Info

Name: Jacqui Unciano **ID:** jdu5sq **Assignment:** HW5

Directions

Using the notebook from this module (M05_01_BOW_TFIDF.ipynb) and the LIB and CORPUS tables generated from the collection of texts (Austen and Melville) in Module 4 (see the output directory from your env.ini file), create a notebook to perform the following tasks:

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly_express as px

sns.set()
```

In [2]:
```
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'

OHCO = ['book_id', 'chap_id', 'para_num', 'sent_num', 'token_num']
bags = dict(
    SENTS = OHCO[:4],
    PARAS = OHCO[:3],
    CHAPS = OHCO[:2],
    BOOKS = OHCO[:1]
)

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).drop
```

In [3]:
```
CORPUS.sample(5)
```

Out[3]:

book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str	term_str
21816	**78**	**3**	**0**	**25**	('and', 'CC')	CC	and	and
105	**11**	**2**	**1**	**55**	('in', 'IN')	IN	in	in
13720	**35**	**15**	**2**	**2**	('had', 'VBD')	VBD	had	had
141	**11**	**16**	**1**	**2**	('contrary,', 'NN')	NN	contrary,	contrary
105	**8**	**17**	**0**	**12**	('smiling.', 'NN')	NN	smiling.	smiling

In [4]:
```
LIB.sample(5)
```

Out[4]:

book_id	source_file_path	author	title	chap_rege
121	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	NORTHANGER ABBEY	^CHAPTER\s+\d+
1212	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	LOVE AND FREINDSHIP SIC	^\s*LETTER .* t .*
158	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	EMMA	^\s*CHAPTER\s [IVXLCM]+\s*
161	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	AUSTEN, JANE	SENSE AND SENSIBILITY	^CHAPTER\s+\d+
13720	/Users/jacqu/OneDrive/Documents/MSDS-at-UVA-20...	MELVILLE, HERMAN	MARDI AND A VOYAGE THITHER VOL I	^\s*CHAPTER\s [IVXLCM]+\s*

In [5]:
```
ab_dict = {key: group.index.tolist() for key, group in LIB.groupby('author')}
authors = []
for id in CORPUS.index.get_level_values(level=0):
    if id in ab_dict["AUSTEN, JANE"]:
        authors.append("AUSTEN, JANE")
    else:
        authors.append("MELVILLE, HERMAN")

CORPUS_1 = CORPUS.copy()
CORPUS_1['author'] = authors
CORPUS_1 = CORPUS_1.reset_index().set_index(['author']+OHCO)
CORPUS_1.head()
```

Out[5]:

author	book_id	chap_id	para_num	sent_num	token_num	pos_tuple	pos	token_str
AUSTEN, JANE	**105**	**1**	**1**	**0**	**0**	('Sir', 'NNP')	NNP	Sir
					1	('Walter', 'NNP')	NNP	Walter
					2	('Elliot,', 'NNP')	NNP	Elliot,
					3	('of', 'IN')	IN	of
					4	('Kellynch', 'NNP')	NNP	Kellynch

Write a function to generate a bag-of-words (BOW) representation of the CORPUS table (or some subset of it) that takes the following arguments:

1. A tokens dataframe which can be a filtered version of the dataframe you import. This will be the CORPUS table or some subset of it.
2. A choice of bag, i.e. OHCO level, such as book, chapter, or paragraph.

In [6]:
```
def bow(DF, bag):
    BOW = DF.groupby(bags[bag]+['term_str']).term_str.count().to_frame('n')
    return BOW
```

Write a function that returns the TFIDF values for a given BOW, with the following arguments:

1. The BOW table.
2. The type of TF measure to use.

To compute IDF, use the formula log2(N/DF) where N where is the number of documents (aka 'bags') in your BOW.se.

In [7]:
```
def tfidf(BOW, tf_method):
    DTCM = BOW.n.unstack(fill_value=0)

    if tf_method == 'sum':
        TF = DTCM.T / DTCM.T.sum()
    elif tf_method == 'max':
        TF = DTCM.T / DTCM.T.max()
    elif tf_method == 'log':
        TF = np.log2(1 + DTCM.T)
    elif tf_method == 'raw':
        TF = DTCM.T
    elif tf_method == 'double_norm':
        TF = DTCM.T / DTCM.T.max()
```

```
    elif tf_method == 'binary':
        TF = DTCM.T.astype('bool').astype('int')
    TF = TF.T

    DF = DTCM.astype('bool').sum()

    N = DTCM.shape[0]
    IDF = np.log2(N / DF)

    TFIDF = TF * IDF
    return TFIDF
```

You will need to generate your own VOCAB table from CORPUS and compute max_pos. When generating your own VOCAB table from CORPUS, be sure to name your index term_str.

VOCAB = CORPUS_1.groupby(level=0)['term_str'].value_counts().to_frame('n') VOCAB.index.names = ['author', 'term_str'] VOCAB['max_pos'] = CORPUS_1.groupby(level=0)[['term_str','pos']].value_counts().unstack(fill_value=0).idxmax(1) VOCAB.head()

Use these functions to get get the appropriate TFIDF to answer the following questions (or perform the task):

Question 1

Show the function you created.

Answer 1

In [8]:
```
bow_ex = bow(CORPUS, 'BOOKS')
bow_ex.sample(5)
```

Out[8]:

		n
book_id	**term_str**	
8118	**compelled**	1
4045	**crept**	5
8118	**varro**	1
2701	**darmonodes**	1
10712	**rivet**	1

In [9]:
```
tfidf_ex = tfidf(bow_ex, 'sum')
tfidf_ex.sample(5)
```

Out[9]:

term_str book_id	0	1	10	100	1000	10000	10000000	10440	10800	10th	...	zoroas
946	0.0	0.0	0.000097	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
121	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
141	0.0	0.0	0.000042	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
1900	0.0	0.0	0.000021	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
15422	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

5 rows × 40281 columns

Question 2

What are the top 20 words in the corpus by TFIDF mean using the `max` count method and `book` as the bag?

Answer 2

In [10]:
```
bow_df = bow(CORPUS, 'BOOKS')
df = tfidf(bow_df, 'max').stack().to_frame('n').reset_index().set_index(['book_id',
df.sort_values(by='n', ascending=False).head(20)
```

Out[10]:

book_id	term_str	n
161	elinor	0.642969
34970	pierre	0.586998
946	vernon	0.493614
158	emma	0.399292
161	marianne	0.388236
1342	darcy	0.366742
946	reginald	0.351225
	frederica	0.341733
141	crawford	0.337235
105	elliot	0.324016
158	weston	0.315232
13721	babbalanja	0.313431
141	fanny	0.295380
15422	israel	0.292173
158	knightley	0.288490
121	tilney	0.262482
	catherine	0.259876
158	elton	0.259316
1342	bingley	0.252012
105	wentworth	0.243650

Question 3

What are the top 20 words in the corpus by TFIDF mean, if you using the `sum` count method and `chapter` as the bag? Note, because of the greater number of bags, this will take longer to compute.

Answer 3

In [11]:
```
bow_df = bow(CORPUS, 'CHAPS')
# tfidf(bow_df, 'sum').stack()
df = tfidf(bow_df, 'sum').stack().to_frame('n').reset_index().set_index(['book_id',
df.sort_values(by='n', ascending=False).head(20)
```

Out[11]:

			n
book_id	**chap_id**	**term_str**	
8118	**55**	**communion**	3.520373
21816	**39**	**hypothetical**	2.875236
8118	**29**	**sailors**	2.315854
	55	**confidential**	2.269123
21816	**29**	**boon**	2.134439
8118	**20**	**elephants**	1.962914
	18	**dream**	1.935411
	43	**charmers**	1.725142
21816	**43**	**charming**	1.701658
2701	**125**	**um**	1.651731
21816	**45**	**increases**	1.642134
8118	**30**	**guide**	1.641756
21816	**8**	**charitable**	1.617706
8118	**6**	**slushing**	1.535112
21816	**4**	**renewal**	1.510187
	19	**soldier**	1.421777
8118	**1**	**bred**	1.411130
	43	**adorable**	1.408149
	18	**book**	1.343555
21816	**45**	**seriousness**	1.280663

Question 4

Characterize the general difference between the words in Question 3 and those in Question 2 in terms of part-of-speech.

Answer 4

To me, it looks like there are a lot of named entities (names specifically--proper nouns?) for question 2 whereas for question 3, I don't see any names but a whole bunch of different POS. I don't know how else to describe it, it's a bunch of different ones...

Question 5

Compute mean `TFIDF` for vocabularies conditioned on individual author, using *chapter* as the bag and `max` as the `TF` count method. Among the two authors, whose work has the most significant adjective?

In [12]:
```
df1 = CORPUS_1.query('author=="AUSTEN, JANE" & pos=="JJ"')
df2 = CORPUS_1.query('author=="MELVILLE, HERMAN" & pos=="JJ"')
```

In [13]:
```
# df1.head()
df2.head()
```

Out[13]:

						pos_tuple	pos	token_str
author	**book_id**	**chap_id**	**para_num**	**sent_num**	**token_num**			
MELVILLE, HERMAN	**1900**	**1**	**1**	**1**	**11**	('land;', 'JJ')	JJ	land;
					15	('sperm', 'JJ')	JJ	sperm
					16	('whale', 'JJ')	JJ	whale
					31	('wide', 'JJ')	JJ	wide
				2	**5**	('fresh', 'JJ')	JJ	fresh

In [14]:
```
bow_df = bow(df1, 'CHAPS')
tfidf(bow_df, 'max').stack().groupby('term_str').mean().sort_values(ascending=False
# df = tfidf(bow_df, 'max').stack().to_frame('n').reset_index().set_index(['book_id
# df.groupby(level=2).mean().sort_values(by='n', ascending=False).head()
```

Out[14]:

	n
term_str	
sure	0.146418
dear	0.145220
old	0.139177
lady	0.136585
poor	0.128352

In [15]:
```
bow_df = bow(df2, 'CHAPS')
tfidf(bow_df, 'max').stack().groupby('term_str').mean().sort_values(ascending=False
# df = tfidf(bow_df, 'max').stack().to_frame('n').reset_index().set_index(['book_id
# df.groupby(level=2).mean().sort_values(by='n', ascending=False).head()
```

Out[15]:

term_str	n
thy	0.229047
old	0.226575
little	0.203486
good	0.192374
such	0.186745

Answer 5

The author with the most significant adjective is Melville with the adjective 'thy'.