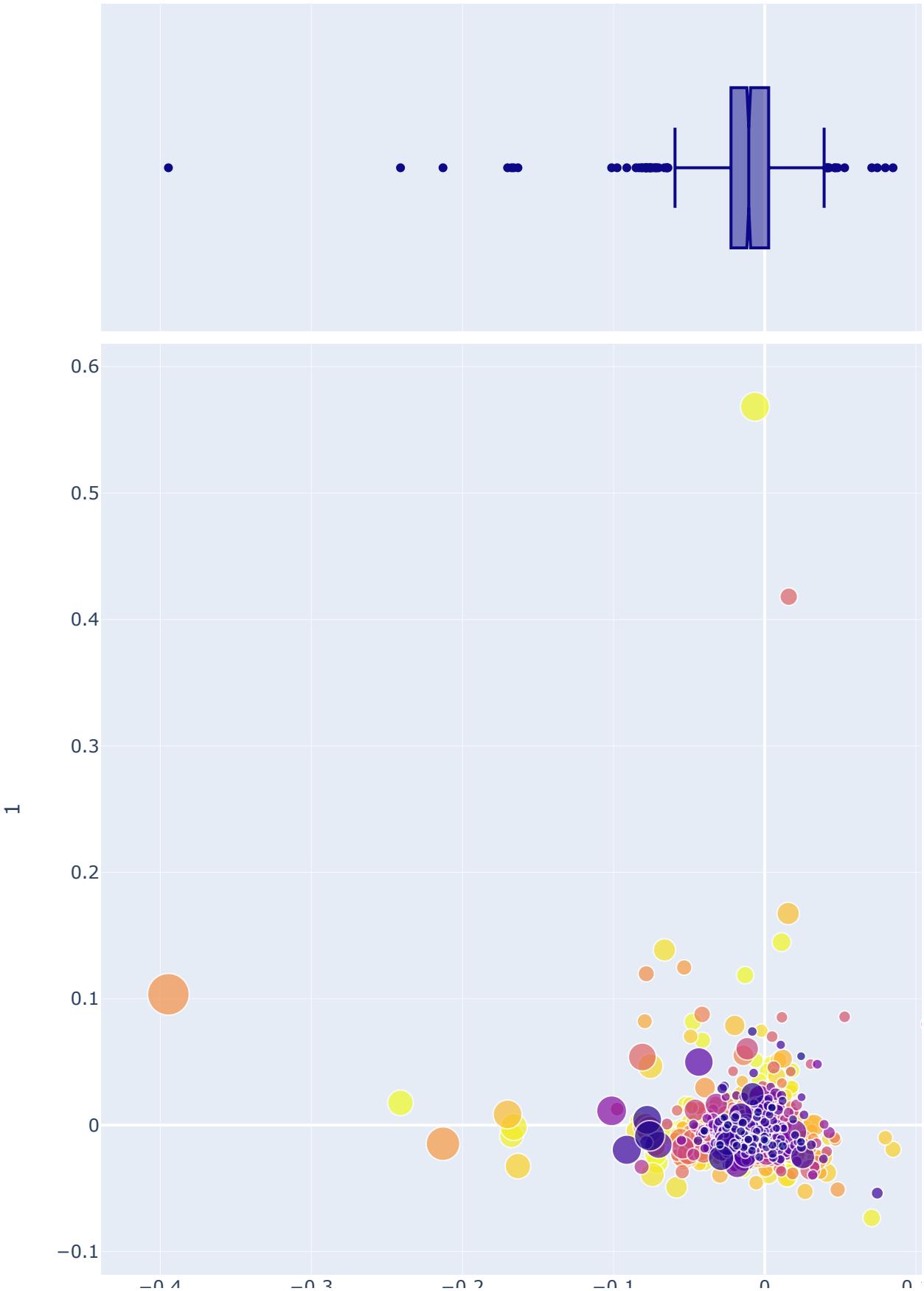
In [348…
```
X[['term_str', 'n']].sort_values('n', ascending = False).head(20)
```

Out[348…

	term_str	n
433	sir	10567
411	dear	6912
985	room	5633
762	door	5508
918	quite	5272
958	looking	5135
72	says	5120
309	gentleman	5064
851	yes	5052
540	oh	4743
895	having	4509
124	lady	4429
587	friend	4147
15	replied	4010
574	round	3995
235	returned	3985
977	heart	3964
813	morning	3891
230	boy	3847
795	work	3829

In [173…
```
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)
```



−0.4 −0.3 −0.2 −0.1 0 0.

0

In [174…
```
COMPINF
```

Out[174…

pc_id	pos	neg	eig_val	exp_var
0	river miles land war city	sir replied dear gentleman cried	0.028557	0.242161
1	says aint didnt couldnt thats	city war church sea love	0.017450	0.147974
2	river water feet sea horses	sir letter letters wrote book	0.013204	0.111970
3	dear letter child father love	sir gentleman gentlemen replied ladies	0.012453	0.105599
4	replied sir boys cried boy	says letter lady sea letters	0.009562	0.081081
5	lady says ladies gentleman child	sir letter river miles letters	0.009260	0.078523
6	replied letter coach boys bill	sir says father court child	0.008036	0.068145
7	river dear water miles bank	court boys church crowd judge	0.006797	0.057639
8	river war ladies lady gentlemen	letter letters wrote write coach	0.006495	0.055075
9	court judge returned gentlemen public	lady boys boy mother replied	0.006112	0.051832

Prince PCA Method with entire TFIDF

In [175…
```
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 [176…
```
pca = pca.fit(TFIDF)
```

In [177…
```
dcm = pca.transform(TFIDF)
```

In [178…
```
dcm
```

Out[178…

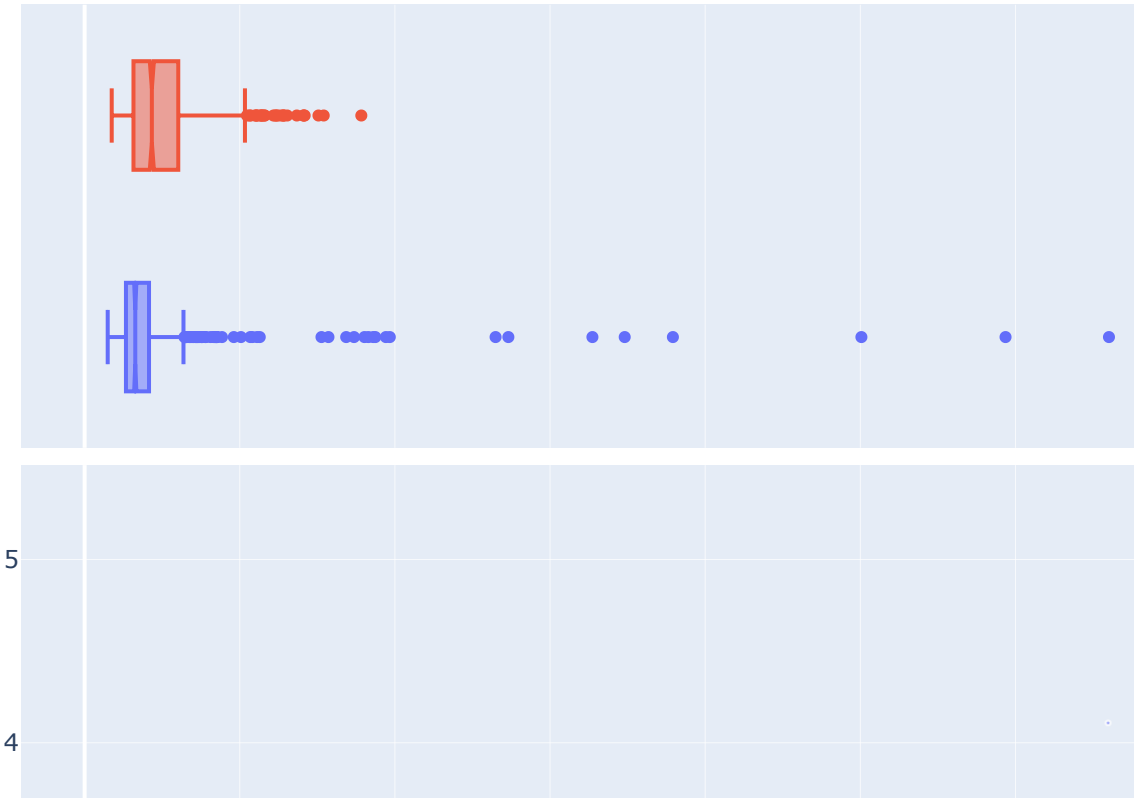
book_id	chap_id	0	1	2	3	4	5

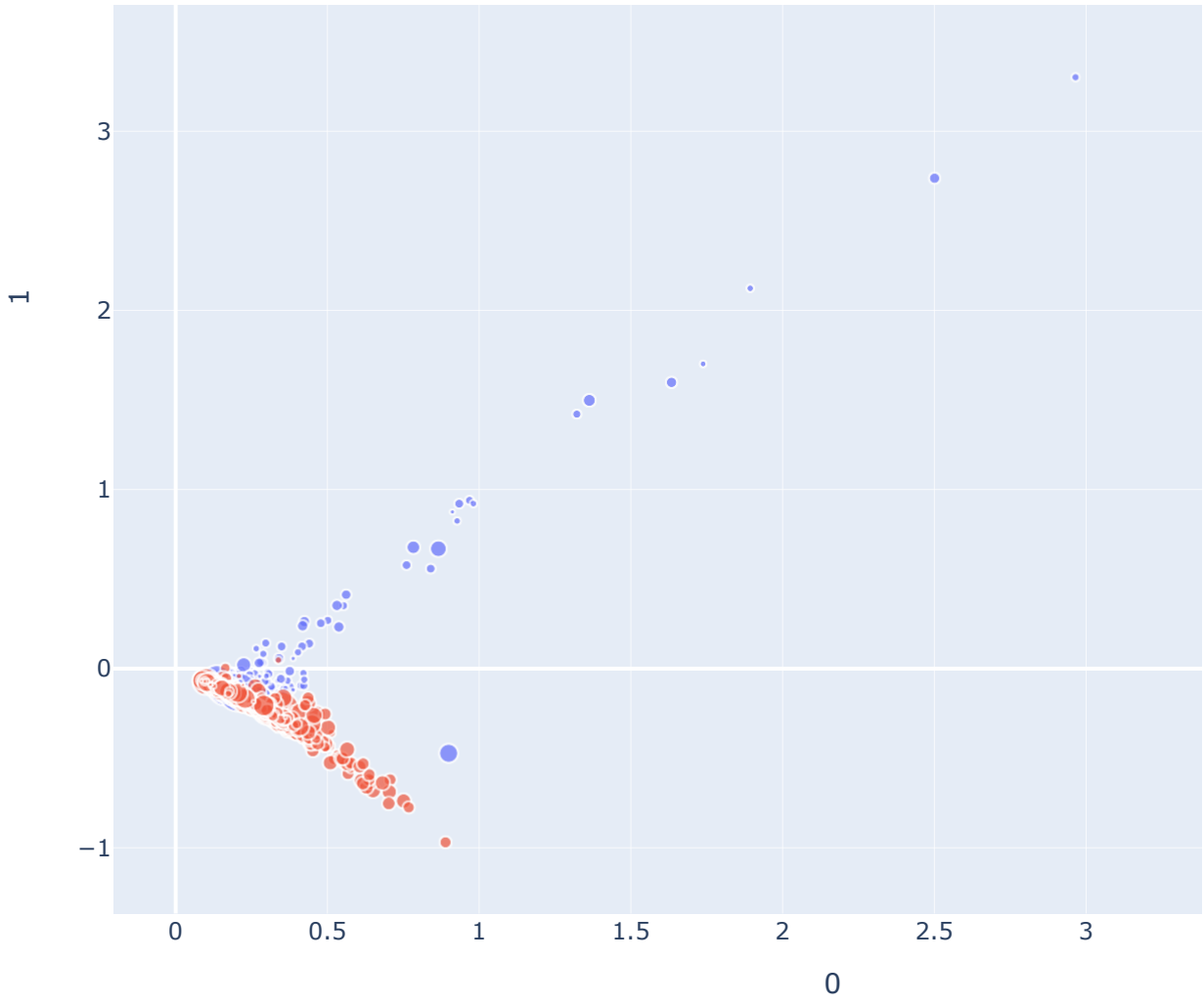
		0	1	2	3	4	5
book_id	**chap_id**						
70	**1**	0.172461	-0.122249	-0.039210	0.016044	-0.060932	0.081302
	2	0.164853	-0.115454	-0.040473	-0.008365	-0.075365	-0.092566
	3	0.213344	-0.145501	-0.050912	-0.013227	-0.061922	-0.071014
	4	0.111982	-0.075756	-0.024901	-0.007186	-0.036465	-0.036146
	5	0.135308	-0.089378	-0.035740	-0.002155	-0.049145	-0.051082
...	...	...	...	...	...	...	...
62739	**2**	0.145802	-0.092178	-0.038503	0.001615	-0.046724	-0.051259
	3	0.199191	-0.125980	-0.055431	-0.004438	-0.075567	-0.071980
	4	0.090000	-0.063336	-0.026424	0.004593	-0.034386	-0.032789
	5	0.099791	-0.074448	-0.020735	-0.007564	-0.036651	-0.022671
	6	0.117428	-0.091937	-0.026943	0.005288	-0.044062	-0.008128

2290 rows × 6 columns

In [179…

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





Prince PCA with Outliers Removed

- **(mostly due to distinct vernacular used by some characters in Dickens and (mostly) Twain novels)**

In [180…

```
dickens_books = LIB.loc[LIB.author.str.contains('dickens', case = False)].index.

twain_books = LIB.loc[LIB.author.str.contains('twain', case = False)].index.valu
```

In [181…

```
# 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 [182…

```
# upper fences for pc 0 for both authors
dickens_0_upper_fence = upper_fence(dcm, dickens_books, 0)
twain_0_upper_fence = upper_fence(dcm, twain_books, 0)

# upper fences for pc 1 for both authors
```

```
dickens_1_upper_fence = upper_fence(dcm, dickens_books, 1)
twain_1_upper_fence = upper_fence(dcm, twain_books, 1)
```

In [183…

```
# outliers the chapters in books with PC 0 or PC 1 greater than the max of the u
outliers = dcm.loc[(dcm[0] > max(dickens_0_upper_fence, twain_0_upper_fence)) |
```

In [184…

```
# known Twain outliers from experimentation (see twain_analysis_M7.ipynb)
twain_outliers = [(76, 4), (76, 8), (76, 15), (76, 23), (76, 38), (76, 43), (318
                  (102, 15), (102, 19), (142, 16), (3180, 10), (3250, 3), (60900
```

In [185…

```
# 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 [188…

```
# 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 [189…

```
# remove ~19% of VOCAB data

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

Out[189…

```
0.18917202211926468
```

In [190…

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

In [191…

```
# remove ~7% of data

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

Out[191…

```
0.06968396739677998
```

In [192…

```
# create new BOW with reduced CORPUS
small_BOW = create_bow(small_CORPUS, CHAPS)
```

In [203…

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

In [204…
```
# create new DTCM, TFIDF, DFIDF, and update BOW and VOCAB based on reduced VOCAB
small_DTCM, small_TFIDF, small_BOW, small_DFIDF, small_VOCAB = get_tfidf(small_B
```

In [205…
```
# DOC df
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

small_chap_DOC['mean_tfidf'] = small_TFIDF.mean(1)

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

In [210…
```
small_chap_DOC.head()
```

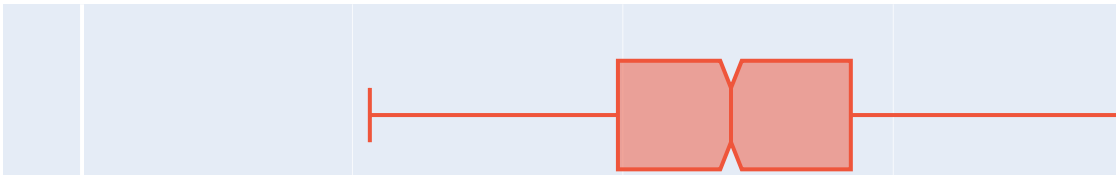
Out[210…

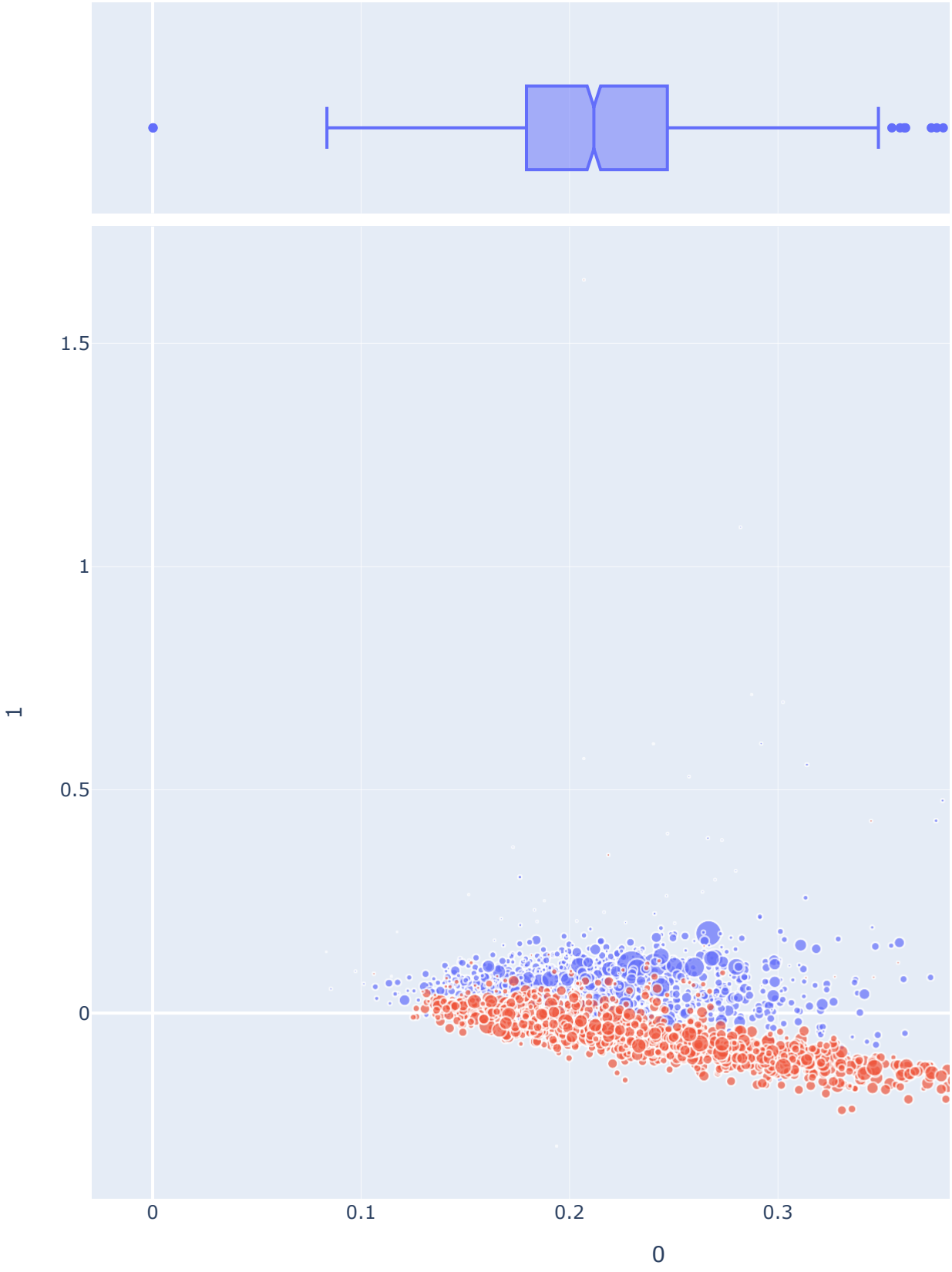
book_id	chap_id	author	title	type	decade	label	mean_tfidf	n_tokens
70	1	twain	what is man	non-fiction	1900	70-twain--what_is_man-1	0.000258	55
	2	twain	what is man	non-fiction	1900	70-twain--what_is_man-2	0.000471	26377
	3	twain	what is man	non-fiction	1900	70-twain--what_is_man-3	0.000328	4174
	4	twain	what is man	non-fiction	1900	70-twain--what_is_man-4	0.000270	3580
	5	twain	what is man	non-fiction	1900	70-twain--what_is_man-5	0.000287	5223

In [207…
```
small_pca = pca.fit(small_TFIDF)
```

In [208…
```
small_dcm = pca.transform(small_TFIDF)
```

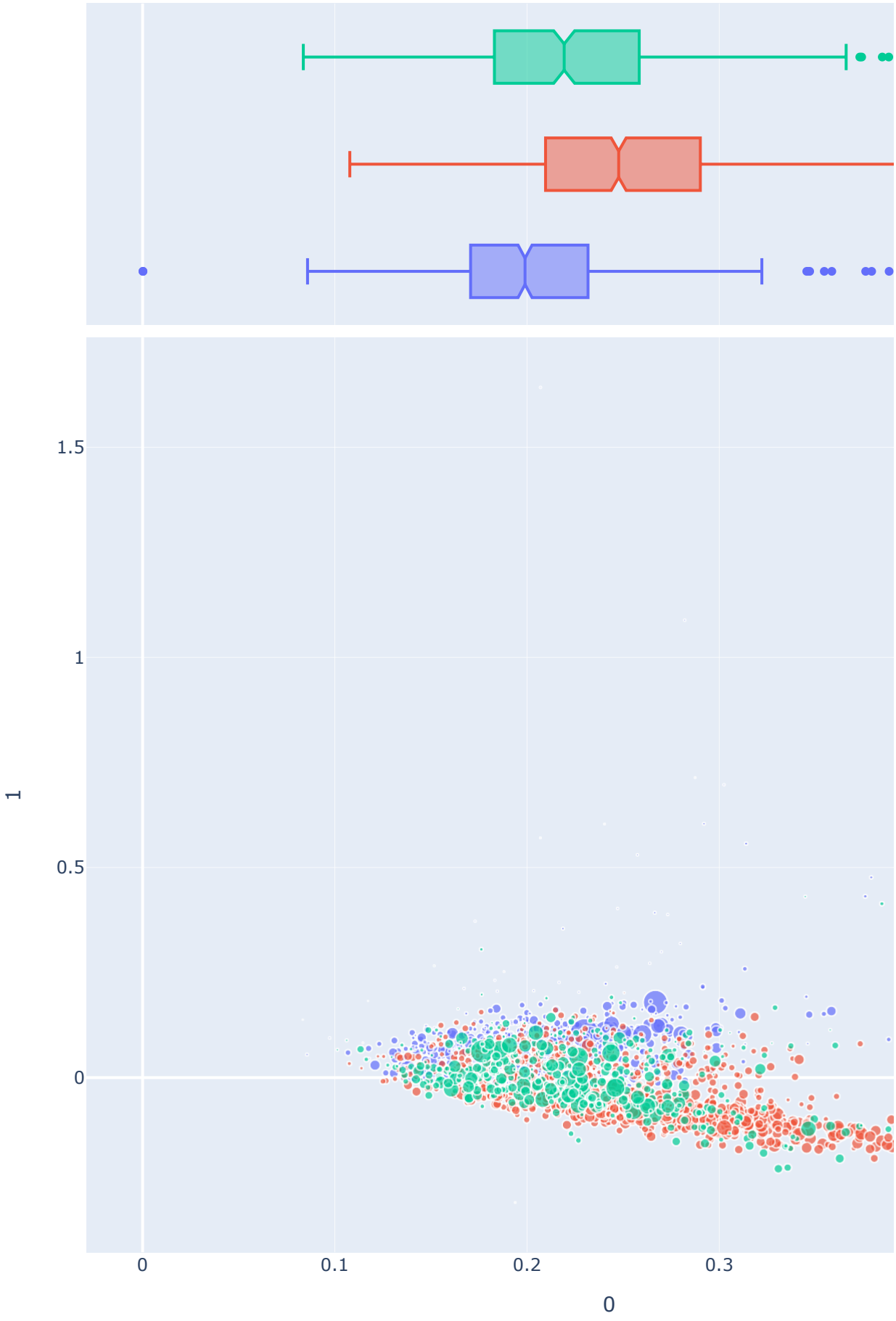
In [441…
```
px.scatter(small_dcm, 0, 1,
           color=small_chap_DOC.author,
           size=small_chap_DOC.n_tokens, hover_name=small_chap_DOC.label,
           height=1000, width=1200,
           marginal_x='box', marginal_y='box')
```





In [209…

```
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')
```


1.5
1
0.5
0
1
0
0.1
0.2
0.3
0

Prince PCA Method with 1000 most significant terms excluding proper nouns (`TFIDF_sigs`)

In [361…
```
pca_sigs = pca.fit(TFIDF_sigs)
```

In [362…
```
dcm_sigs = pca_sigs.transform(TFIDF_sigs)
```

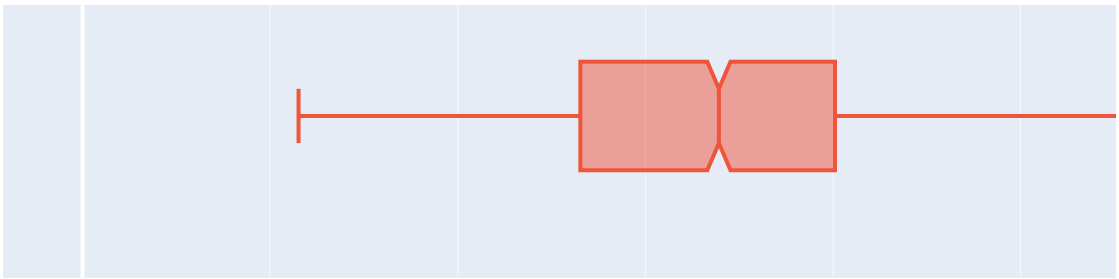
In [363…
```
dcm_sigs
```

Out[363…

		0	1	2	3	4	5
book_id	**chap_id**						
70	**1**	0.054950	0.018187	-0.024015	0.021493	0.005132	-0.020202
	2	0.134838	0.025184	-0.024721	0.034432	0.005296	0.013880
	3	0.162351	0.035561	-0.046020	0.068568	-0.008541	0.047222
	4	0.103860	0.036881	-0.028647	0.033381	0.011945	-0.012927
	5	0.111012	0.045823	-0.033744	0.039040	0.012074	-0.020692
…	**…**	…	…	…	…	…	…
62739	**2**	0.116934	0.030810	-0.012732	0.043372	-0.007458	-0.009305
	3	0.113717	0.026802	-0.010216	0.021238	0.017411	0.002403
	4	0.051398	0.015489	-0.016005	0.017383	0.011312	-0.000032
	5	0.065626	-0.003963	-0.014093	0.024007	0.023705	0.012245
	6	0.042407	-0.002607	-0.012446	0.016525	0.005939	-0.000810

2290 rows × 6 columns

In [442…
```
px.scatter(dcm_sigs, 0, 1,
           color=chap_DOC.author,
           size=chap_DOC.n_tokens, hover_name=chap_DOC.label,
           height=1000, width=1200,
           marginal_x='box', marginal_y='box')
```



In [364…

```
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')
```


0.3
0.2
0.1
0
−0.1
−0.2
1
0
0.05
0.1
0.15
0.2
0.25
0

In [359…
```
BOW.to_csv('full_BOW.csv')

VOCAB.to_csv('full_VOCAB.csv')
```

In [360…
```
VOCAB.columns
```

Out[360…
```
Index(['term_rank', 'n', 'n_chars', 'p', 'i', 'max_pos', 'n_pos', 'cat_pos',
       'stop', 'stem_porter', 'stem_snowball', 'stem_lancaster', 'term_rank2',
       'zipf_k', 'zipf_k2', 'tfidf_mean_chap_max', 'tfidf_max_chap_max', 'df',
       'idf', 'dfidf'],
      dtype='object')
```

In [ ]:
```

```