Final Project
DS 5001
Rehan Merchant: rm2bt
8/14/2021

In order to create a Digital Critical Edition of a Corpus, I created a library of long-form texts. The purpose of creating the Digital Critical Edition of a Corpus was to explore the cultural content of the corpus. While exploring the corpus, we focused on sentiments of the corpus. Analyzing the corpus for sentiment gives us an idea of how the tone changes throughout the books. We can see where the tone the author used is more positive in some areas and more negative in others.

The long-form texts I used to perform this analysis were found on Project Gutenberg. The library consisted of Mansfield Park by Jane Austen, Northanger Abbey by Jane Austen, and Pride and Prejudice by Jane Austen. The three books were downloaded and stored in a folder called epubs. I loaded the three csv files into Python and stored them in their source formats. Once they were loaded in they were transformed into a set of tables that conform to the Standard Text Analytic Data Model and to the Machine Learning Corpus Format. The OHCO definition that we used to Tokenize the data was book id, chapter number, paragraph number, sentence number, and token number.

```
OHCO = ['book_id','chap_num', 'para_num', 'sent_num', 'token_num']
epub_dir = 'epubs'
```

The first step in getting the documents into a Tokenized table is to create a table named lib and doc. The lib table consists of the book id, book title and the file location of each book. The doc table consists of book id, chapter number, paragraph number and the paragraph string. This table is my first step to getting our corpus into the token form. I created these two tables using a function that extracts elements from each of the documents and stores them in pandas dataframes. The function also cleans the files so that everything is stored neatly.

Once I got the doc table set up, I created another function called tokenize and that is used to transform the doc table into the token table. Getting the texts into this format is very important since the token table is the basis of all the analysis that we did on this corpus. The token table is broken up into 8 columns.

```
TOKEN.sample(25)
```

book_id	chap_num	para_num	sent_num	token_num	pos_tuple	pos	token_str
42671	54	29	0	10	(to, TO)	TO	to
	44	10	0	43	(or, CC)	CC	or
	54	27	0	20	(be, VB)	VB	be
	18	70	0	22	(her, PRP$)	PRP$	her
121	54	2	7	37	(different, JJ)	JJ	different
141	95	30	10	4	(astonished, JJ)	JJ	astonished
	82	25	1	1	(myself,, NN)	NN	myself,
42671	42	3	3	2	(consequently, RB)	RB	consequently
141	66	4	1	45	(all, DT)	DT	all
	91	4	4	27	(the, DT)	DT	the
42671	36	6	2	25	(his, PRP$)	PRP$	his
121	38	7	2	23	(any, DT)	DT	any
		1	1	52	(however, RB)	RB	however
141	53	39	2	7	(the, DT)	DT	the
121	51	1	0	30	(own, JJ)	JJ	own
	33	5	1	24	(those, DT)	DT	those
42671	6	48	1	1	(mind, NN)	NN	mind
	43	58	2	7	("when, IN)	IN	"when
121	49	19	4	15	(judge, VB)	VB	judge
141	62	16	0	6	(wife,", NN)	NN	wife,"
	83	21	8	8	((as, JJ)	JJ	(as
121	51	5	7	6	(circumstanced, VBN)	VBN	circumstanced
42671	19	8	3	10	(you, PRP)	PRP	you
	43	52	3	9	(last, JJ)	JJ	last
121	55	10	1	5	(be, VB)	VB	be

After creating the token table, I then created the vocab table. This table consisted of 4 columns; term id, term string, n and num. After creating the vocab table, we imported stop words from the nltk library and coded words that overlapped between the stop words and the vocab table as a 1. This column was then added to the vocab table. Now that we have coded our stop words, I used the Porter Stemmer to create a new column of just the porter stem of each word. This new column was also added to the vocab table. The last transformation that I made to the vocab table included adding the tokenized term strings to the vocab table. Our new vocab table had columns consisting of term id, term string, n, num, stop, porter stem and position max. After all these transformations, the last step was to save doc, lib, vocab and token table as csvs back to the directory for further use.

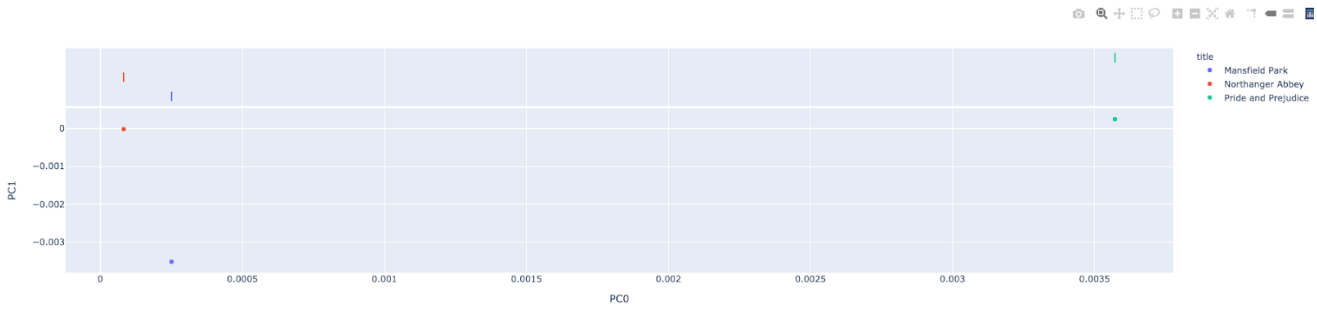
term_id	term_str	n	num	stop	p_stem	pos_max
0		922	0	0		NN
1	1	1	1	0	1	JJ
2	10	3	1	0	10	NN
3	14th	2	1	0	14th	CD
4	15th	1	1	0	15th	CD
...	...	...	...	...	...	...
11878	youths	1	0	0	youth	NNS
11879	youwill	1	0	0	youwil	VB
11880	zeal	5	0	0	zeal	NN
11881	zealous	1	0	0	zealou	JJ
11882	à	1	0	0	à	NNP

11883 rows × 6 columns

Vocab table

Now that the token table has been created, my next step is to transform the data into a vector representation of the corpus. This involves generating a table that we will name TFIDF, since this table will include the TFIDF values. The TFIDF function assigns a value to a term according to its importance in a document scaled by its importance across all documents in my corpus, which mathematically eliminates naturally occurring words in the English language, and selects words that are more descriptive of your text.

After creating the TFIDF matrix, my next step in the process was to apply the corpus to various unsupervised methods of analysis. I chose to analyze the corpus using Principal Component Analysis (PCA), Latent Dirichlet Allocation (LDA) and Word2Vector. These 3 methods provided a deeper understanding of the corpus and allowed me to explore the corpus at a deeper level. The first method of analysis I did was PCA. PCA is a linear algebraic technique that creates a new matrix with a reduced set of features. We are projecting the matrix onto a reduced subspace and PCA works by identifying the axes of maximum variance, therefore the most information.



The graph above shows the 3 books plotted by the first and second principal components, PC0 and PC1 respectively. This deeper understanding provided by this analysis shows which books are more similar by the language used in them. Based on the graph we can see that Mansfield Park and Northanger Abbey have similar language used throughout the book compared to Pride and Prejudice which is not grouped with the other two books.

The next analysis I did was LDA. LDA is a type of model that falls under Topic models. Topic models are a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. In this case we are looking at our corpus of texts and trying to identify topics and classify them. In order to perform LDA analysis, I needed to do some data prep. The model works with an F1 style corpus, so I transformed the token table to the F1 format. The F1

format is a table with columns book id, chapter number, paragraph number, and paragraph string.

```
PARAS.head()
```

book_id	chap_num	para_num	para_str
121	31	0	note
		1	advertisement
		2	work year publication bookseller business fart...
	32	1	one infancy born heroine situation character f...
		2	ten appearances hair balls complexion improved...

The next step was to convert the F1 corpus of paragraphs into a document-term vector space of word counts. I then ran Scikit Learn's Latent Dirichlet Allocation algorithm and extracted the theta and phi tables. Theta represents how much each document prefers a topic and phi represents how much each topic prefers a word.

```
THETA.sample(20).style.background_gradient()
```

Python

book_id	chap_num	para_num	topic_id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
141	90	12		0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.344444	0.011111	0.344444	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111	0.011111
42671	18	25		0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.516667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667
121	53	26		0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.806667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667	0.006667
42671	16	50		0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.385449	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778
121	56	39		0.593348	0.001754	0.001754	0.001754	0.001754	0.357529	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754	0.001754
42671	18	57		0.001515	0.001515	0.001515	0.956061	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515	0.001515
141	77	15		0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333
42671	53	51		0.003333	0.003333	0.003333	0.003333	0.003333	0.213359	0.003333	0.003333	0.003333	0.693307	0.003333	0.003333	0.003333	0.003333	0.003333	0.003333	0.003333	0.003333	0.003333	0.003333	0.003333	0.003333
141	63	1		0.959722	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389	0.001389
	57	26		0.004762	0.004762	0.322203	0.004762	0.004762	0.004762	0.544464	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762
42671	29	1		0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.002778	0.919444	0.002778	0.002778	0.002778	0.002778	0.002778
121	42	60		0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.516667	0.016667	0.016667	0.016667	0.016667
42671	55	50		0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.508333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333	0.008333
141	50	11		0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.723400	0.143266	0.004762	0.004762	0.004762
	80	43		0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.016667	0.516667	0.016667	0.016667	0.016667	0.016667
42671	42	4		0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.341561	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222	0.002222
141	93	10		0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.964198	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235	0.001235
121	56	8		0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333
141	57	21		0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.861905	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762	0.004762
42671	20	24		0.002381	0.002381	0.002381	0.002381	0.353853	0.002381	0.002381	0.002381	0.002381	0.002381	0.002381	0.190410	0.002381	0.002381	0.002381	0.002381	0.002381	0.002381	0.002381	0.002381	0.002381	0.391452

```
PHI.T.head().style.background_gradient()
```

Python

topic_id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
term_str																								
abatement	0.033333	0.033333	0.033333	1.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	1.033333	0.033333	0.033333	0.033333	0.033333	0.033333	1.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333
abbey	2.254763	0.033333	0.033333	0.033333	2.368230	3.007307	0.033333	0.033333	0.033333	0.033333	2.215456	1.033333	0.033333	0.033333	3.067315	2.478092	0.614601	0.033333	0.033333	3.250129	1.698437	1.168513	0.033333	3.418664
abbeys	2.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333
abhorrence	0.033333	0.033333	0.033333	0.033333	1.480584	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	1.033333	0.033333	0.033333	1.033333	1.033333	0.033333	0.033333	0.033333	0.033333	0.033333	1.033333	0.033333	0.033333
abilities	1.033333	0.033333	3.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	0.033333	1.033333	0.033333	1.033333	0.033333	1.033333	0.033333	0.033333	0.033333	1.033333	0.033333	0.033333	0.033333	0.033333

Now that I have the theta and phi tables the next step is to get the top terms per topic. To do this we create a matrix of the top term strings by topic id. The Topics matrix is shown below.

TOPICS

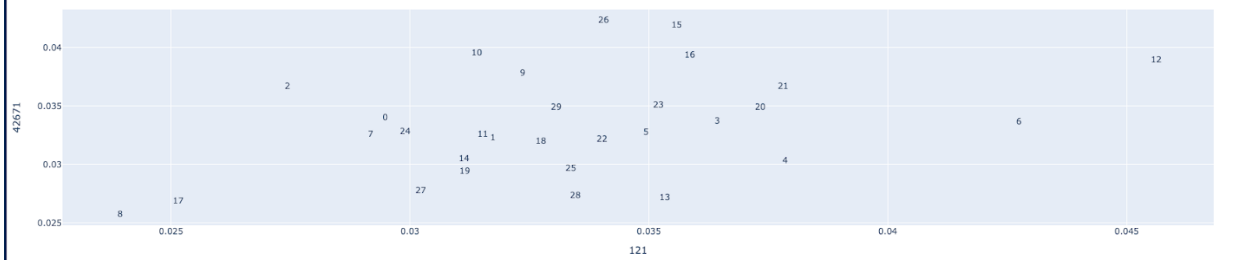
✓ 0.1s

term_str	0	1	2	3	4	5	6	7	8	9
topic_id										
0	opinion	feelings	time	nature	subject	pleasure	state	mind	heart	cousins
1	time	day	party	pleasure	house	question	family	hour	dinner	morning
2	family	country	character	mind	world	idea	fortune	house	marriage	sister
3	man	choice	comfort	husband	thing	time	daughter	father	gentleman	life
4	man	time	family	sister	feelings	happiness	home	opinion	day	life
5	acquaintance	friends	friend	home	family	pause	wish	kind	opinion	years
6	room	time	countenance	moment	man	mother	letter	day	beauty	way
7	distance	things	room	people	night	thing	miles	world	time	kind
8	time	home	rest	sister	heart	truth	necklace	room	people	sorry
9	place	time	family	sisters	plan	year	house	wife	people	ladies
10	time	friend	days	pleasure	spirits	visit	day	reason	way	family
11	oh	time	house	moment	room	theatre	door	sister	man	turn
12	yes	thing	sister	good	time	months	oh	moments	pleasure	away
13	look	man	woman	ladies	word	character	oh	sister	kind	family
14	madam	table	morning	house	servants	said	time	change	hour	subject
15	answer	time	day	pleasure	feelings	way	mother	oh	mind	word
16	sir	sisters	care	oh	sister	town	power	men	wonder	cousin
17	love	girl	sort	moment	dear	world	deal	thing	time	eyes
18	man	thing	horse	companion	play	friend	elizabeth	conversation	day	world
19	time	eyes	room	feelings	thing	object	mother	satisfaction	mind	heart
20	walk	smile	half	time	end	house	attention	place	behaviour	eyes
21	letter	reply	affection	brother	feelings	reason	head	trouble	oh	morning
22	dear	ball	place	evening	manner	room	day	aunt	time	hour
23	aunt	uncle	sister	pleasure	better	time	way	moment	room	did
24	friend	time	hope	father	day	home	morning	subject	sister	heart
25	room	uncle	moment	mind	door	brother	sister	house	time	friend
26	time	friend	day	letter	sister	uncle	father	brother	morning	character
27	way	house	time	evening	sister	news	days	minutes	pleasure	till
28	time	day	heart	surprise	word	letter	home	family	subject	sister
29	father	family	mind	house	attention	fathers	gentlemen	mother	time	subject

The next set of analysis we did was to create a table that showed all the top terms per topic by book id. This table would also show the average theta per term. Once I had this table I plotted the table and see how much each document prefers a topic.

```
px.scatter(BOOK.reset_index(), 121, 42671, hover_name='topterms', text='topic_id')\
    .update_traces(mode='text')
```

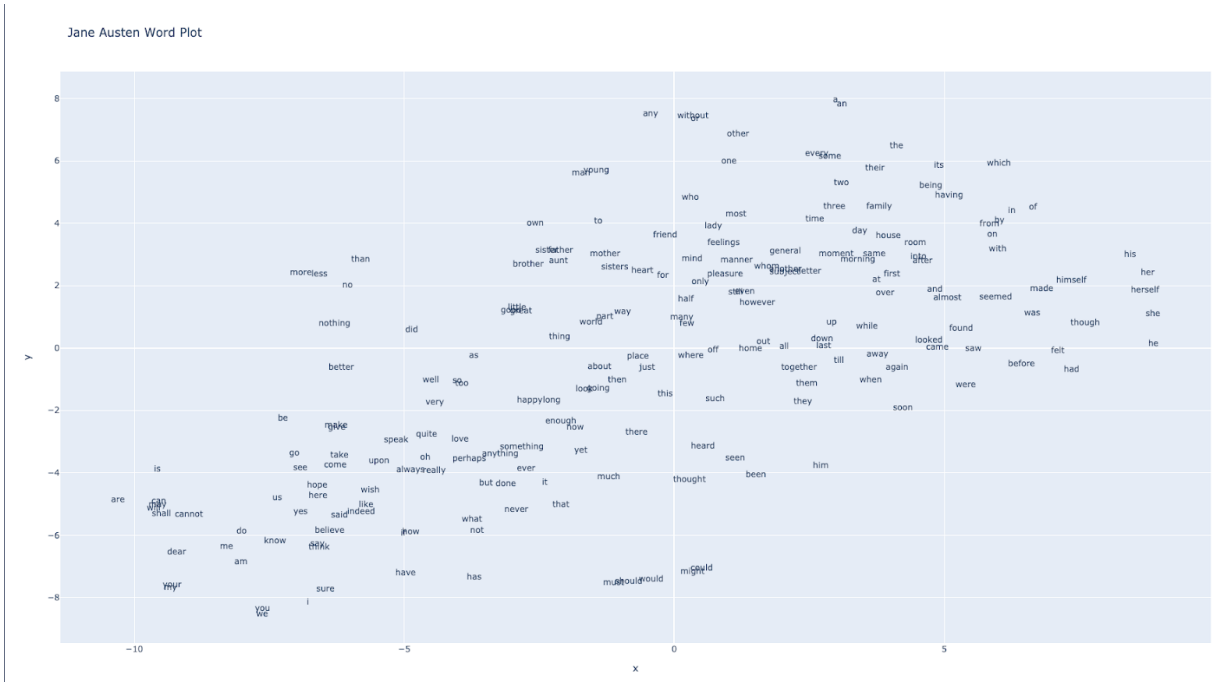
✓ 3.6s



This plot shows that both books prefer topic 12 whereas topic 2 is much more preferred by doc 42671 compared to doc 121.

The last model I used was the Word2Vec model. The Word2Vec model is a type of word embedding approach. The function uses one-hot encoding to label the word and their context as positive or negative. To perform this algorithm, the first step I did was to group the corpus of tokens by paragraphs, which I set as the bag. Once I passed the new corpus into the Word2Vec function, I stored the results of the model into a dataframe. In order to visualize the model, I had to put it into a TSNE function in order to get the coordinates to plot the model. The word plot is

shown below.



The plot shows which words are closely related to each other. We can see the clusters of pronouns near (8.7, 1.8). We also see another cluster of words that are about family members at (-1.3,3.1)

The last piece of analysis I performed was Sentiment analysis on the corpus. The first thing I did was to import the lexicon csv after setting up the OHCO and importing the tokenized corpus. Once everything was imported, we joined the token column to the salex table. The new token table is shown below.

```
TOKENS.head()
```

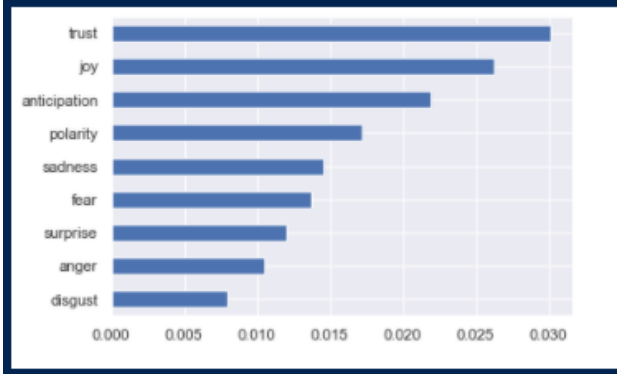
Python

author	title	chap_num	para_num	sent_num	book_id	token_num	pos_tuple	pos	token_str	term_str	term_id	book_title	year	anger	anticipation	disgust	fear	joy	negative	positive	sadness	surprise	trust	polarity
austen	Mansfield Park	49	1	0	141	0	('About', 'IN')	IN	About	about	44	Mansfield Park, by Jane Austen	1814	0.0	0.0	0.0	0.0	0.0	NaN	NaN	0.0	0.0	0.0	0.0
				0	141	1	('thirty', 'CD')	CD	thirty	thirty	10554	Mansfield Park, by Jane Austen	1814	0.0	0.0	0.0	0.0	0.0	NaN	NaN	0.0	0.0	0.0	0.0
				0	141	2	('years', 'NNS')	NNS	years	years	11832	Mansfield Park, by Jane Austen	1814	0.0	0.0	0.0	0.0	0.0	NaN	NaN	0.0	0.0	0.0	0.0
				0	141	3	('ago', 'RB')	RB	ago	ago	312	Mansfield Park, by Jane Austen	1814	0.0	0.0	0.0	0.0	0.0	NaN	NaN	0.0	0.0	0.0	0.0
				0	141	4	('Miss', 'NNP')	NNP	Miss	miss	6704	Mansfield Park, by Jane Austen	1814	0.0	0.0	0.0	0.0	0.0	NaN	NaN	0.0	0.0	0.0	0.0

The next three graphs show how the different emotions were broken out by book in a bar graph.

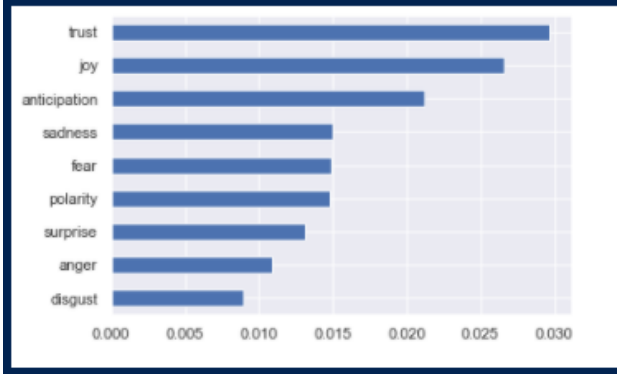
```
MANSFIELD[emo_cols].mean().sort_values().plot.barh()
```

```
<AxesSubplot:>
```



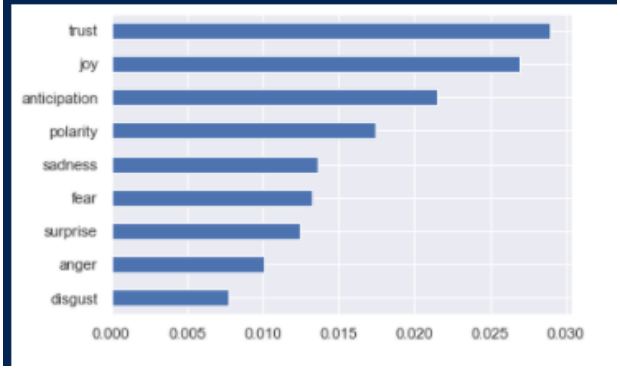

```
ABBEY[emo_cols].mean().sort_values().plot.barh()
```

```
<AxesSubplot:>
```




```
PRIDE[emo_cols].mean().sort_values().plot.barh()
```

```
<AxesSubplot:>
```



I also analyzed the sentiments of the corpus by chapters. I used the VADER sentiment analyzer package to perform the analysis. The graphs below show how the sentiments changed throughout the course of the book.

Based on the analysis done above I have a deeper understanding of the three novels in our corpus. Even though they were all written by the same author and two of the novels were published within one year of the other, we can see how different these three novels are but we also see areas where they are very similar. PCA analysis showed us that Mansfield Park and Northanger Abbey have similar language used throughout the book compared to Pride and Prejudice. LDA showed us that a lot of the topics within the books were seen in all of them; however, there were some unique topics that were more prevalent in some of the novels. The Word2Vec analysis showed how the words within the corpus were grouped together. This gave us an insight into the vocabulary used by the author and how it was representative of the time period that books were written in. The last analysis was the sentiment analysis which is what the

other pieces of analysis were leading up to. The sentiment analysis used what we built in the other models and showed us how the tone of each novel was changing throughout the novel. Overall, the skills that we acquired throughout this course were set on full display in the analysis of this corpus and helped gain a deeper understanding of the corpus.