M09 Homework

Michael Vaden, mtv2eva

In [ ]:
```
import pandas as pd
import numpy as np
from glob import glob
import re
import nltk
import plotly_express as px
from lib.textparser import TextParser
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer
from sklearn.decomposition import LatentDirichletAllocation as LDA
from numpy.linalg import norm
from scipy.spatial.distance import pdist
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
from scipy.linalg import eigh
from sklearn.decomposition import PCA
import seaborn as sns

from gensim.models import word2vec
from gensim.corpora import Dictionary
from sklearn.manifold import TSNE
import gensim
gensim.__version__
```

Out[ ]:
```
'4.3.0'
```

In [ ]:
```
import configparser
config = configparser.ConfigParser()
config.read("../env.ini")
data_home = config['DEFAULT']['data_home']
output_dir = config['DEFAULT']['output_dir']
data_prefix = 'austen-melville'
```

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

PARA = OHCO[:4] # Paragraphs
SENT = OHCO[:5] # Sentences
BAG = PARA
```

In [ ]:
```
LIB = pd.read_csv(f"{output_dir}/{data_prefix}-LIB.csv").set_index('book_id')
CORPUS = pd.read_csv(f"{output_dir}/{data_prefix}-CORPUS.csv").set_index(OHCO)
VOCAB = pd.read_csv(f'{output_dir}/{data_prefix}-VOCAB.csv').set_index('term_str').dropna()
```

In [ ]:
```
auths = LIB['author']
```

In [ ]:
```
parts_of_speech = ['NN', 'NNS', 'VB']

austen_corp = CORPUS.join(auths).query("author == 'AUSTEN, JANE'").query("pos_group in @parts_of_speech").drop('author', axis=1)
melville_corp = CORPUS.join(auths).query("author == 'MELVILLE, HERMAN'").query("pos_group in @parts_of_speech").drop('author', axis=1)
```

In [ ]:
```
austen_corp
```

Out[ ]:

					pos_tuple	pos	token_str	term_str	pos_group
book_id	chap_id	para_num	sent_num	token_num					
105	1	1	0	0	('Sir', 'NNP')	NNP	Sir	sir	NN
				1	('Walter', 'NNP')	NNP	Walter	walter	NN
				2	('Elliot,', 'NNP')	NNP	Elliot,	elliot	NN
				4	('Kellynch', 'NNP')	NNP	Kellynch	kellynch	NN
				5	('Hall,', 'NNP')	NNP	Hall,	hall	NN
...	...	...	...	...	...	...	...	...	...
1342	61	18	0	5	('EBook', 'NNP')	NNP	EBook	ebook	NN
				7	('Pride', 'NNP')	NNP	Pride	pride	NN
				9	('Prejudice,', 'NNP')	NNP	Prejudice,	prejudice	NN
				11	('Jane', 'NNP')	NNP	Jane	jane	NN
				12	('Austen', 'NNP')	NNP	Austen	austen	NN

336048 rows × 5 columns

In [ ]:
```
docs_austen = austen_corp.dropna(subset='term_str')\
    .groupby(BAG)\
    .term_str.apply(lambda  x:  x.tolist())\
    .reset_index()['term_str'].tolist()
docs_austen = [doc for doc in docs_austen if len(doc) > 1] # Lose single word docs
```

In [ ]:
```
docs_melville = melville_corp.dropna(subset='term_str')\
    .groupby(BAG)\
    .term_str.apply(lambda  x:  x.tolist())\
    .reset_index()['term_str'].tolist()
docs_melville = [doc for doc in docs_melville if len(doc) > 1] # Lose single word docs
```

In [ ]:
```
w2v_params_austen = dict(
    window = 2,
    vector_size = 256,
    min_count = 50
)

w2v_params_melville = dict(
    window = 2,
    vector_size = 256,
    min_count = 80
)
```

In [ ]:
```
austen_vocab = austen_corp.reset_index().groupby('term_str').count()['pos'].to_frame()\
    .join(austen_corp.reset_index()[['term_str', 'pos_group']].drop_duplicates('term_str').set_index('term_str'), how='left')\
    .rename({'pos':'count'}, axis=1).drop_duplicates()

austen_vocab = austen_vocab[~austen_vocab.index.isna()]

austen_vocab.head()
```

Out[ ]:

term_str	count	pos_group
10th	1	NN
7th	2	NN
a	55	NN
abandoned	5	VB
abashed	2	VB

In [ ]:
```
melville_vocab = melville_corp.reset_index().groupby('term_str').count()['pos'].to_frame()\
    .join(melville_corp.reset_index()[['term_str', 'pos_group']].drop_duplicates('term_str').set_index('term_str'))\
    .rename({'pos':'count'}, axis=1).drop_duplicates()

melville_vocab = melville_vocab[~melville_vocab.index.isna()]

melville_vocab.head()
```

Out[ ]:

term_str	count	pos_group
1	6	NN
1684	1	NN
1775	1	VB
200000	2	NN
a	269	NN

word vectors

In [ ]:
```
model_austen = word2vec.Word2Vec(docs_austen, **w2v_params_austen)

model_austen.wv.vectors
```

Out[ ]:
```
array([[ 0.07827917, -0.10398872,  0.20390862, ..., -0.20217916,
        -0.22253633,  0.03150739],
       [-0.11350518, -0.18718411,  0.05608533, ..., -0.04148346,
        -0.26016787,  0.01420193],
       [-0.2291056 , -0.5653246 ,  0.21682751, ...,  0.09361234,
        -0.57990235,  0.1992455 ],
       ...,
       [-0.04918791, -0.0861868 ,  0.04615204, ...,  0.01521278,
        -0.16973397,  0.04257454],
       [-0.02975562, -0.08872374,  0.05235496, ..., -0.00756529,
        -0.16266184,  0.02956098],
       [-0.01829399, -0.09269781,  0.06108769, ..., -0.01247484,
        -0.17399617,  0.03409144]], dtype=float32)
```

In [ ]:
```
model_melville = word2vec.Word2Vec(docs_melville, **w2v_params_melville)

model_melville.wv.vectors
```

Out[ ]:
```
array([[-0.24609177, -0.03790091, -0.13768338, ..., -0.01848806,
        -0.19537704,  0.01801555],
       [-0.10476958, -0.15555055, -0.3334349 , ...,  0.09076785,
        -0.21436925,  0.16535182],
       [-0.09547845, -0.12997505, -0.16424717, ...,  0.12734936,
        -0.09287709,  0.1522856 ],
       ...,
       [-0.0435376 , -0.00857505, -0.03265905, ..., -0.00428493,
        -0.12678473, -0.05725246],
       [-0.05611232, -0.00533343, -0.02286693, ..., -0.0218174 ,
        -0.12438182, -0.07919938],
       [-0.02508631, -0.02382843,  0.002317  , ..., -0.05587981,
        -0.17159   , -0.07102297]], dtype=float32)
```

TSNE

In [ ]:
```
def get_vector_austen(row):
    w = row.name
    try:
```

```
        vec = model_austen.wv[w]
    except KeyError as e:
        vec = None
    return vec

def get_vector_melville(row):
    w = row.name
    try:
        vec = model_melville.wv[w]
    except KeyError as e:
        vec = None
    return vec
```

In [ ]:
```
WV_austen = pd.DataFrame(austen_vocab.apply(get_vector_austen, axis=1).dropna()).apply(lambda x: pd.Series(x[0]), axis=1).drop_duplicates()
```

In [ ]:
```
WV_melville = pd.DataFrame(melville_vocab.apply(get_vector_melville, axis=1).dropna()).apply(lambda x: pd.Series(x[0]), axis=1).drop_duplicate
```

In [ ]:
```
WV_melville
```

Out[ ]:

term_str	0	1	2	3	4	5	6	7	8	9	...	246	247	248	249	
a	-0.146529	0.086254	-0.243910	0.063628	0.352372	0.223228	0.066664	0.119554	-0.101264	0.140265	...	-0.093191	0.009396	0.110530	-0.240099	
about	-0.027666	-0.013620	0.010504	0.112983	0.187049	0.118483	0.061567	0.109999	-0.098988	0.073143	...	0.076625	0.067283	-0.012364	-0.174056	0
above	-0.034515	-0.005458	0.009067	0.103626	0.167970	0.086237	0.084670	0.116462	-0.077424	0.058242	...	0.057450	0.029590	0.003445	-0.178305	-(
according	-0.042916	-0.018019	-0.093173	0.037530	0.227288	0.025400	0.182504	0.067811	-0.024109	0.098406	...	0.023913	-0.005879	0.011895	-0.162835	0
account	-0.041965	-0.014134	-0.117297	0.048050	0.223433	0.053234	0.154047	0.017332	-0.020738	0.091966	...	0.029124	0.017220	-0.026896	-0.170878	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
work	-0.012746	-0.028901	-0.046387	0.087149	0.190285	0.042817	0.155109	0.075591	-0.038559	0.092212	...	0.062438	0.042911	-0.018241	-0.135657	0
world	0.104784	-0.063245	-0.067239	-0.088767	0.267179	0.021023	0.221022	-0.062271	0.001722	0.123710	...	0.113195	0.006624	-0.122112	-0.108535	0
years	-0.013782	0.038987	-0.038692	-0.023547	0.165969	-0.026383	0.198183	0.076300	0.042367	0.107385	...	-0.004775	-0.095519	0.025517	-0.171548	0
yes	-0.068439	0.028329	-0.074308	0.117246	0.192531	0.126519	0.000323	-0.225193	-0.030196	0.073072	...	0.040516	0.093257	-0.198773	-0.295646	0
you	0.034576	0.066105	-0.072507	0.046274	0.257479	0.184376	0.016045	-0.341328	-0.043264	0.126729	...	0.182325	0.097987	-0.315119	-0.293871	0

504 rows × 256 columns

In [ ]:
```
tsne_engine_austen = TSNE(learning_rate = 200, perplexity=20, n_components=2, init='random', n_iter=1000, random_state=42)
tsne_model_austen = tsne_engine_austen.fit_transform(WV_austen.to_numpy())
TSNE_austen = pd.DataFrame(tsne_model_austen, columns=['x','y'], index=WV_austen.index)
TSNE_austen
```

Out[ ]:

term_str	x	y
a	-7.925046	-1.120482
abbey	5.332089	12.729301
about	-1.808287	-17.039213
absence	7.135981	15.910089
accept	3.466157	-11.719110
...	...	...
world	5.813323	-30.655991
write	-9.048848	-24.139776
written	19.589153	-7.242359
years	19.031887	22.371311
you	-8.780233	-37.623138

482 rows × 2 columns

In [ ]:
```
tsne_engine_melville = TSNE(learning_rate = 200, perplexity=20, n_components=2, init='random', n_iter=1000, random_state=42)
tsne_model_melville = tsne_engine_melville.fit_transform(WV_melville.to_numpy())
TSNE_melville = pd.DataFrame(tsne_model_melville, columns=['x','y'], index=WV_melville.index)
TSNE_melville
```

Out[ ]:

term_str	x	y
a	-5.787805	-0.576860
about	6.426618	-10.447926
above	-4.889963	-5.511650
according	-10.819836	10.323805
account	1.866584	16.106071
...	...	...
work	-5.185720	7.624615
world	4.048825	27.663759
years	-9.094588	22.480089
yes	43.125992	17.294096
you	42.668396	13.518506

504 rows × 2 columns

In [ ]:
```
X_austen = TSNE_austen.join(austen_vocab, how = 'left')
X_melville = TSNE_melville.join(melville_vocab, how = 'left')
```

1.Identify two regions of word clusters in the Austen plot that clearly contain words with associated meanings. Give a gloss of what you think these clusters "mean."

In [ ]:
```
px.scatter(X_austen.reset_index(), 'x', 'y',
           text='term_str',
           color='pos_group',
           hover_name='term_str',
           size='count',
           height=1000).update_traces(
                mode='markers+text',
                textfont=dict(color='black', size=14, family='Arial'),
                textposition='top center')
```

From looking at the word clusters shown above in the Austen TSNE coordinates, we can see various different groups of words with associated meanings. One of the most clear clusters is of the verbs *had, have, has, having, and being*. Each of these words relates to possession or a state of being. They are all very common with high counts within the Austen collection. This cluster means that these verbs are all used very similarly to convey similar meanings. Another cluster of words that is interesting in the Austen plot is the cluster of nouns that include *it, thing, something, anything, nothing, people, world, and love (verb)*. Each of these words is either ambiguous in nature or relates to very broad themes, which in the context of the cluster means that these words are often associated with similar overarching themes or uncertainty.

2.Identify two regions of word clusters in the Melville plot that clearly contain words with associated meanings. Give a gloss of what you think these clusters "mean."

In [ ]:
```
px.scatter(X_melville.reset_index(), 'x', 'y',
           text='term_str',
           color='pos_group',
           hover_name='term_str',
           size='count',
           height=1000).update_traces(
                mode='markers+text',
                textfont=dict(color='black', size=14, family='Arial'),
                textposition='top center')
```

Looking at the Melville TNSE coordinates, one immediate cluster that stood out to me is that of *boat, boats, sail, sails, ground, shore, distance, etc.* Other words that are close to this cluster include *water, feet, deck, mast, bay, house, sea, and ships*. We see a very distinct nautical theme within this larger cluster of mostly nouns, all relating to the idea of traveling by water. Along with the sailing-themed words, we see words such as ground, shore, and distance which reference the search for land or shore that often drive and determine nautical expeditions. Another cluster of words that I found interesting which stood out from the coordinates was the group of *been, seen, years, taken, given, and gone*. These words all have a distinct theme of referencing or describing the past, often ominously. This cluster makes sense potentially within Melville's narrative structure as the past is referenced.

3.Based on your inspection of the results, come up with two analogies from the Austen model using 'man' and 'woman' as the A and C terms of the analogy. Describe your results.

In [ ]:
```
def complete_analogy_austen(A, B, C, n=2):
    try:
        cols = ['term', 'sim']
        return pd.DataFrame(model_austen.wv.most_similar(positive=[B, C], negative=[A])[0:n], columns=cols)
    except KeyError as e:
        print('Error:', e)
        return None
```

In [ ]:
```
complete_analogy_austen('man', 'brother', 'woman', 3)
```

Out[ ]:

	term	sim
0	attend	0.976239
1	smiles	0.974439
2	telling	0.973853

In [ ]:
```
complete_analogy_austen('man', 'father', 'woman', 3)
```

Out[ ]:

	term	sim
0	law	0.975001
1	receiving	0.972316
2	fortnight	0.972052

Unfortunately it appears for our Austen analogies that they are pretty nonsensical. When inputing brother as the B, we might expect our new term for woman to be 'sister'. However, the words that we get appear, although potentially used in familial interactions, are not as relevant. Similarly, when using father as our B in the algebra equation, the terms that we get do not fit the family relations we might expect, despite the higher similarity score.

4.Do the same thing with Melville.

In [ ]:
```
def complete_analogy_melville(A, B, C, n=2):
    try:
        cols = ['term', 'sim']
        return pd.DataFrame(model_melville.wv.most_similar(positive=[B, C], negative=[A])[0:n], columns=cols)
    except KeyError as e:
        print('Error:', e)
        return None
```

In [ ]:
```
complete_analogy_melville('man', 'boys', 'woman', 3)
```

Out[ ]:

	term	sim
0	began	0.934677
1	break	0.929921
2	beat	0.926378

In [ ]:
```
complete_analogy_melville('man', 'ships', 'woman', 3)
```

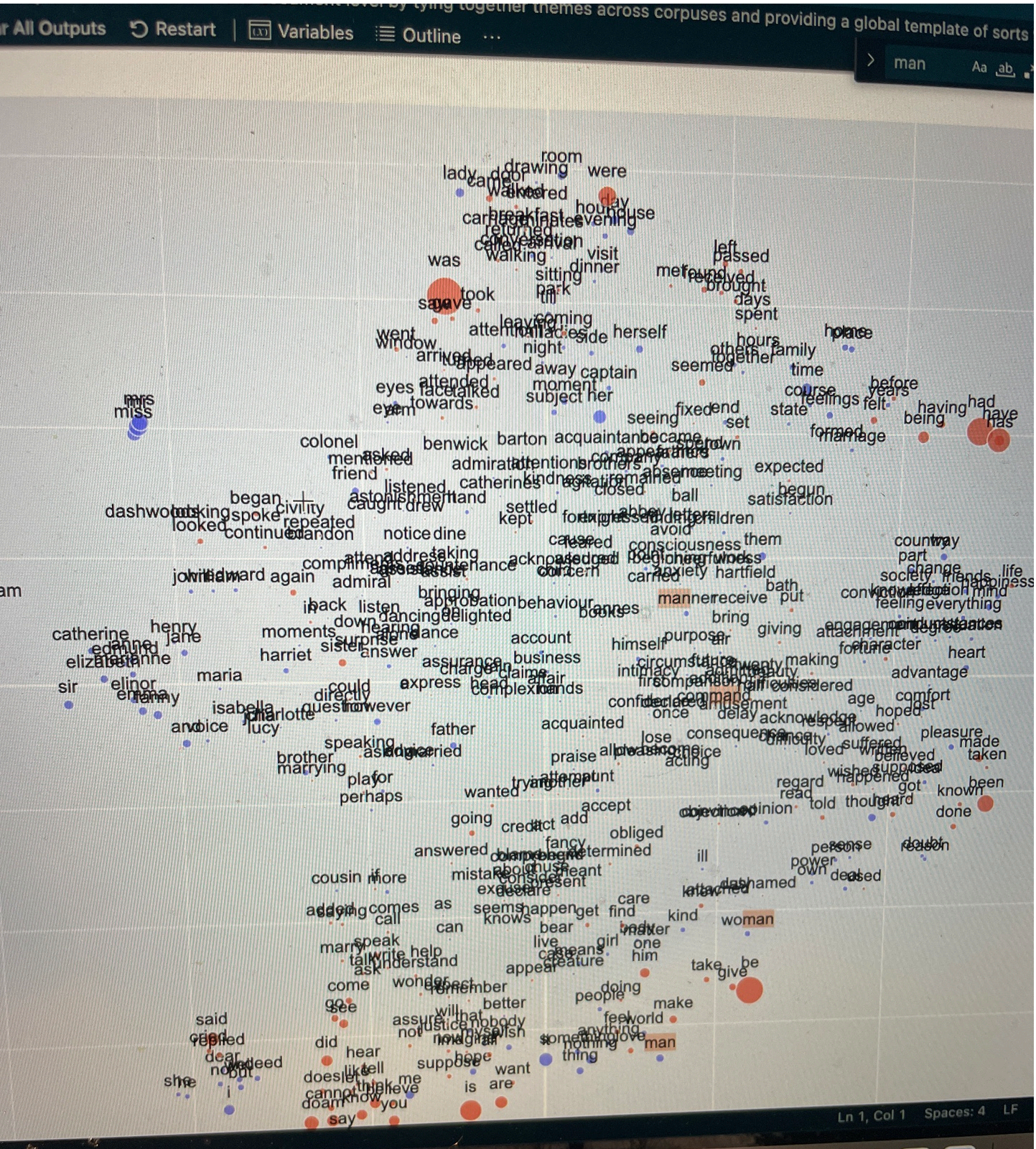
Out[ ]:

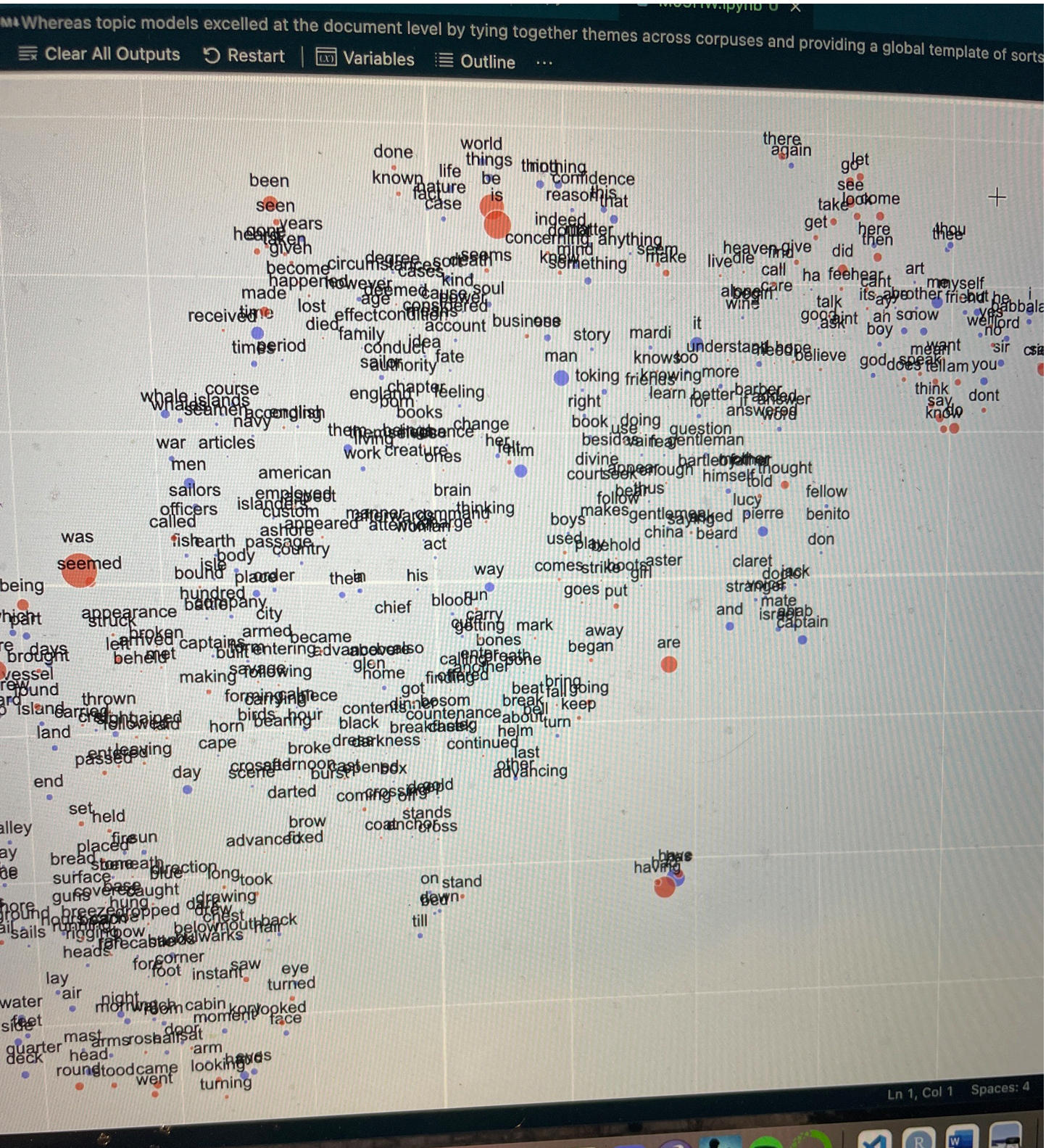
	term	sim
0	hours	0.946160
1	sailing	0.923042
2	distance	0.922004

Like for Austen, we try boys as one of the terms to produce an analogy but do not get similar words that result. Rather, we get more words that begin with B and describe actions. However, when we use the word ships in our analogy, we at least get words that can be used to describe sea travel such as 'hours', 'sailing', and 'distance'. Overall, however, I would say that our analogy production was less successful than our clustering based on what we would intuitively expect the results to be.

5.Consider the information that topic models provide in comparison to what word embeddings provide. How would you describe their differences? Might they complementary to each other?

Whereas topic models excelled at the document level by tying together themes across corpuses and providing a global template of sorts for documents, word embedding captures the structure and relationships of individual words within a corpus. Both of these techniques are interpretable, but word embeddings are more low-level whereas topic models are higher level. However, in theory, these techniques could be very complimentary when used in conjunction as word embedding would examine the word-level relationships whereas topic models can examine the relationships among documents to get more total information about a collection of documents.


r All Outputs
Restart
Variables
Outline
man
Aa
ab
Ln 1, Col 1
Spaces: 4
LF
MacBook Pro


Whereas topic models excelled at the document level by tying together themes across corpuses and providing a global template of sorts
Clear All Outputs
Restart
Variables
Outline
Ln 1, Col 1
Spaces: 4
MacBook Pro