Final Project - Analysis of Anti-Catholicism Sentiment in Literature from 1820-1865 - by Daniel Rusk (dgr73)

1. Introduction and hypothesis:

My question for this analysis stems from a excerpt from the Norton Anothology of American Literature (Herschel Parker, in Norton Anothology of American Literature, 2nd ed., Baym et al., eds., vol. 1, 1985, pp. 691-708). There is one section that discusses anti-catholicism during the time period 1820-1865. In the section discussing immigration and xenophobia, the book talks about how most Americans (especially protestants) were very anti-catholic as all the immigrants coming from Ireland and elsewhere during the time period from 1820-1865 were Catholic and Americans were very xenophobic. In fact, on page 703, the reading talks about how the country was saturated with anti-catholic books and pamphlets. Moreover, page 704 reads, "an extreme of xenophobia was reached in the summer of 1844, when rioters in Philadelphia (the city everyone pointed out, of brotherly love) burned Catholic churches and a seminary". Once I read this, I wondered if throughout the time period 1820-1865, as more and more Catholic immigrants came into the country, anti-catholicism grew in America and ultimately surfaced itself in literature of the time, and so I came up with my question.

Question:

Was there a significant increase in anti-catholic sentiment in literature during the time period 1820-1865?

Hypothesis:

My hypothesis is that there was an increase in anti-catholic sentiment in literature over the time period 1820-1865. I trust Parker's claim that there was rampant anti-catholicism during this time period. I also believe that this sentiment would either conciously or subconciously make its way into the literature of this time period. I believe that the anti-catholicism sentiment would increase during this time period because more and more immigrants flooded into the country during this time period and I feel that would cause the general public and writers to be increasingly more biased against catholics and catholicism.

Importance to Literary Studies:

This question is interesting in of itself for multiple reasons. One, it will show if the general public's opinions and ideas bleed into the literature of that time period. Two, it will show if there was any change in anti-catholic sentiment in literature over the mentioned time period. Moreover, if I do find that anti-catholicism ideas can be found in the literature of this time period, it would beg the question: what other general public views have seeped into historical literature of other time periods and could be analyzed by using this same method.

2. Corpus, data, and methods:

For this analysis, I used the corpus that was given to us for this project consisting of 1,540 volumes of American fiction published between 1789 and 1875. I also used the metadata file that accompanied the corpus. This was used to select only novels from certain time periods.

Now, before I get into exactly what I did, I would like to provide a brief overview of the steps I will take for this analysis as well as justify the decisions I made for each step in terms of approach and methods used.

Steps for Analysis:

- Split corpus into 3 intervals across the time period 1820-1865. Why do this? This is the time period I am focussing in on and I chose three intervals so I could see how my mean sentiment values change over time rather than having just one mean for the whole time period with which I cannot compare to anything else.
- Select only the novels that have between 50,000 and 150,000 words and have more than 5 catholic related words. Why do this? Remove the outliers from the group (novels with very few or very many words) and make sure that there is a sufficient amount of catholic related words with which I could conduct discrepancy sparsing.
- Get all words associated with the target catholic words for each time interval using spaCy discrepancy sparsing. Why? This is a great method to get associated words of a given word and it allows me to not only get the words that are directly before or after a word, but also the children of the words that come before or after a word.
- Preform sentiment analysis on the associated words of each time interval for the positive and negative sentiment. Why? I am using the simple emolex dictionary because I do not need to worry about sentence structure for this analysis as I already have a list of the words that really matter. Why include positive sentiment? I am including both negative and positive sentiments because this could show the the negative sentiment scores increase while the positive scores decrease. I thought it would be helpful to have more data as well to enhance my analysis.
- Graph the sentiment means of each of the three intervals on a bar and line chart (stacked bar chart, multi series line chart) On each interval, plot the mean positive and negative sentiment scores. Why? This is a great way to visualize any change in sentiment score over the intervals from the time period and an increase in sentiment scores would be very clearly shown.
- Preform a t-test on the diff between means from the first to second interval and the second to third interval. Why? This will tell us if there is statitistically significant increase between the mean pos or neg scores between intervals. This will also either prove or refute our hypothesis.

Now that I laid out the higher-level steps, we can begin the actual analysis. To start, I will import all the helpful libraries, imports, and metadata that I will need to complete my analysis.

```
# Imports
import os
import pandas as pd

# File locations
#   Note that metadata are supplied as a TSV file
#   Text files are in a directory, one file per (long, novel-like)
document
metadata_file = os.path.join('..', 'data', 'us_fiction',
'corpus_data.tsv')
text_dir      = os.path.join('..', 'data', 'us_fiction', 'us_texts')

# import spacy
import spacy

#import stopwords
from nltk.corpus import stopwords

#import sklearn
from sklearn.feature_extraction.text import CountVectorizer

#import default dict
from   collections import defaultdict

#imports for sentiment analysis
from   nltk import word_tokenize, sent_tokenize

# numpy for stats
import numpy as np

# seaborn for graphing, matplotlib for storing and showing graph later
import seaborn as sns
import matplotlib.pyplot as plt

# for t-test for significance
import statsmodels.stats.api as sms
from scipy import stats

# Load the metadata
metadata = pd.read_csv(
    metadata_file,
    sep='\t',
    low_memory=False
).set_index('source_id')
```

Now I will get the novel ids that correspond with each interval (1820-1835, 1835-1850, 1850-1865) and novels that have more than 50,000 words and less than 150,000. This is mainly to make computation easier and eliminate very long and very short novels. Also, I will create my list of words that are associated with catholicism. Note that I struggled creating the list because I needed to populate it with terms that are strictly catholic and not more generally christian, because my analysis only focusses in on words that are solely related to catholicism. I also create a basic stopwords list and a standard nlp from spaCy.

```
# get novels from time period 1820 - 1835
novels_interval_1 = metadata.loc[(metadata['pub_date'] < 1835) &
(metadata['pub_date'] >= 1820) & (metadata['words'] < 150000) &
(metadata['words'] > 50000)]

# get novels from time period 1835 - 1850
novels_interval_2 = metadata.loc[(metadata['pub_date'] < 1850) &
(metadata['pub_date'] >= 1835) & (metadata['words'] < 150000) &
(metadata['words'] > 50000)]

# get novels from time period  1850 - 1865
novels_interval_3 = metadata.loc[(metadata['pub_date'] <= 1865) &
(metadata['pub_date'] >= 1850) & (metadata['words'] < 150000) &
(metadata['words'] > 50000)]


# get novel titles from each subset of novels above
novel_int_1_ids = novels_interval_1.index
novel_int_2_ids = novels_interval_2.index
novel_int_3_ids = novels_interval_3.index

# list of catholic related words to pull from each novel
catholic_words = ["catholic", "catholicism", "pope", "anglo-catholic",
"holy father", "papal", "pontiff", "nun", "convent", "apostolic",
"rosary", "diocese", "bishop", ]

# list of stopwords
stops = stopwords.words('english')

# initialize nlp
nlp = spacy.load("en_core_web_lg")
```

Now I will create a few functions:

- get_ass_words will give me all the words associated with Catholic associated words in a given novel.
- check_text will give me the count of catholic related words in a given novel. I only want to use novels that have more than 5 catholic related words for this analysis. I chose this number because I do not want to analyze a novel if it only mentions the

target words once or twice as that would skew my results by being too small of a sample to analyze sentiment.

```
# function takes in a novel, nlp, and source_id and adds an entry to
the overall associated word dictionary in the format (source-id:
ass_word list)
def get_ass_words(novel, factory, name):
    doc = nlp(novel)
    ass_words = []
    for token in doc:
        if token.text in catholic_words:
            if token.text not in stops:
                children = [child for child in token.children if
child.text not in stops and child.text != "" and child.text != " " and
child.text != "," and child.text != "  "]
                if children != []:
                    ass_words.append(children)
    return ass_words

def check_text(text):
    vectorizer = CountVectorizer()
    matrix = vectorizer.fit_transform([text])
    counts = pd.DataFrame(matrix.toarray(),
                          columns=vectorizer.get_feature_names())
    catholic_word_count = 0
    for word in catholic_words:
        if word in vectorizer.get_feature_names():
            catholic_word_count += sum(counts[word])
    return catholic_word_count
```

Now I will filter out all novels that do not meet the requirement of more than 5 target words present for each interval and print the size of each interval to see if each has enough samples.

```
# initialize list of source id's to be populated with novels' source
ids that meet the below criteria
good_novels_int_1 = []
good_novels_int_2 = []
good_novels_int_3 = []

 # filter out novels that do not include catholic related words at
least 5 times
def get_good_novels(lst, ids):
    for novel in ids:
        with open (text_dir + "/" + novel, 'r') as f:
            text = f.read()
            if any(word in text for word in catholic_words):
                num = check_text(text)
                if num > 5:
                    lst.append(novel)
```

```
# populate each good novel list
get_good_novels(good_novels_int_1, novel_int_1_ids)
get_good_novels(good_novels_int_2, novel_int_2_ids)
get_good_novels(good_novels_int_3, novel_int_3_ids)


# print size of each interval now ready for spaCy dependency sparsing
print("Length of Interval 1: ", len(good_novels_int_1))
print("Length of Interval 2: ", len(good_novels_int_2))
print("Length of Interval 3: ", len(good_novels_int_3))
```

```
Length of Interval 1:  27
Length of Interval 2:  54
Length of Interval 3:  104
```

Now I create the function get_ass_dict to populate a dictionary for each interval consisting of the source id as key and all words associated with my target catholic terms in each novel in a given interval as the value. Then, I use the function to create a dictionary as described above for each interval.

```
# takes in list of novel source ids and returns a dictionary
containing (key: value) pair of (source-id: [words assoc. with
catholic related words from the text])
def get_ass_dict(novels):
    # init associated words
    ass_words = {}

    # iterate over each novel, run get ass_words
    for novel in novels:
        with open (text_dir + "/" + novel, 'r') as f:
            text = f.read()
            ass_words[novel] = get_ass_words(text, nlp, novel)
    return ass_words

interval_1_dict = get_ass_dict(good_novels_int_1)

interval_2_dict = get_ass_dict(good_novels_int_2)

interval_3_dict = get_ass_dict(good_novels_int_3)
```

Now I create the function unpack. This function will take a dictionary of a given interval and pull out each associated word for each novel and put them in a list. This is how I prime the data for the sentiment analysis.

```
# takes in a dictionary and returns a list of lists (one for each
novel) that are primed for sentiment analysis
def unpack(dict):
    total_ass_words = []
    for key in dict:
        ass_word_list = []
```

```
        lst = dict[key]
        flat_lst = [item for sublist in lst for item in sublist]
        for item in flat_lst:
            ass_word_list.append(item)
        total_ass_words.append(ass_word_list)
    return total_ass_words
```

This function is imported from Pset3 to assist with reading and parsing the emolex file.

```
# helper function to read and parse the emolex file (From Pset 3)
def read_emolex(filepath=None):
    '''
    Takes a file path to the emolex lexicon file.
    Returns a dictionary of emolex sentiment values.
    '''
    if filepath==None: # Try to find the emolex file
        filepath = '../data/lexicons/emolex.txt'
        if os.path.isfile(filepath):
            pass
        elif os.path.isfile('emolex.txt'):
            filepath = 'emolex.txt'
        else:
            raise FileNotFoundError('No EmoLex file found')
    emolex = defaultdict(dict) # Like Counter(), defaultdict eases
dictionary creation
    with open(filepath, 'r') as f:
    # emolex file format is: word emotion value
        for line in f:
            word, emotion, value = line.strip().split()
            emolex[word][emotion] = int(value)
    return emolex

# get EmoLex data
emolex = read_emolex()
```

Now I will create my sentiment scoring function. This function uses the given emolex from class to return sentiment scores for each word in the list of associated words for a given interval. One challenge I ran into was turning the spaCy token into a string that would be compatible with the emolex. Also, note that I only care about the positive and negative sentiment scores of each novel.

```
# get sentiment score for list of ass words
def sentiment_score(text, lexicon=None):
    # make list of just positive and negative sentiments
    select_sents = ["positive", "negative"]
    if lexicon == None:
        lexicon = read_emolex()
    sent_score = defaultdict(int)
    length = len(text)
    for word in text:
```

```
        # odd conversion from spacy token to actual string
representation so the sentiment analysis works
        word = word.orth_
        temp_sents = lexicon[word]
        for key in temp_sents:
            if key in select_sents:
                sent_score[key] += temp_sents[key]
    for key in sent_score:
        sent_score[key] = round(sent_score[key]/length, 3)

    return sent_score

# helper function to tokenize the words in a list of ass words
def tokenize_text(text):
    tokens = [word_tokenize(sent.lower()) for sent in
sent_tokenize(text)]
    return tokens
```

Now I will create my get_sent_score function. This will iterate over each novel's ass words in a given interval and return a list of the pos and neg scores of each novel.

```
# get sentiment score for each list of ass words for a given interval
list
def get_sent_score(lst):
    scores = []
    for words in lst:
        sent_scores = sentiment_score(words)
        scores.append(sent_scores)
    return scores
```

Now I will use the unpack and get_sent_score function to create a list of sentiment score dictionaries for each interval.

```
word_list_1 = unpack(interval_1_dict)
word_list_2 = unpack(interval_2_dict)
word_list_3 = unpack(interval_3_dict)

interval_1_sent_scores = get_sent_score(word_list_1)
interval_2_sent_scores = get_sent_score(word_list_2)
interval_3_sent_scores = get_sent_score(word_list_3)
```

Now I will create my function get_score_lists to pull the pos and neg sentiment score from each intervals sentiment score dictionaries. I use the try and except because some dictionaries are empty or missing a pos or neg key, so this avoids the potential error of accessing a key that does not exist.

```
# takes in an interval sent score and returns tuples of all the pos
and negative sent scores in that interval
def get_score_lists(interval):
    pos = []
    neg = []
```

```
    for item in interval:
        try:
            pos.append(item['positive'])
        except:
           pass
        try:
            neg.append(item['negative'])
        except:
            pass
    return (pos, neg)
```

Now I will get the score_lists of each interval and use numpy to get the mean and std of each interval and print the means and stds of each interval.

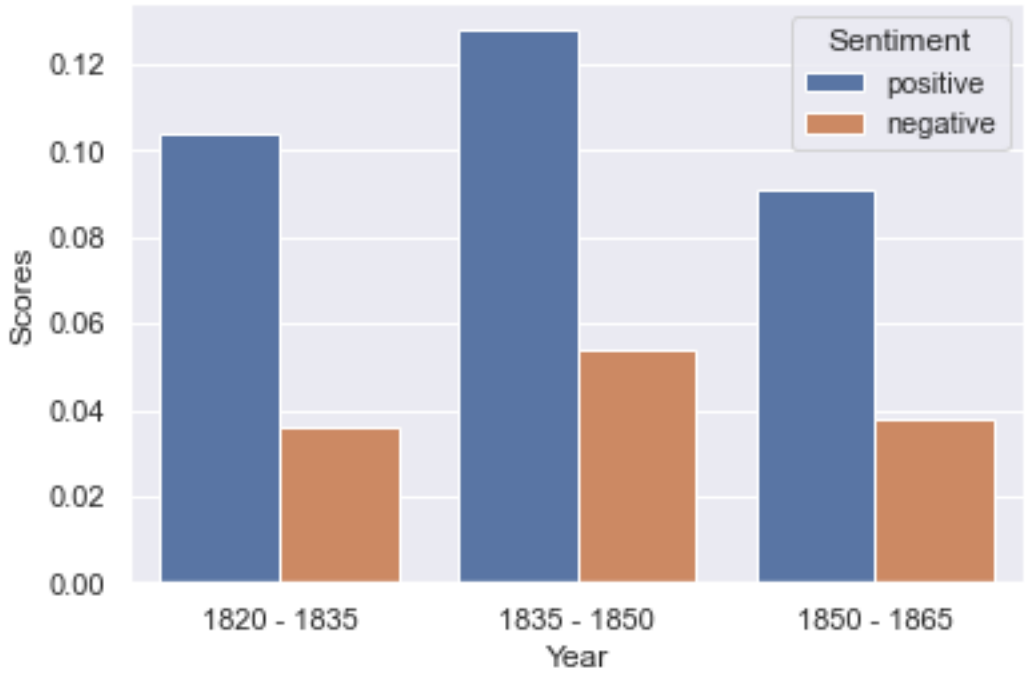
```
interval_1_scores = get_score_lists(interval_1_sent_scores)
interval_2_scores = get_score_lists(interval_2_sent_scores)
interval_3_scores = get_score_lists(interval_3_sent_scores)

int_1_pos_mean = np.mean(interval_1_scores[0])
int_1_pos_std = np.std(interval_1_scores[0])

int_2_pos_mean = np.mean(interval_2_scores[0])
int_2_pos_std = np.std(interval_2_scores[0])

int_3_pos_mean = np.mean(interval_3_scores[0])
int_3_pos_std = np.std(interval_3_scores[0])

int_1_neg_mean = np.mean(interval_1_scores[1])
int_1_neg_std = np.std(interval_1_scores[1])

int_2_neg_mean = np.mean(interval_2_scores[1])
int_2_neg_std = np.std(interval_2_scores[1])

int_3_neg_mean = np.mean(interval_3_scores[1])
int_3_neg_std = np.std(interval_3_scores[1])

print("int 1 pos mean: ", int_1_pos_mean, "int 2 pos mean: ",
int_2_pos_mean, "int 3 pos mean: ", int_3_pos_mean, "int 1 neg mean:
", int_1_neg_mean, "int 2 neg mean: ", int_2_neg_mean, "int 3 neg
mean: ", int_3_neg_mean)
print("\n")
print("int 1 pos std: ", int_1_pos_std, "int 2 pos std: ",
int_2_pos_std, "int 3 pos std: ", int_3_pos_std, "int 1 neg std: ",
int_1_neg_std, "int 2 neg std: ", int_2_neg_std, "int 3 neg std: ",
int_3_neg_std)
```

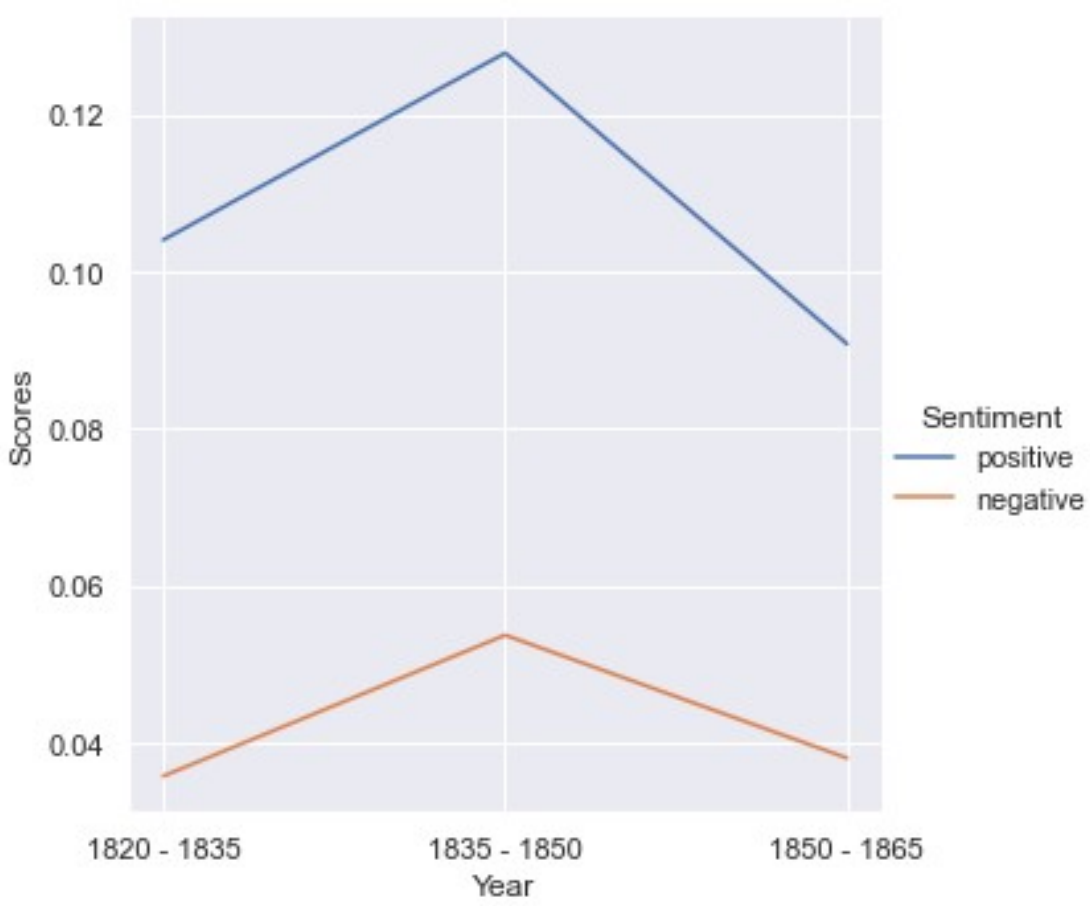
```
int 1 pos mean:  0.10407407407407408 int 2 pos mean:
0.1278888888888889 int 3 pos mean:  0.09082692307692307 int 1 neg
mean:  0.035777777777777776 int 2 neg mean:  0.05374074074074074 int 3
```

```
neg mean:  0.03807692307692308


int 1 pos std:  0.2081776175263745 int 2 pos std:  0.21214867835646548
int 3 pos std:  0.20441179413830368 int 1 neg std:
0.10025461412982939 int 2 neg std:  0.1265327047732288 int 3 neg std:
0.09791034874073601
```

Upon looking at the values, it seems there could be some differences so I will now graph them in a bar and line chart to see if this difference between means over the intervals are visible.

```
# create df to graph
df = pd.DataFrame({'Year': ['1820 - 1835', '1820 - 1835', '1835 -
1850', '1835 - 1850', '1850 - 1865', '1850 - 1865'],
                   'Scores': [int_1_pos_mean, int_1_neg_mean,
int_2_pos_mean, int_2_neg_mean, int_3_pos_mean, int_3_neg_mean],
                   'Sentiment': ['positive', 'negative', 'positive',
'negative', 'positive', 'negative']})

sns.set_theme(style="darkgrid")

#create grouped bar chart
bar = sns.barplot(x='Year', y='Scores', hue='Sentiment', data=df)

# create multi line chart
line = sns.relplot(data=df, x='Year', y='Scores', kind='line',
hue='Sentiment')
```





Now that I graphed them, you can visually see an interesting trend of the pos and neg scores both rising and falling throughout the 1820 - 1865 time period. Now I want to see if either the rise or fall for both pos and neg sentiment scores is statistically significant so I will run a series of t-tests using a helper function run_t_test which will produce the t-statistic and p-value for a difference in means analysis.

```
# runs t-test for diff in means between 1820-1835 and 1835-1850
def run_t_test(int_1, int_2, s):
    result = stats.ttest_ind(
        int_1,
        int_2,
        equal_var=False
    )
    print("T Test for " + s + "\n")
    print('t-statistic:', result[0])
    print('p-value:    ', result[1])
    print("\n")

t_test_1_neg = run_t_test(interval_1_scores[1], interval_2_scores[1],
"Difference in Neg Means for 1820-1835 and 1835-1850")
t_test_1_pos = run_t_test(interval_1_scores[0], interval_2_scores[0],
"Difference in Pos Means for 1820-1835 and 1835-1850")
```

```
T Test for Difference in Neg Means for 1820-1835 and 1835-1850

t-statistic: -0.6845023877700308
p-value:     0.4961490943659572


T Test for Difference in Pos Means for 1820-1835 and 1835-1850

t-statistic: -0.4747763175463156
p-value:     0.6369136876191238
```

```
t_test_2_neg = run_t_test(interval_2_scores[1], interval_3_scores[1],
"Difference in Neg Means for 1835-1850 and 1850-1865")
t_test_2_pos = run_t_test(interval_2_scores[0], interval_3_scores[0],
"Difference in Neg Means for 1835-1850 and 1850-1865")
```

```
T Test for Difference in Neg Means for 1835-1850 and 1850-1865

t-statistic: 0.7879750367644578
p-value:     0.43286620499493955


T Test for Difference in Neg Means for 1835-1850 and 1850-1865

t-statistic: 1.046238149518541
p-value:     0.29788668066826524
```

Now that my analysis is complete, I can move onto results.

3. Results

From my analysis there is statistical evidence that suggests there is not a statistically significant increase in mean neg and pos sentiment scores throughout the time period 1820-1865. This is indicated by the p-values from both of my t-tests being very large.

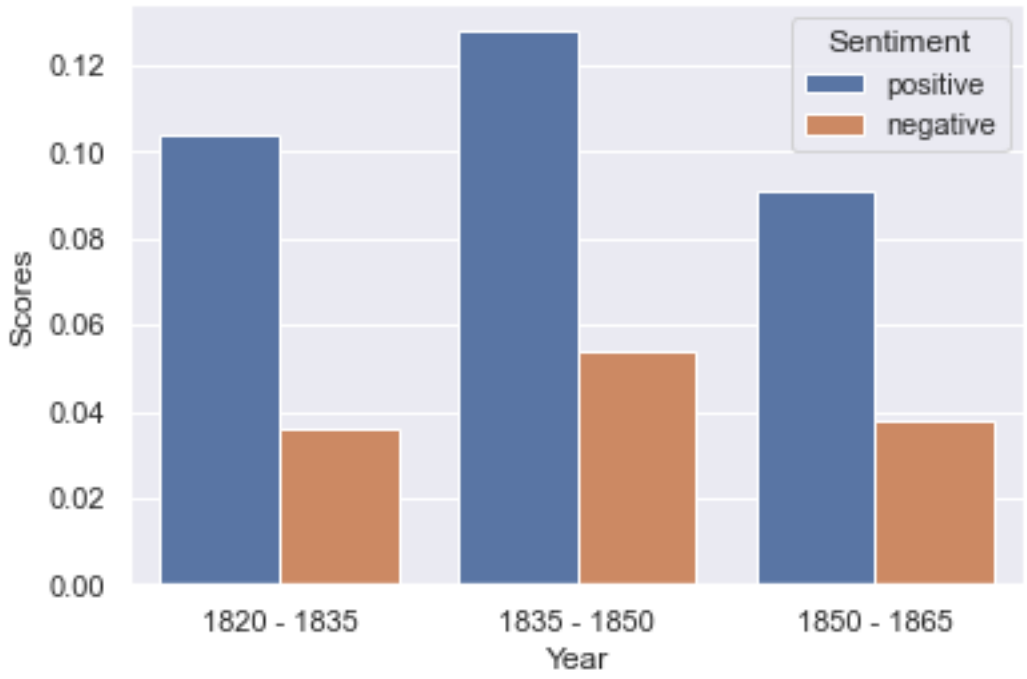
```
t_test_1_neg = run_t_test(interval_1_scores[1], interval_2_scores[1],
"Difference in Neg Means for 1820-1835 and 1835-1850")
t_test_1_pos = run_t_test(interval_1_scores[0], interval_2_scores[0],
"Difference in Pos Means for 1820-1835 and 1835-1850")
t_test_2_neg = run_t_test(interval_2_scores[1], interval_3_scores[1],
"Difference in Neg Means for 1835-1850 and 1850-1865")
t_test_2_pos = run_t_test(interval_2_scores[0], interval_3_scores[0],
"Difference in Neg Means for 1835-1850 and 1850-1865")
```

```
T Test for Difference in Neg Means for 1820-1835 and 1835-1850
```

```
t-statistic: -0.6845023877700308
p-value:     0.4961490943659572


T Test for Difference in Pos Means for 1820-1835 and 1835-1850

t-statistic: -0.4747763175463156
p-value:     0.6369136876191238


T Test for Difference in Neg Means for 1835-1850 and 1850-1865

t-statistic: 0.7879750367644578
p-value:     0.43286620499493955


T Test for Difference in Neg Means for 1835-1850 and 1850-1865

t-statistic: 1.046238149518541
p-value:     0.29788668066826524
```

This proves my hypothesis was not correct. I thought there would be a steady increase in negative sentiment score and steady decrease in positive sentiment score for catholic related words for the time period 1820-1865. I am glad that I did the t-test because the graphs for the positive and negative mean sentiment scores both visually suggested there was a significant difference. Interestingly enough, the visible trend was not a steady increase but rather an increase, peak, and decrease in both the neg and pos mean sentiment scores over the time period as seen below. Furthermore, it was interesting to see that in each interval the mean positive sentiment score was greater than the negative sentiment score.

```
bar = sns.barplot(x='Year', y='Scores', hue='Sentiment', data=df)
```



Regardless of the trend, the difference in means for both positive and negative scores between the first and second and second and third interval were both statistically insignificant. With that being said, onto the dicussion and conclusion.

4. Discussion and conclusions

As I stated above, my results did not support my hypothesis. From this analysis, I can conclude that anti-catholicism sentiment did not increase from 1820-1865.

I think this is because the forms of media that would typically carry the anti-catholicism sentiment were not novels. I think that perhaps magazines and newspapers would be more likely to include anti-catholicism sentiment. Books are political at times and make stands, but I think magazines and newspapers would more likely carry propaganda and messages speaking out against catholics and catholicism and reflect the religious Xenophobia of the time. In future work, I would love to look into corpuses of magazine and newspaper articles and conduct the same analysis to see if there are significant increases or decreases in negative and positive mean sentiment scores respectively for catholic related words in these types of works rather than just literature.

In terms of limitations, there is a chance that an increase in negative sentiment around catholicism in literature did occur, but my analysis just failed to capture the way in which the anti-catholic sentiment manifests itself in the novels. One limitation is that I only looked at the words surrounding catholic related words, however, catholicism could have been negatively portrayed in novels in a more between-the-lines type of way that this analysis did not capture. It could have been how the writer characterized catholic characters instead of the words that the writer put around catholic related words. This is to say, rather than the author describing the character in a negative way, the author makes the character

do negative things or act in ways that would make the reader have a negative attitude towards them. In future research, perhaps I would look into how to analyze how catholic characters are characterized in novels and see if there is a difference in the ways catholic and non-catholic characters are characterized.

Another limitation is that I did shrink down the corpus to only work with novels with 50,000 to 150,000 words. I did this so that computation would be quicker (the spaCy nlp was a very slow process) and to exclude very small and very large novels. Perhaps if I included more than just the 189 novels from 1820-1865 I did, then my analysis results would have been different. Moving forward, I would use a more powerful computer and process more novels over that same time period. Perhaps this would lead to a different conclusion than this analysis did.

Overall, it seems that either the anti-catholicism did not seep into literature of the time or my analysis was not the correct way to detect this sentiment and measure the way in which it changed over the time period 1820-1865. Regardless, my findings ultimately refute my hypothesis that there was an increase in anti-catholic sentiment in literature over the time period 1820-1865.